# ON THE EFFICIENCY OF CRYPTOGRAPHIC CONSTRUCTIONS

by

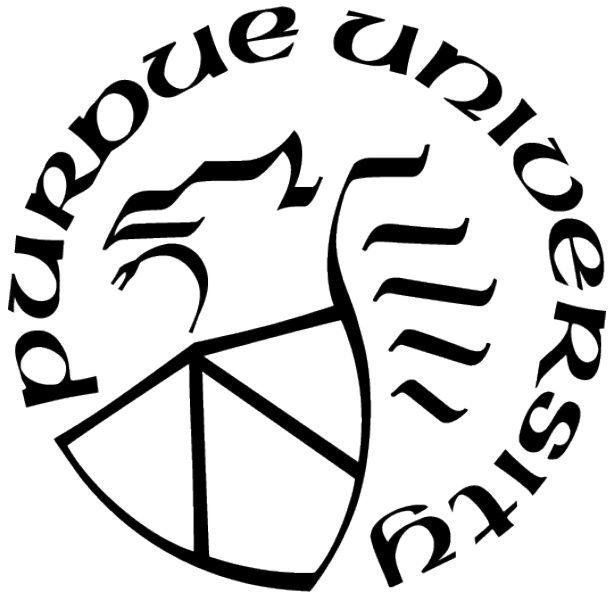
**Mingyuan Wang**

**A Dissertation**

*Submitted to the Faculty of Purdue University*

*In Partial Fulfillment of the Requirements for the degree of*

**Doctor of Philosophy**




Department of Computer Science

West Lafayette, Indiana

December 2021

# THE PURDUE UNIVERSITY GRADUATE SCHOOL
# STATEMENT OF COMMITTEE APPROVAL

**Dr. Hemanta K. Maji, Chair**

Department of Computer Science

**Dr. Mikhail Atallah**

Department of Computer Science

**Dr. Jeremiah Blocki**

Department of Computer Science

**Dr. Ninghui Li**

Department of Computer Science

**Dr. Mohammad Mahmoody**

Department of Computer Science, University of Virginia

**Approved by:**

Dr. Kihong Park

# ACKNOWLEDGMENTS

First and foremost, I would like to thank my advisor Hemanta Maji for his wise guidance throughout the years. Maji lured me into the beautiful world of cryptography with his inspiring lectures. Since then, he has been a constant source of exciting problems and insightful ideas. He trained me in all aspects of being a researcher, ranging from selecting research problems to presenting research findings. I am very grateful to him for everything.

Second, I thank my committee members Mikhail Atallah, Jeremiah Blocki, Ninghui Li, and Mohammad Mahmoody for their time reading this work.

I would also like to thank my co-authors: Donald Q. Adams, Divya Gupta, Hamidreza Amini Khorasgani, Himanshi Mehta, Hai H. Nguyen, Minh L. Nguyen, Anat Paskin-Cherniavsky, Tom Suad, Xiuyu Ye, and Albert Yu. I had countless fruitful discussions with them, and collaborating with them has been incredible. Furthermore, I would like to thank the crypto and theory groups at Purdue for all the friends I met there and make this journey the most fun.

Finally, I would not have been here without my parents, my wife Yunyi, and my entire family. I am forever indebted to them for their love and support.

# TABLE OF CONTENTS

# LIST OF FIGURES

# ABSTRACT

Cryptography allows us to do magical things ranging from private communication over a public channel to securely evaluating functions among distrusting parties. For the real-world implementation of these tasks, efficiency is usually one of the most desirable objectives. In this work, we advance our understanding of efficient cryptographic constructions on several fronts.

- Non-malleable codes are a natural generalization of error-correcting codes. It provides a weaker yet meaningful security guarantee when the adversary may tamper with the codeword such that error-correcting is impossible. Intuitively, it guarantees that the tampered codeword either encodes the original message or an unrelated one. This line of research aims to construct non-malleable codes with a high rate against sophisticated tampering families. In this work, we present two results. The first one is an explicit rate-1 construction against all tampering functions with a small locality. Second, we present a rate-1/3 construction for three-split-state tampering and two-lookahead tampering.

- In multiparty computation, fair computation asks for the most robust security, namely, guaranteed output delivery. That is, either all parties receive the output of the protocol, or no party does. By relying on oblivious transfer, we know how to construct MPC protocols with optimal fairness. For a long time, however, we do not know if one can base optimal fair protocol on weaker assumptions such as one-way functions. Typically, symmetric-key primitives (e.g., one-way functions) are much faster than public-key primitives (e.g., oblivious transfer). Hence, understanding whether one-way functions enable optimal fair protocols has a significant impact on the efficiency of such protocols. This work shows that it is impossible to construct optimal fair protocols with only black-box uses one-way functions. We also rule out constructions based on public-key encryptions and $f$-hybrids, where $f$ is any incomplete function.

- Collective coin-tossing considers a coin-tossing protocol among $n$ parties. A Byzantine adversary may adaptively corrupt parties to bias the output of the protocol. The security $\varepsilon$ is defined as how much the adversary can change the expected output of the protocol. In this work, we consider the setting where an adversary corrupts at most one party.

Given a target security $\varepsilon$, we wish to understand the minimum number of parties $n$ required to achieve $\varepsilon$-security. In this work, we prove a tight bound on the optimal security. In particular, we show that the insecurity of the well-known threshold protocol is at most two times the optimal achievable security.

# 1. INTRODUCTION

Cryptography enables mutually distrusting parties to accomplish diverse tasks without divulging secretive information. For applications of these cryptographic primitives in real-world scenarios, efficiency is one of the most desired features. For different tasks, one usually measures efficiency through various means.

For encryption/encoding schemes, the most crucial efficiency measure is the *rate* of the scheme. That is, we would like to minimize the ratio between the length of the message and the length of the ciphertext/codeword. This measure is evident as higher rates bring lower communication and storage costs. As another example, in the interactive setting, one would like to encode the protocol between parties into a new protocol such that even if an adversary may maliciously tamper a certain fraction of the interaction, parties could still realize the original protocol correctly. In this fascinating research field, namely *interactive coding* [Sch96], minimizing the rate between the communication cost of the original protocol and that of the encoded protocol is one of the most significant objectives [KR13, Hae14, GH15, BEGH16].

Another critical principle is the *computational hardness assumptions* required to realize cryptographic tasks. In cryptography research, we typically aim to build our primitives using minimal computational hardness assumptions. Intuitively, the weaker assumptions a construction assumes, the more reliable it shall be. Besides security concerns, founding cryptographic primitives on weaker assumptions often translate into higher efficiency as well. For example, (the stronger) public-key assumptions are typically more computational costly in practice than (the weaker) symmetric assumptions. Hence, to encrypt a long message using public-key encryption, it is preferable to first establish a (short) secret key using PKE and then encrypt the long message using this secret key under a private-key encryption scheme. In another example, to generate multiple instances of *oblivious transfer* correlation,[1] it is also preferable to first generate a few instances of oblivious transfer and then extend it through symmetric-key primitives. Such protocols are called *OT extension* protocols [Bea96, IKNP03, GMMM18].

[1]↑In an oblivious transfer correlation, Alice holds randomness $(c_0, c_1)$, and Bob holds randomness $(b, c_b)$. Such correlations are fundamental to secure function evaluation protocols.

Finally, in interactive protocols, *round complexity* is another notable measure of efficiency. This is motivated as network latency is usually several magnitudes higher than the local computation costs. Hence, the round complexity usually dictates the time cost of a cryptographic protocol, and a lot of research effort is devoted to finding round-optimal protocols. For example, in the field of multi-party computation, a gigantic body of literature [BMR90, KOS03, KO04, Pas04, PW10, Goy11, GMPP16, ACJ17, BHP17, COSV17b, COSV17a, BL18, GS18, CCG+20] studies the lower and upper bounds on the round complexity.

In this work, we continue the research on finding efficient constructions of various cryptographic primitives. In particular, we study the following primitives.

**Non-malleable Codes.** Non-malleable code [DPW10] is a primitive in tamper-resilient cryptography. Intuitively, it is a coding scheme with the security guarantee that a tampering function cannot tamper the encoding of a message $m$ into the encoding of a related message $m'$. In this work, we show how to construct non-malleable code against various tampering families with (near) optimal rates.

**Fair Multiparty Computation.** Fair multiparty computation facilitates multiple parties to evaluate functions over their respective inputs securely. It mandates the strongest security guarantee such that the honest parties shall always receive the output even when malicious parties prematurely abort. For the representative task of fair coin-tossing, we study the minimal computational assumptions required for achieving optimal fairness.

**Collective Coin-tossing.** A collective coin-tossing protocol [BL85] enables multiple parties to upgrade their private randomness into public randomness. A (computationally unbounded) Byzantine adversary may adaptively corrupt some parties and fix their messages arbitrarily. The insecurity is the maximum deviation an adversary can cause in the expected output of the protocol. Given a target insecurity $\varepsilon$, our work studies the minimum number of parties/rounds required to ensure $\varepsilon$-security.

## 1.1 Our Results

### 1.1.1 Non-malleable Codes

Consider a signature scheme with a signing function $\mathsf{Sign}$. The traditional security guarantees that an adversary cannot forge the valid signature of a message $m$ under the secret key $\mathsf{sk}$ even if she may query the signing function $\mathsf{Sign}(\mathsf{sk}, \cdot)$ with any message $m' \neq m$. However, a real-world adversary might not adhere to only using the signing function $\mathsf{Sign}(\mathsf{sk}, \cdot)$ as a black box. For example, it might tamper with the physical memory that stores the secret key $\mathsf{sk}$. Consequently, the memory might store a different secret key $\mathsf{sk}'$ and the adversary now gains access to the functionality $\mathsf{Sign}(\mathsf{sk}', \cdot)$. The traditional security of signature schemes provides no guarantee under such attacks. However, observe that if the adversary could only change the original secret key $\mathsf{sk}$ to some fixed secret key $\mathsf{sk}'$, then this attack is ineffective as querying $\mathsf{Sign}(\mathsf{sk}', \cdot)$ provides no information of $\mathsf{sk}$.

Non-malleable code [DPW10] is a primitive that provides an algorithmic solution against such tampering attacks. Intuitively, encoding the secret key using a non-malleable encoding ensures that the tampered codeword either encodes the original secret key $\mathsf{sk}$ or an unrelated one $\mathsf{sk}'$. Thus, the security of the signature scheme is restored even in the presence of tampering attacks.

Formally, we consider a coding scheme with a (possibly probabilistic) encoding function $\mathsf{Enc}\colon \{0,1\}^{\ell} \to \{0,1\}^{n}$ and a (deterministic) decoding function $\mathsf{Dec}\colon \{0,1\}^{n} \to \{0,1\}^{\ell} \cup \{\bot\}$. A tampering family $\mathcal{F}$ is an arbitrary collection of functions $f\colon \{0,1\}^{n} \to \{0,1\}^{n}$. For any message $m \in \{0,1\}^{\ell}$ and any tampering function $f$, consider the following tampering experiment.

$$\mathsf{Tamper}_{f}^{m} := \left\{ \begin{array}{c} c \leftarrow \mathsf{Enc}(m),\ \widetilde{c} = f(c),\ \widetilde{m} = \mathsf{Dec}(\widetilde{c}) \\ \\ \text{Output } \widetilde{m} \end{array} \right\}.$$

To ensure that $\mathsf{Tamper}_{f}^{m}$ is either the orignal message $m$ or an unrelated message $m'$, we require the existence of a simulator $\mathsf{Sim}(f)$. This simulator $\mathsf{Sim}(f)$ only takes the tampering function $f$ as input and outputs a distribution over $\{0,1\}^{\ell} \cup \{\bot, \mathsf{same}^{*}\}$. Intuitively, $\mathsf{Sim}(f)$ outputs $m' \in \{0,1\}^{\ell} \cup \{\bot\}$ if the tampering function $f$ replaces the encoding of $m$ to be the encoding of $m'$. On the other hand, $\mathsf{Sim}(f)$ outputs $\mathsf{same}^{*}$ if the tampering function $f$

does not change the encode message. Formally, it guarantees that the following statistical distance[2] is small.

$$\mathsf{SD}\left(\mathsf{Tamper}_f^m\ ,\ \mathsf{Copy}(\mathsf{Sim}(f), m)\right),$$

where $\mathsf{Copy}(x, y)$ outputs $y$ if $x = \mathsf{same}^*$ and output $x$ otherwise.

We say a coding scheme $(\mathsf{Enc}, \mathsf{Dec})$ is $(\mathcal{F}, \varepsilon)$-non-malleable if for all message $m \in \{0,1\}^\ell$ and tampering function $f \in \mathcal{F}$,

$$\mathsf{SD}\left(\mathsf{Tamper}_f^m\ ,\ \mathsf{Copy}(\mathsf{Sim}(f), m)\right) \leqslant \varepsilon.$$

Observe that it is impossible to achieve non-malleability for the tampering family $\mathcal{F}$ consisting of all tampering functions. To see this, consider the tampering function $f(c) := \mathsf{Enc}(\mathsf{Dec}(c) + 1)$. That is, this tampering function decodes to obtain the original message $m$, increases the message by 1, and finally re-encodes the new message $m+1$. Clearly, the tampering experiment with message $m$ and tampering function $f$ always outputs $m + 1$ (which is impossible to simulate). Therefore, non-malleability is only achievable for restricted tampering families.

Naturally, the quality of non-malleable codes can be measured through (1) the rate of the coding scheme (i.e., $\ell/n$) and (2) the sophistication of the tampering family. In this work, we consider the following three natural tampering families.

**Local Tampering.** A tampering function is $\delta$-local if every output bit depends on (at most) $\delta$ input bits. The $\delta$-local tampering family is the collection of all $\delta$-local tampering functions. We obtain the following results regarding non-malleable codes for local tampering.

**Informal Result 1.** *For any positive constant $\xi < 1$, there is an explicit rate-1 non-malleable code against $\xi \log_2 n$-local tampering functions.*

Since for any $\delta = \omega(1)$, the $\delta$-local tampering family is a superset of the $\mathsf{NC}^0$ tampering,[3] we also get the following corollary.

**Informal Result 2.** *There is an explicit rate-1 non-malleable code against $\mathsf{NC}^0$ tampering.*

[2]↑The statistical distance between two distributions $A$ and $B$ over the same (enumerable) sample space $\Omega$ is defined as $\frac{1}{2}\sum_{\omega\in\Omega} |\Pr[A = \omega] - \Pr[B = \omega]|$.

[3]↑$\mathsf{NC}^0$ is the class of functions that can be computed by constant-depth circuits with bounded fan-in.

**Three-Split-state Tampering.** In the $k$-split-state tampering model, the codeword is divided into $k$ states. The adversary tampers each state independently (and arbitrarily). In this work, we consider the case of $k = 3$.

**Informal Result 3.** *There is an explicit rate-*$1/3$ *non-malleable code against the three-split-state tampering family.*

As Cheraghchi and Guruswami [CG14a] show that the optimal achievable rate for $k$-split-state tampering family is $1 - 1/k$, our construction is a two-approximation of the optimal construction.

**Two-Lookahead Tampering.** In the $k$-lookahead tampering model, the codeword is also divided into $k$ states. However, the adversary tampers each state in a streaming fashion. That is, the adversary sees one block at a time, and the tampering on each block can only depend on previous blocks. We first extend the negative result of [CG14a] for split-state tampering to lookahead tampering. Next, for the case of $k = 2$, we give an explicit construction achieving the rate of $1/3$.

**Informal Result 4.** *The optimal achievable rate for the* $k$*-lookahead tampering is* $1 - 1/k$.

**Informal Result 5.** *There is an explicit rate-*$1/3$ *non-malleable code against the two-lookahead tampering family.*

### 1.1.2 Optimal-fair Coin-tossing

Fair multi-party computation mandates that honest parties shall always receive the output even if malicious parties abort the protocol prematurely. Unfortunately, this strong security guarantee is impossible to achieve with negligible simulation error. In an elegant work, Cleve [Cle86] shows that for the simplest task of coin-tossing, any $r$-message protocol[4] between two parties is inevitably $\Omega(1/r)$-unfair. That is, regardless of the computational hardness assumption that one assumes, there always exists an efficient attacker who may prematurely abort to deviate the expected output of the other party by $\Omega(1/r)$. On the

[4]↑In an $r$-message protocol between two parties, Alice and Bob exchange (at most) $r$ messages in any complete execution of the protocol.

other hand, the seminal work of Moran, Naor, and Segev [MNS09] prove a matching upper bound by presenting an explicit $r$-message protocol that achieves $\mathcal{O}(1/r)$-unfairness. The MNS protocol relies on the assumption that oblivious transfer exists. Therefore, $r$-message coin-tossing protocol achieving $\mathcal{O}(1/r)$-unfairness is called optimal-fair.

Given the MNS protocol, it is natural to ask if we could construct an optimal-fair coin-tossing protocol from weaker assumptions. For example, is it possible to build an optimal-fair coin-tossing protocol from one-way functions? In light of the current progress in complexity theory, an unconditional answer to this question is unlikely. To address this issue, among several possible approaches, a prominent technique is to study it via the lens of black-box separations, as introduced by Impagliazzo and Rudich [IR89].[5]

Our first result is a black-box separation between optimal-fair coin-tossing and one-way functions.

**Informal Result 6.** *Any $r$-message two-party coin-tossing protocol that uses one-way functions in a fully black-box manner is $\Omega(1/\sqrt{r})$-unfair.*

This black-box separation from one-way functions indicates that the two-party coin-tossing protocol of Blum [Blu82] and Cleve [Cle86], which uses one-way functions in a black-box manner and builds on the protocols of [ABC+85, BD84], achieves the best possible security for any $r$-message protocol. Their protocol is $\Omega(1/\sqrt{r})$-unfair, and any $r$-message protocol cannot have asymptotically better security by only using one-way functions in a black-box manner, thus resolving this fundamental question after over three decades.

Next, we extend our first result to any coin-tossing protocol that uses public-key encryption in a black-box manner in the information-theoretic $f$-hybrid model. Let $f\colon X \times Y \to \mathbb{R}^Z$ be a two-party secure symmetric function evaluation functionality, possibly with randomized output. The function takes private inputs $x$ and $y$ from the parties and samples an output $z \in Z$ according to the probability distribution $p_f(z|x,y)$. The *information-theoretic $f$-hybrid* is an information-theoretic model where parties have additional access to the (unfair)

[5]↑ Intuitively, a construction from one-way functions is black-box if it only relies on the input/output behavior of the one-way function and the security reduction only utilizes the adversary in a black-box manner as well.

$f$-functionality.[6] As an aside, we highlight that the fair $f$-hybrid (where the adversary cannot block output delivery to the honest parties), for any $f$ where both parties influence the output, straightforwardly yields *perfectly or statistically secure* fair coin-tossing protocol.[7]

Observe that if $f$ is the (symmetrized) oblivious transfer functionality,[8] then the Moran, Naor, and Segev protocol [MNS09] is an optimal fair coin-tossing protocol in the (unfair) $f$-hybrid. More generally, if $f$ is a functionality such that there is an oblivious transfer protocol in the $f$-hybrid, one can emulate the Moran, Naor, and Segev optimal coin-tossing protocol; consequently, optimal coin-tossing exists in the $f$-hybrid. Kilian [Kil00] characterized all functions $f$ such that there exists a secure oblivious transfer protocol in the $f$-hybrid, referred to as *complete* functions.

Our work explores whether a function $f$ that is *not complete* may enhance the security of fair coin-tossing protocols. In particular, we show that incomplete functionalities are useless for fair coin-tossing.

**Informal Result 7.** *Let $f$ be any incomplete functionality. Any $r$-message two-party coin-tossing protocol in the $f$-hybrid model that uses public-key encryption in a black-box manner is $\Omega(1/\sqrt{r})$-unfair.*

### 1.1.3 Collective Coin-tossing

Collective coin-tossing in the full information model is a fundamental primitive introduced by Ben-Or and Linial [BL85]. In this model, parties have unbounded computation power and access to a broadcast channel. A coin-tossing protocol among $n$ parties aims to upgrade each parties' local randomness into public randomness that they shall agree on. An adaptive Byzantine adversary, however, may corrupt some processors during the course of the protocol

[6]↑The functionality delivers the output to the adversary first. If the adversary wants, it can abort the protocol and block the output delivery to the honest parties. Otherwise, if the adversary wants, it can permit the delivery of the output to the honest parties and continue with the protocol execution.

[7]↑Suppose $f = \text{XOR}$. In a fair $f$-hybrid, the adversary cannot block the output delivery to the honest parties. So, parties input random bits to the $f$-functionality and agree on the output. This protocol has 0-insecurity.

[8]↑In the symmetrized oblivious transfer functionality, the sender has input $(x_0, x_1) \in \{0,1\}^2$, and the receiver has input $(b, r) \in \{0,1\}^2$. The symmetric oblivious transfer functionality returns $x_b \oplus r$ to both parties. If the receiver picks $r \leftarrow \{0,1\}$, then this functionality hides the receiver's choice bit $b$ from the sender.

and set their messages arbitrarily. The insecurity of the protocol is the maximum amount of deviation an adversary could cause to the expected output.

In this work, we study a simple setting called the single-turn protocol. That is, every party only speaks once during the evolution of the protocol. Moreover, we assume that the Byzantine adversary shall only corrupt (at most) one party. The asymptotic insecurity in this setting is well-understood. If there are $n$ parties, the protocol is $\Omega(1/\sqrt{n})$-insecure. The constant factor on the insecurity is, however, not well-understood. In this work, we give a tight analysis proving that the insecurity of the elegant threshold protocol[9] is at most two times the optimal insecurity in this setting.

**Informal Result 8.** *For a single-turn $n$-party collective coin-tossing protocol, the threshold protocol is a two-factor approximation of the optimal protocol against a Byzantine adversary who may corrupt (at most) one party.*

### 1.2 Organization

In CHAPTER 2, we survey the relevant literature on non-malleable codes and coin-tossing. In CHAPTER 3 and CHAPTER 4, we present our results on explicit constructions of non-malleable codes with high rate. These results are published as [GMW19, GMW18]. In CHAPTER 5, we present our studies on the minimal complexity of optimal-fair coin-tossing protocols. These results are published as [MW20, MW21]. In CHAPTER 6, we present our results on collective coin-tossing protocols. This result is published as [KMW21].

[9]↑A threshold protocol among $n$ parties with threshold $t$ is as follows. Every party broadcasts a uniformly random bit. The final output of the protocol is 1 if the total number of 1-messages exceeds the threshold $t$.

# 2. PRIOR WORKS

In this section, we survey relevant literature on non-malleable codes and coin-tossing.

## 2.1 Non-malleable codes

Non-malleable code is an elegant notion introduced by Dziembowski, Pietrzak, and Wichs [DPW10]. Let us first recall the definition of non-malleable codes and relative notions.

**Definition 2.1.1** (Coding Scheme)**.** *A pair of (possibly randomized) functions* $\mathsf{Enc}\colon \{0,1\}^{\ell} \to \{0,1\}^{n}$ *and* $\mathsf{Dec}\colon \{0,1\}^{n} \to \{0,1\}^{\ell} \cup \{\perp\}$ *is a coding scheme with block length* $n$ *and message length* $\ell$ *if it satisfies perfect (resp., statistical) correctness. That is, for all message* $m \in \{0,1\}^{\ell}$*, over the randomness of* $\mathsf{Enc}$ *and* $\mathsf{Dec}$*,* $\Pr[\mathsf{Dec}(\mathsf{Enc}(m)) = m] = 1$ *(resp.,* $\Pr[\mathsf{Dec}(\mathsf{Enc}(m)) = m] = 1 - \mathsf{negl}(\ell)$*). The rate of this encoding scheme is defined as* $\ell/n$*.*

**Definition 2.1.2** (Tampering family)**.** *Let* $\mathcal{F}_n$ *denote the set of all functions* $f\colon \{0,1\}^{n} \to \{0,1\}^{n}$*. A tampering family* $\mathcal{F}$ *is any subset of* $\mathcal{F}_n$*.*

**Definition 2.1.3** ($(n, \ell, \mathcal{F}, \varepsilon)$-Non-malleable Codes [DPW10])**.** *A coding scheme* $(\mathsf{Enc}, \mathsf{Dec})$ *with block length* $n$ *and message length* $\ell$ *is said to be non-malleable against a tampering family* $\mathcal{F}$ *with (simulation) error* $\varepsilon$*, if for all functions* $f \in \mathcal{F}$*, there exists a simulator* $\mathsf{Sim}(f)$*, which outputs a distribution over* $\{0,1\}^{\ell} \cup \{\perp, \mathsf{same}^*\}$*, such that for all messages* $m \in \{0,1\}^{\ell}$*,*

$$\mathsf{SD}\left(\mathsf{Tamper}_f^m \ , \ \mathsf{Copy}\left(\mathsf{Sim}(f), m\right)\right) \leqslant \varepsilon,$$

*where* tampering experiment $\mathsf{Tamper}_f^m$ *stands for*

$$\mathsf{Tamper}_f^m := \left\{ \begin{array}{c} c \leftarrow \mathsf{Enc}(m),\ \widetilde{c} = f(c),\ \widetilde{m} = \mathsf{Dec}(\widetilde{c}) \\ \\ \textit{Output: } \widetilde{m}. \end{array} \right\}$$

*and*

$$\mathsf{Copy}(x, y) = \begin{cases} y, & \textit{if } x = \mathsf{same}^*; \\ \\ x, & \textit{otherwise.} \end{cases}$$

**Impossible to achieve non-malleability for all tampering functions.** Note that it is impossible to construct non-malleable codes for all tampering functions, i.e., $\mathcal{F} = \mathcal{F}_n$. To see this, consider the function $f(c) := \mathsf{Enc}(\mathsf{Dec}(c) + 1)$. That is, $f$ decodes the codeword, increases the message by 1, and re-encodes the message again. Observe that $\mathsf{Tamper}_f^m$ always outputs $m + 1$. Consequently, $\mathsf{Sim}(f)$ cannot simulate $\mathsf{Tamper}_f^m$. Therefore, non-malleable code is usually defined with respect to a tampering family that is restricted in some way. As we shall see, the tampering families that have been considered in the literature can be roughly partitioned into two categories: split-state tampering family and computationally-bounded tampering family. We shall present a summary of these results in Section 2.1.3 and Section 2.1.4 respectively.

**Efficient Simulation.** We note that the definition non-malleable code only asks for the existence of a simulator $\mathsf{Sim}(f)$. In particular, it does not require this simulator to be computationally efficient. However, as observed by [CG14a], if a simulator $\mathsf{Sim}(f)$ exists, then the following simulator $\mathsf{Sim}^*(f)$ (Figure 2.1) is always an efficient simulator (given that $\mathsf{Enc}, \mathsf{Dec}, f$ are all efficiently computatable). We refer to this simulator as the *canonical simulator*. One could prove the following bound on the simulation error of the canonical simulator.

1. $m' \leftarrow \{0,1\}^\ell$.
2. $\widetilde{m} \leftarrow \mathsf{Dec}(f(\mathsf{Enc}(m')))$.
3. If $\widetilde{m} = m'$, output $\mathsf{same}^*$.
4. Otherwise, output $\widetilde{m}$.

**Figure 2.1.** The Canonical Simulator $\mathsf{Sim}^*(f)$.

**Theorem 2.1.1** (Canonical Simulator)**.** *Let* $(\mathsf{Enc}, \mathsf{Dec})$ *be an* $(n, \ell, \mathcal{F}, \varepsilon)$*-non-malleable code. Let* $\mathsf{Sim}(f)$ *be the corresponding simulator. That is, for all* $f \in \mathcal{F}$ *and* $m \in \{0,1\}^\ell$*, we have*

$$\mathsf{SD}\left(\mathsf{Tamper}_f^m\ ,\ \mathsf{Copy}\left(\mathsf{Sim}(f), m\right)\right) \leqslant \varepsilon.$$

*Then the simulation error of the canonical simulator* $\mathsf{Sim}^*(f)$ *satisfies*

$$\mathsf{SD}\left(\mathsf{Tamper}_f^m \ , \ \mathsf{Copy}\left(\mathsf{Sim}^*(f), m\right)\right) \leqslant 2^{-\ell+1} + 2\varepsilon.$$

*Proof.* One may prove this through a simple hybrid argument. Consider the following hybrid $H(f)$.

$H(f)$:

1. $m' \leftarrow \{0,1\}^\ell$.
2. $\widetilde{m} \leftarrow \mathsf{Copy}\left(\mathsf{Sim}(f), m'\right)$.
3. If $\widetilde{m} = m'$, output $\mathsf{same}^*$.
4. Otherwise, output $\widetilde{m}$.

The simulation error of the original simulator $\mathsf{Sim}(f)$ guarantees that

$$\mathsf{SD}\left(\mathsf{Sim}^*(f) \ , \ H(f)\right) \leqslant \varepsilon.$$

On the other hand, the only possibility that the output of $H(f)$ is different from the sample it draws from $\mathsf{Sim}(f)$ is when $\mathsf{Sim}(f)$ outputs $m'$. This happens with probability $2^{-\ell}$ (since $m'$ is uniformly at random) and thus

$$\mathsf{SD}\left(H(f) \ , \ \mathsf{Sim}(f)\right) \leqslant 2^{-\ell}.$$

This completes the proof as

$$\begin{aligned}
&\mathsf{SD}\left(\mathsf{Tamper}_f^m \ , \ \mathsf{Copy}(\mathsf{Sim}^*(f), m)\right) \\
\leqslant &\mathsf{SD}\left(\mathsf{Tamper}_f^m \ , \ \mathsf{Copy}(\mathsf{Sim}(f), m)\right) + \mathsf{SD}\left(\mathsf{Sim}(f) \ , \mathsf{Sim}^*(f)\right) \\
= &\left(\varepsilon + 2^{-\ell}\right) + \left(\varepsilon + 2^{-\ell}\right).
\end{aligned}$$

□

### 2.1.1 On the Existence of (High-rate) Non-malleable Codes

We have discussed that achieving non-malleability against all possible tampering functions is impossible. Moreover, for any coding scheme $(\mathsf{Enc}, \mathsf{Dec})$, there always exists some tampering function (e.g., $f(c) := \mathsf{Enc}(\mathsf{Dec}(c) + 1)$) such that $(\mathsf{Enc}, \mathsf{Dec})$ is not non-malleable against $f$. However, if we fix any small tampering family $\mathcal{F}$, there does exist some coding scheme $(\mathsf{Enc}, \mathsf{Dec})$, which is non-malleable against $\mathcal{F}$.

Observe that there are $2^{n2^n}$ different tampering functions, i.e., $|\mathcal{F}_n| = 2^{n2^n}$. In their original work, Dziembowski et al. [DPW10] proved that for any tampering family $\mathcal{F}$ such that $|\mathcal{F}| \leqslant 2^{2^{n-3\ell}}$, non-malleable code against $\mathcal{F}$ exists. In particular, they consider a random function $\mathsf{Dec}\colon \{0,1\}^n \to \{0,1\}^\ell$ as the decoding function and let $\mathsf{Enc}(m)$ returns a random preimage of $m$ under $\mathsf{Dec}$. They showed that the coding scheme drawn according to this distribution is non-malleable against $\mathcal{F}$ with overwhelming probability.

Later, Cheraghchi and Guruswami [CG14a] proved that for any tampering family $\mathcal{F}$ such that $|\mathcal{F}| \leqslant 2^{2^{an}}$ for some constant $a \in (0,1)$, non-malleable code for $\mathcal{F}$ with rate $(1-a)$ exists. In comparison to [DPW10], they considered a random coding scheme with high distance. In particular, this implies that most codewords are invalid and will be decoded to $\perp$. They also showed that the rate $1-a$ is the optimal achievable rate for any tampering family of size $2^{2^{an}}$.

The probabilistic constructions of [DPW10] and [CG14a] are inefficient. In another work, Faust et al. [FMVW14] (and further improved by Jafargholi and Wichs [JW15]) showed that for any tampering family $\mathcal{F}$ of size $2^{\mathsf{poly}(n)}$, *efficient* non-malleable code exists for $\mathcal{F}$. In particular, let $\mathcal{H}_1$ and $\mathcal{H}_2$ be two hash function families with appropriate independence guarantees. They considered a random encoding function $\mathsf{Enc}_{h_1,h_2}$ (where $h_1 \leftarrow \mathcal{H}_1$ and $h_2 \leftarrow \mathcal{H}_2$) as

$$\mathsf{Enc}_{h_1,h_2}(m) = \Big(r,\ h_1(r) \oplus m,\ h_2(r, h_1(r) \oplus m)\Big),$$

where $r$ is the private randomness of the encoding function $\mathsf{Enc}$. They prove that $\mathsf{Enc}_{h_1,h_2}$ is non-malleable with overwhelming probability. Their construction can be efficiently instantiated in the common randomness string (CRS) model, where the CRS determines the hash function $h_1$ and $h_2$.

### 2.1.2 Useful Tools for Explicit Construction of Non-malleable Codes

#### Non-malleable Extractor

Non-malleable extractors [DW09] extends the notions of traditional randomness extractors with non-malleability property. Informally, it guarantees that if the source gets tampered by some tampering function, the extractor output on the original source is uniform even conditioned on the extractor output on the tampered source. More formally, we define the seedless randomness extractors below.

**Definition 2.1.4** ((Seedless) Non-malleable Extractor [CG14b])**.** *Let $\mathcal{F}$ be a tampering family and $\mathcal{W}$ be a class of sources. A function* $\mathrm{nmExt}\colon \{0,1\}^n \to \{0,1\}^\ell$ *is said to be an $\varepsilon$-non-malleable extractor if the following holds. For any tampering function $f \in \mathcal{F}$, there exists a simulator $\mathsf{Sim}(f)$, which outputs a distribution over $\{0,1\}^\ell \cup \{\mathsf{same}^*\}$, such that for any source $W \in \mathcal{W}$,*

$$\Big(\mathrm{nmExt}(W), \mathrm{nmExt}\left(f(W)\right)\Big) \approx_\varepsilon \Big(U_\ell, \mathsf{Copy}(\mathsf{Sim}(f), U_\ell)\Big).$$

As observed by [CG14b], non-malleable extractor directly gives construction of non-malleable codes as one may treat nmExt as the decoding function $\mathsf{Dec}$ and $\mathrm{nmExt}^{-1}$ as the encoding function $\mathsf{Enc}$. In particular, they proved the following theorem.

**Theorem 2.1.2** ([CG14b])**.** *Let* $\mathrm{nmExt} : \{0,1\}^n \to \{0,1\}^\ell$ *be an $\varepsilon$-seedless non-malleable extractor for uniform source on $\{0,1\}^n$ and tampering family $\mathcal{F}$. Suppose* $\mathrm{nmExt}^{-1}$ *is efficiently computable. Then the coding scheme* $(\mathrm{nmExt}^{-1}, \mathrm{nmExt})$ *is an efficient non-malleable code against tampering family $\mathcal{F}$ with (simulation) error $2^\ell \cdot \varepsilon$.*

This surprising connection brings the exciting progress on randomness extractors to the field of non-malleable codes. In fact, many influential works in non-malleable codes come from this unique perspective [CZ14, CGL16, Li17, CL17, Li19].

### Non-malleable Reduction

Let us define $\mathsf{NM}_\ell$ as the set of constant functions, identity function, and the bot function[1] over $\{0,1\}^\ell$. Suppose $(\mathsf{Enc}, \mathsf{Dec})$ is an $(n, \ell, \mathcal{F}, \varepsilon)$-non-malleable codes. Intuitively, this implies that the effect on the message by applying any tampering $f \in \mathcal{F}$ on the codeword can be simulated by a convex combination of "tampering function" from $\mathsf{NM}_\ell$ on the message.

The notion of non-malleable reduction generalizes the above intuition to the setting where the "tampering function" on the message could come from an arbitrary family $\mathcal{G}$. Formally, we define it as follows.

**Definition 2.1.5** (Non-malleable Reduction [ADKO15])**.** *Let $\mathcal{F}$ be a tampering family on $\{0,1\}^n$. Let $\mathcal{G}$ be a tampering family on $\{0,1\}^\ell$. We say coding scheme $(\mathsf{Enc}, \mathsf{Dec})$ reduces $\mathcal{F}$ to $\mathcal{G}$ with simulation error $\varepsilon$, denoted as*

$$\mathcal{F} \leqslant_\varepsilon^{\mathsf{Enc},\mathsf{Dec}} \mathcal{G},$$

*if the following holds. For every tampering function $f \in \mathcal{F}$, there exists a simulator $\mathsf{Sim}(f)$, which outputs a distribution over $\mathcal{G}$, such that for $x \in \{0,1\}^n$,*

$$\mathsf{SD}\left(\mathsf{Dec}(f(\mathsf{Enc}(x)) \ , \ \mathsf{Sim}(f)(x)\right) \leqslant \varepsilon.$$

**Remark 2.1.1** (Non-malleable code as non-malleable reduction)**.** *The following two statements are equivalent (refer to Figure 2.2):*

- $(\mathsf{Enc}, \mathsf{Dec})$ *is an $(\mathcal{F}, \varepsilon)$-non-malleable code.*
- $\mathcal{F} \leqslant_\varepsilon^{\mathsf{Enc},\mathsf{Dec}} \mathsf{NM}_\ell$.

Non-malleable reduction is a particularly useful notion as it gives a modular way of constructing non-malleable codes (refer to the composition theorem below). Now, suppose we have that $\mathcal{F} \leqslant_\varepsilon^{\mathsf{Enc},\mathsf{Dec}} \mathcal{G}$. To construct a non-malleable code for tampering family $\mathcal{F}$, it suffices to construct a non-malleable code for tampering family $\mathcal{G}$. On the other hand, suppose we

[1]↑That is, the function $f$ that always output $\perp$ for all input $x \in \{0,1\}^\ell$.

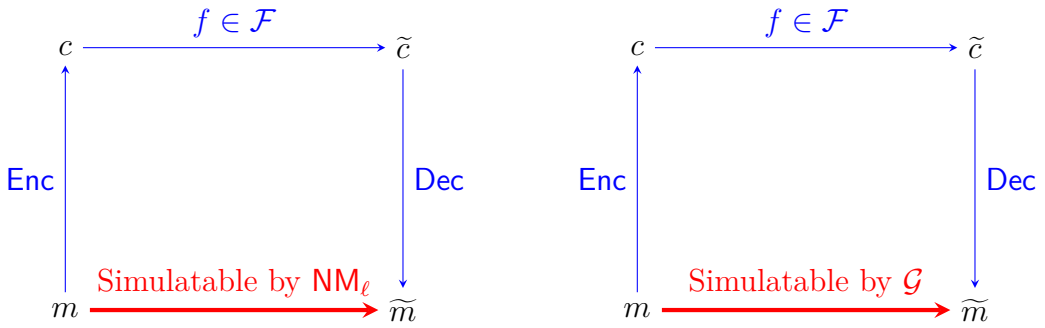



**Figure 2.2.** A side by side comparison between non-malleable code and non-malleable reduction. The blue route is simulatable by the red route using a distribution over some class of functions.

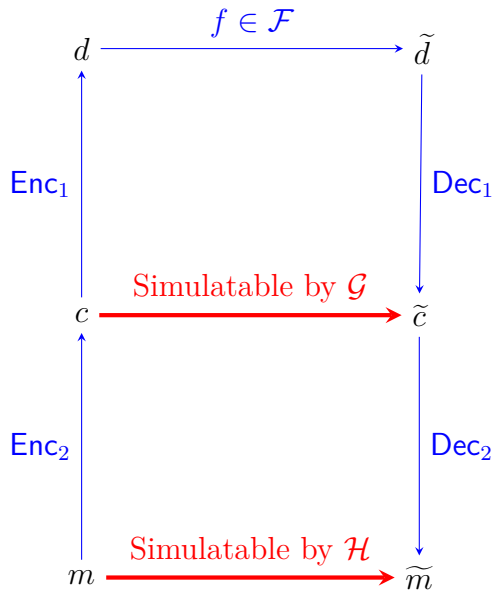



**Figure 2.3.** A pictorial summary of the composition of non-malleable reductions (Theorem 2.1.3).

have a non-malleable code for $\mathcal{G}$, it suffices to construct a coding scheme $(\mathsf{Enc}, \mathsf{Dec})$ that reduces $\mathcal{F}$ to $\mathcal{G}$. As it turns out, this powerful notion gives rise to many breakthrough results in explicit constructions of non-malleable codes [ADL14, ADKO15, BDKM16, CL17, BDG+18].

**Theorem 2.1.3** (Composition of non-malleable reduction [ADKO15], refer to Figure 2.3)**.** *If we have*

$$\mathcal{F} \leqslant_{\varepsilon_1}^{\mathsf{Enc}_1,\mathsf{Dec}_1} \mathcal{G} \qquad \textit{and} \qquad \mathcal{G} \leqslant_{\varepsilon_2}^{\mathsf{Enc}_2,\mathsf{Dec}_2} \mathcal{H},$$

*then*

$$\mathcal{F} \leqslant^{\mathsf{Enc},\mathsf{Dec}}_{\varepsilon_1+\varepsilon_2} \mathcal{H},$$

*where* $\mathsf{Enc}(m) := \mathsf{Enc}_1(\mathsf{Enc}_2(m))$ *and* $\mathsf{Dec}(c) := \mathsf{Dec}_2(\mathsf{Dec}_1(c))$.

### 2.1.3 Split-state Tampering

In $k$-split-state tampering, the codeword $c$ is split into $k$ states, i.e., $c = (c_1, \ldots, c_k)$. The adversary is allowed to pick a tuple of $k$ (arbitrary) functions $(f_1, \ldots, f_k)$ such that the tampered codeword shall be

$$\widetilde{c} = \Big(f_1(c_1), \ldots, f_k(c_k)\Big).$$

This tampering family is motivated by that one could store the codeword in different locations. Such physical separation forces the adversary to tamper with each state independently.

Note that the two-split-state tampering family is the strongest tampering family among $k$-split-state tampering families. Furthermore, the tampering functions in split-state tampering are not required to be efficient. Observe that the size of the $k$-split-state tampering family is $\geqslant 2^{2^{(1/k)\cdot n}}$. Hence, the result of [CG14a] implies that the optimal achievable rate for $k$-split-state tampering is $1 - 1/k$.

In the information-theoretic setting, the first construction of split-state non-malleable code is given by Dziembowski, Kazana, and Obremski [DKO13]. They constructed a non-malleable encoding for one-bit messages in the two-split-state setting. In more detail, they view each state as a vector in $\mathbb{F}^t$ for a suitable finite field $\mathbb{F}$. Whether the message is 0 or 1 is encoded by whether the left state and the right state are orthogonal to each other or not.

In subsequent work, Aggarwal, Dodis, and Lovett [ADL14] gave a construction of multi-bit non-malleable code in the two-split-state setting. Their work draws a beautiful connection from additive combinatorics. In particular, suppose $U$ and $U'$ are independent uniform distributions over $\mathbb{F}^t$. For any functions $f$ and $g$, the joint distribution

$$\langle U, U' \rangle \ , \ \langle f(U), g(U') \rangle$$

is close to a convex combination of affine distributions $\{(V\,,\,aV+b)\mid a,b\in\mathbb{F}\}$ where $V$ is uniformly distributed over $\mathbb{F}$. In other words, this result in additive combinatorics implies that the inner product encoding is a non-malleable reduction from the two-split-state tampering to affine tampering over $\mathbb{F}$ (Definition 2.1.5). Finally, they completed their construction by giving a construction of non-malleable code for the affine tampering family.

In another research direction initiated by Cheraghchi and Guruswami [CG14b], several works have been constructing (non-malleable) multi-source extractors.[2] Such extractor guarantees that if every source is tampered with independently, the extractor output of the original sources is uniform even given the extractor output of the tampered sources (Definition 2.1.4). As discussed earlier, such a non-malleable extractor directly implies non-malleable code given that the preimage could be efficiently sampled. In a breakthrough result, Chattopadhyay and Zuckerman [CZ14] constructed a non-malleable extractor for ten independent sources. Moreover, the corresponding non-malleable code their construction implies has an (inexplicit) constant rate. For the case of the two-source non-malleable extractor, there has been a sequence of influential works [CGL16, Li17, Li19]. The current state-of-the-art construction [Li19] gives a two-source extractor where the length of the extractor output $\ell=\Omega\Big(\frac{n\cdot\log\log\log n}{\log\log n}\Big)$. That is, the corresponding two-split-state non-malleable code has a rate of $\Omega\Big(\frac{\log\log\log n}{\log\log n}\Big)$. In another recent work, [CL20] considers the scenario where the way codeword split into each state is not fixed a priori but picked by the adversary as a part of the tampering function.

In a seminal work, Kanukurthi, Obbattu, and Sekar [KOS17] constructed a compiler that takes any two-split-state non-malleable with an inverse polynomial rate and compiles it into a four-split-state non-malleable with a rate of 1/3. This is the first work that gives a split-state non-malleable code with an explicit constant rate. Afterward, in independent and concurrent works, Kanukurthi et al. [KOS18] and Gupta et al. [GMW18] further compress the number of states from 4 to 3. Hence, they gave a three-split-state non-malleable code with

[2]↑ It is well-known that extracting even one bit for a single source is impossible. In particular, for any extractor $\mathrm{Ext}\colon\{0,1\}^n\to\{0,1\}$, there exists a source $W$ over $\{0,1\}^n$ with entropy $\geqslant n-1$ such that $\mathrm{Ext}(W)$ is totally biased. Therefore, for a single source, one typically requires a short seed $U\leftarrow\{0,1\}^d$ and guarantees that $\mathrm{Ext}(W,U)$ is uniform for all sources $W$ with sufficiently high entropy. Namely, this is a seeded extractor. On the other hand, if we have two independent sources $W_1$ and $W_2$, then extracting pure randomness without a seed is possible. Namely, constructing a (seedless) multi-source extractor is possible.

a rate of 1/3. Very recently, Aggarwal and Obremski [AO20] construct the first constant-rate non-malleable code in the two-split-state setting, which culminates this long line of work.

Recently, Rasmussen and Sahai [RS20] showed an interesting connection between two-split-state non-malleable code and expander graphs. Specifically, given a graph $G = (V, E)$, one may encode the one-bit message in whether a pair of vertices $(u, v)$ is an edge or a non-edge. That is, the message is 1 if $(u, v) \in E$ and 0 if $(u, v) \notin E$. They proved that this graph-based encoding (where the left state stores the vertex $u$ and the right state stores the vertex $v$) is non-malleable against two-split-state tampering if the graph $G$ satisfies appropriate expander properties. It is a fascinating open problem to extend their result to multi-bit messages.

Finally, if one restricts to split-state tampering functions that are efficiently computable, Aggarwal et al. [AAG+16] gave a rate-1 construction of two-split-state non-malleable codes based on any one-way functions. In their setting, the simulator and the tampering experiment is only required to be computationally indistinguishable.

### 2.1.4 Computationally-bounded Tampering Families

Another natural way of restricting the tampering family is to only consider tampering families with low computational complexity. This is also motivated as real-world tampering attacks usually can be modeled as some rather simple functions.

**Bit Tampering.** In the bit tampering model, the tampering function tampers each bit independently. Equivalently, one may think of it as a split-state tampering function where every bit is a single state. For this tampering family, [CG14b] gave a rate-1 construction. In another two works, Agrawal et al. [AGM+15a, AGM+15b] consider the bit tampering where the tampering function may additionally permute the bits after tampering each bit independently. They gave a rate-1 construction for this stronger tampering family as well.

**Local Tampering.** Ball et al. [BDKM16] considered the tampering functions where the tampering on each bit may only depend on a small number $\delta$ of bits. This bound $\delta$ is called the locality of tampering functions. They gave a construction for locality $\delta = o\left(\frac{n}{\log n}\right)$. Interestingly, their construction relies on a non-malleable reduction from local tampering

function to two-split-state tampering (Definition 2.1.5). In subsequent work, Gupta, Maji, and Wang [GMW19] built a compiler that compiles a non-malleable code for local tampering with an inverse polynomial rate into a rate-1 construction. Their compiler, however, works only for locality $\delta = \mathcal{O}(\log n)$. As a corollary, they gave the first rate-1 non-malleable code for $\mathsf{NC}^0$ tampering.

**Small-depth Circuits.** Another natural tampering family is circuits with bounded depth and unbounded fan-in (e.g., $\mathsf{AC}^0$). Chattopadhyay and Li [CL17] gave the first construction for $\mathsf{AC}^0$ tampering. However, their codeword has a super-polynomial length and, hence, is inefficient. En route to their final result, they also constructed an (efficient) non-malleable code for affine (over the field $\mathbb{F}_2$) tampering function. In subsequent work, Ball et al. [BDG+18] gave an efficient non-malleable code for $\mathsf{AC}^0$ tampering. They gave their construction through a sequence of non-malleable reductions from depth-$d$ circuits tampering to depth-$(d-1)$ circuits tampering.

**Small-depth Decision Tree.** In another recent work, Ball, Guo, and Wichs [BGW19] consider all the tampering functions that can be computed by a decision tree of depth $\mathcal{O}\left(n^{1/4}\right)$. Following the paradigm of [BDKM16, BDG+18], they gave a non-malleable reduction from a decision tree to split-state tampering.

**Low degree polynomials.** Finally, the recent work by Ball et al. [BCL+20] considered all tampering functions that can be expressed as a small-degree polynomial over an appropriate finite field $\mathbb{F}$ and gave a construction for this tampering family. However, their construction only works for an exponentially large field. In particular, the size of the field $|\mathbb{F}|$ has to be large than the total number of messages $2^\ell$. Therefore, constructing non-malleable code for polynomial tampering over a small field $\mathbb{F}$ (e.g., $\mathbb{F}_2$) remains open.

**Construction that relies on computational assumptions.** If one is willing to assume computational assumptions, Ball et al. [BDK+19] (which is built upon [BDKM18]) showed how to construct non-malleable codes for any bounded polynomial-time tampering family. Their result relies on both worst-case hardness assumptions and average-case hardness assumptions. Additionally, their scheme only guarantees inverse polynomial (computational) indistinguishability. Recently, Ball et al. [BDKM20] presented some barriers in constructing

non-malleable code based on hardness assumption. They showed that for certain tampering classes, constructing non-malleable code based on hardness assumption when the security proof only uses the tampering function in a black-box manner is impossible.

**Spaces bounded.** There also have been works that consider the tampering function with a limited amount of space. The works of [FHMV17, CCHM19] constructed non-malleable codes for such tampering functions by relying on problems (e.g., *proof-of-space*) that are hard for space-bounded adversaries.

### 2.1.5 Variants of Non-malleable codes

In the literature, some variants of non-malleable codes have been introduced and studied. We discuss some variants below.

**Continuous non-malleable codes.** Continuous non-malleable codes, introduced by Faust et al. [FMNV14], consider the setting where the adversaries may continuously tamper the codeword. Furthermore, every new tampering function the adversary picks may depend on the decoding of the previously tampered codeword. A number of works [JW15, CGL16, AKO17, OPVV18, ADN+19, DK19] have studied and constructed continuous non-malleable codes in the split-state model that achieve diverse security guarantees.

**Local non-malleable codes.** A coding scheme is said to be locally decodable if one may decode every bit of the message by only querying a small number of bits from the codeword. Similarly, a locally updatable code ensures that to update a codeword of some message $m$ to be a codeword of another related message $m'$, one only needs to update a small number of bits from the codeword. Recently, a sequence of works [DLSZ15, CKR16, DKS17, DKS18] study non-malleable codes that are additionally locally decodable and updatable.

**Interactive non-malleable codes.** Inspired by *interactive coding* [Sch96] as a generalization of error-correcting codes for interactive protocols, interactive non-malleable codes [FGJ+19] aims to extend non-malleable coding to interactive protocols as well. Similarly, it ensures that the tampered interaction either encodes the correct protocol or encodes an unrelated execution of the protocol.

### 2.1.6 Applications of Non-malleable codes

Since the introduction of non-malleable code, it has found applications in various scenarios. We list some prominent applications below.

**Non-malleable Commitments.** Non-malleable commitment scheme [DDN91] is a commitment protocol, which guarantees that a man-in-the-middle (MIM) attacker cannot maul the commitments it receives from Alice to commit to a related secret to Bob. In such communication protocols, a natural restriction arises for *synchronous* adversaries.[3] That is, the tampering on each message in the protocol could not depend on future messages. Such tampering family is called blockwise tampering [CGM+16] (and also lookahead tampering [ADKO15]). Non-malleable codes for this tampering family turn out very useful for constructing non-malleable commitment schemes [GPR16, CGM+16].

**Non-malleable Secret-Sharing.** Non-malleable secret-sharing, introduced by [GK18a], is a useful primitive in secure multi-party computation. Similar to non-malleable codes, it requires that when the secret shares are tampered with, the recovered secret is unrelated to the original secret. Currently, most constructions [GK18a, GK18b, BS19] of non-malleable secret-sharing are based on non-malleable codes in the split-state model.

**Privacy Amplification.** Suppose two parties have a shared secret that is only guaranteed to have high entropy but not uniformly random. Privacy amplification asks for whether they agree on a new secret that is guaranteed to be uniformly random through communication over a (fully tamperable) public channel. In the seminal work, Dodis and Wichs [DW09] introduced (seeded) non-malleable extractor for constructing privacy amplification protocols. In the work of [CKOS19], Chattopadhyay et al. shows that non-malleable codes could also be leveraged to construct privacy amplification protocols.

[3]↑ Synchronous MIM attacker must send the first message to Bob before it receives the second message from Alice and so on.

## 2.2 Coin-tossing Protocols

### 2.2.1 Fair Coin-tossing

In secure multi-party computation, guaranteed output delivery is a desired yet very challenging objective. Intuitively, we would like the guarantee that either no party receives the output of the protocol, or all parties receive the output of the protocol. In more detail, secure multi-party computation with guaranteed output delivery is usually defined in the following real-world v.s. ideal-world paradigm. In the ideal world, a trusted party takes the inputs from all parties and honestly computes and delivers the output to all parties.[4] For any adversarial behavior in the real world, there must exist a simulator that simulates the same adversarial behavior in the ideal world.

Take the coin-tossing functionality as an example. Here, parties wish to toss an unbiased coin together. Since this functionality takes no input, the trusted party shall simply toss an unbiased coin and send the outcome to all parties in the ideal world. Therefore, our real-world protocol has to guarantee that honest parties shall always agree on an output whose expectation is $1/2$. Such coin-tossing is called *fair coin-tossing protocol*. The *unfairness* of a fair coin-tossing is the amount of change (in the statistical distance) an adversary can cause to the distribution of the output of honest parties.

In a seminal result, Cleve [Cle86] proved that fair coin-tossing with negligible unfairness is impossible. In particular, Cleve showed that any $r$-message[5] two-party coin-tossing protocol is at least $\Omega(1/r)$-unfair (regardless of the computational assumptions one assumes). This implies that any $r$-message multi-party coin-tossing protocol, where honest parties are not in the majority, is at least $\Omega(1/r)$-unfair. Hence, an $r$-message protocol with $\mathcal{O}(1/r)$-unfairness is called *optimal-fair*.

Before we move ahead, we stress that the negative results of fair coin-tossing imply negative results for other functionalities. For instance, consider the two-party (deterministic) XOR functionality. This functionality takes one input bit from Alice and Bob each and

[4]↑If we do not ask for guaranteed output delivery, in the ideal world, the adversary receives the output from the trusted party first. She may then decide to block the output delivery to the other honest parties.
[5]↑In an $r$-message protocol, the total number of messages sent by all parties for any full execution of the protocol is (at most) $r$.

computes the XOR of the input bits. Suppose we have a protocol that fairly realizes the XOR functionality. Then one can realize the coin-tossing functionality fairly by using this protocol as a subroutine and instructing both Alice and Bob to sample a uniform bit as their respective inputs. Therefore, intuitively, Cleve's negative result implies that any $r$-message XOR protocol must be "$\mathcal{O}(1/r)$-insecure".[6] Interestingly, not all hope is lost in fair computation. There are interesting functionalities that can be realized with perfect fairness [GHKL08, ALR13, Ash14, Mak14, ABMO15].

In the two-party setting, Moran, Naor, and Segev [MNS09] presented an elegant protocol that is optimal-fair. Their protocol relies on the existence of oblivious transfer. Intuitively, the MNS protocol has a critical round $i^*$, where the output of the protocol goes from uniformly random to completely fixed. The only effective attack by the adversary is to abort at precisely the round $i^*$ , for which he could only succeed with probability $1/r$.

Building the MNS protocol, Beimel, Omri, and Orlov [BOO10] presented an optimal-fair multi-party coin-tossing protocol. Their protocol assumes that at least $1/3$ fraction of the parties are honest. In particular, their protocol does not give an optimal-fair three-party coin-tossing protocol where two parties may be malicious.

Haitner and Tsfadia [HT14] partially solved the three-party setting. They gave a three-party $r$-message coin-tossing protocol that is $\frac{\log^3 r}{r}$-unfair. Using similar ideas as in [BOO10], Alon and Omri [AO16] compiled the protocol of Haitner and Tsfadia [HT14] into a multi-party coin-tossing protocol. The protocol of Alon and Omri [AO16] assumes that at least $1/4$ fraction of the parties are honest and achieve $\frac{\log^3 r}{r}$-unfairness. Recently, Buchbinder et al. [BHLT17] presented a new three-party protocol that achieves $\frac{\sqrt{\log r}}{r}$-unfairness. They also gave a $n$-party coin-tossing protocol where $n-1$ parties might be malicious for any $n \leqslant \log\log r$. In general, for any number of parties $n \geqslant 4$, there is still a significant gap between the lower bound on the optimal fairness [Cle86] and the fairness of the state-of-the-art construction [BHLT17].

Recently, Beimel et al. [BHMO18] investigated the case where the number of parties $n = \mathsf{poly}(r)$. They proved that for this range of parameters, any $n$-party $r$-message coin-tossing protocol is $\Omega(1/\sqrt{r})$-unfair.

[6]↑ To be more precise, the statistical distance between the real world and ideal world must be $\mathcal{O}(1/r)$.

In the two-party setting, although we know that oblivious transfer enables optimal fairness. It has been a long open problem whether the existence of one-way functions is sufficient for optimal fairness. First, we know that one-way functions imply commitment schemes [Nao91, NOVY98, HR07]. And by using commitment schemes to ensure honest behavior, the majority protocol [Blu82, BD84, ABC+85, Cle86] achieves $\mathcal{O}(1/\sqrt{r})$-unfairness.[7] Cleve and Impaliazzo [CI93] prove that if one only considers fail-stop adversaries,[8] then any coin-tossing protocol in the information-theoretic setting is $\Omega(1/\sqrt{r})$-unfair.

In two works [DLMM11, DMM14], Dachman-Soled et al. showed that optimal-fair coin-tossing is black-box separated from one-way functions under certain restrictions. Recently, Maji and Wang [MW20] resolved this problem in the full generality. They proved that optimal-fair coin-tossing is black-box separated from one-way functions without any restrictions. In a follow-up work, they [MW21] proved that optimal-fair coin-tossing is also black-box separated from public-key encryption and idealized functionalities that do not imply OT.

In another recent work, Haitner, Makriyannis, and Omri [HMO18] proved that the existence of an $r$-message protocol with $\mathcal{O}(1/r)$-unfairness implies (infinitely often) key-agreement protocol. However, their results only work for constant $r$.

### 2.2.2 Collective coin-tossing in the full information model

The full information model is introduced by Ben-Or and Linial [BL89]. In this model, $n$ computationally unbounded parties engage in a communication protocol over a single broadcast channel. In every round, some parties shall broadcast their messages. We assume the adversaries are *rushing*. That is, suppose the $i^{th}$ party is corrupted, and she is going to send her message this round. She will wait for the other parties to send their messages first, before she decides on and sends her message.

We usually consider *Byzantine* adversaries. That is, once a party is corrupted, the adversary takes full control over that party and may send arbitrary messages on her behalf.

[7]↑Intuitively, in the majority protocol, parties sample $r$ random bits, and the final output is the majority of the $r$ random bits sampled.
[8]↑Fail-stop adversary is semi-honest except that it might prematurely abort.

However, we will also mention some results in other models. We say an adversary is *static* if the parties that she corrupts are chosen before the protocol starts. In contrast, an *adaptive* adversary might decide to corrupt some parties during the execution of the protocol. The choices of the parties to corrupt could potentially depend on the partial transcript so far.

In what follows, we survey the results of coin-tossing protocols[9] in the full information model against various type of adversaries. Since parties are computationally unbounded, without loss of generality, we may assume parties are *stateless*. That is, the distribution of the next message a party sends is completely determined by the partial transcript. Moreover, we may assume every message in the protocol is uniformly at random of a certain length. Finally, we may assume that the final output $\in \{0,1\}$ of the protocol is a deterministic function of the full transcript of the protocol. The *insecurity* of a coin-tossing protocol is defined as the maximum changes (in the statistical distance) an adversary could cause to the distribution of the output of the protocol.

**One-round Static Corruption.** In a one-round protocol, every party sends one message. Suppose the $i^{th}$ party's message is uniformly drawn from an alphabet $\Sigma_i$ and the final output of the protocol is determined by the function $f\colon \Sigma_1 \times \cdots \times \Sigma_n \to \{0,1\}$.

In the setting where $\Sigma_1 = \Sigma_2 = \cdots = \Sigma_n = \{0,1\}$, this problem is closely related to the notion of *influence* in boolean function analysis. In particular, for a boolean function $f\colon \{0,1\}^n \to \{0,1\}$, the influence of the $i^{th}$ coordinate is defined as

$$\mathsf{Inf}_i[f] := \Pr_{x \leftarrow \{0,1\}^n} \left[ f(x) \neq f(x^{\oplus i}) \right],$$

where $x^{\oplus i} \neq x$ is the unique string that agrees with $x$ at every coordinate except for $i$. Naturally, one extends this definition to any collection of indices $S \subseteq \{1, 2, \ldots, n\}$. Observe that for a rushing adversary who statically corrupts all the parties from $S$, the insecurity of

[9]↑For the ease of presentation, we assume the expectation of the output is always 1/2. That is, it is an unbiased coin-tossing.

the protocol is exactly the influence of $S$, i.e., $\mathsf{Inf}_S[f]$. In a seminal result, Kahn, Kalai, and Linial [KKL88] proved that for any boolean function $f$, it holds that

$$\max_i \left(\mathsf{Inf}_i[f]\right) = \Omega\left(\frac{\log n}{n}\right).$$

Therefore, a static adversary could always corrupt one party and deviate the expected output by $\Omega\left(\frac{\log n}{n}\right)$. By iterative applications of the KKL theorem, one could prove that there exists a set $S$ of size $\mathcal{O}\left(\frac{n}{\log n}\right)$ such that $\mathsf{Inf}_S[f] = 1 - \mathrm{o}(1)$. That is, a static adversary could always corrupt $\mathcal{O}\left(\frac{n}{\log n}\right)$ parties and make the expected output to be either $\mathrm{o}(1)$ or $1 - \mathrm{o}(1)$.

On the other hand, finding explicit boolean function $f$ (which, in turn, gives explicit collective coin-tossing protocol) that matches the lower bound on the influences turns out to be hard. If one wants to find a boolean function $f$ such that the influence of every single coordinate is at most $\mathcal{O}\left(\frac{\log n}{n}\right)$, it is well-known that the tribe function[10] achieves this bound. However, to find a function $f$ such that a large coalition could not nearly fix the output of $f$ is extremely challenging. Such functions are sometimes refered to as *resilient functions*. Ajtai and Linial [AL93] gave a probabilistic construction of a function $f$ such that any coalition of $\Omega\left(\frac{n}{\log^2 n}\right)$ cannot fix the output of $f$. Only recently, we have found explicit constructions [CZ16, Mek17] that achieves this $\Omega\left(\frac{n}{\log^2 n}\right)$ probabilistic bound.

If one considers arbitrary alphabets $\Sigma_1, \Sigma_2, \ldots, \Sigma_n$, things become much more complicated. First, Bourgain et al. [BKK+92] proved that it still hold that there exists one coordinate with influence $\Omega\left(\frac{\log n}{n}\right)$. That is, one could still corrupt one party to deviated the expected output by $\Omega\left(\frac{\log n}{n}\right)$. However, one could not iteratively apply this general theorem to prove that corrupting $\mathcal{O}\left(\frac{n}{\log n}\right)$ parties could almost fix the output of the protocol to be either 0 or 1.[11] It is conjectured by Friedgut [Fri04] that there always exists a bit $b \in \{0, 1\}$ such that corrupting $\mathcal{O}\left(\frac{n}{\log n}\right)$ parties could almost fix the output of the protocol to be $b$. In recent work, Filmus et al. [FHH+19] made partial progress by proving that Friedgut's con-

[10]↑A tribe function over $x_1, x_2, \ldots, x_n$ is defined as $(x_1 \wedge \cdots \wedge x_w) \vee (x_{w+1} \wedge \cdots \wedge x_{2w}) \vee \cdots \vee (x_{n-w+1} \wedge \cdots \wedge x_w)$, where $w \approx \log n - \log\log n$.

[11]↑In fact, Figure 6.1 and Figure 6.2 present two counterexamples. For Figure 6.1, to significantly increase the probability of the protocol outputing 1 requires corrupting $\Omega(n)$ parties. Analogously, for Figure 6.2, to significantly increase the probability of the protocol outputing 0 requires corrupting $\Omega(n)$ parties.

jecture holds for arbitrary product measures over boolean hypercube $\{0,1\}^n$.[12] The general case of the Friedgut's conjecture remains open.

**Multiple-round Static Corruption.** If one considers multiple rounds protocol, this problem is closely related to the *leader election* problem. In a leader election protocol, the objective is to select a leader among $n$ parties such that honest parties shall be elected with high probability. Observe that collective coin-tossing reduces to leader election as one may first perform a leader election protocol and then ask the elected leader to output an unbiased bit on behalf of everyone. If the elected leader is honest, the final output will be unbiased. However, if the elected leader is malicious, all securities are lost.

Observe that one cannot hope to find a leader election protocol that is resilient to $n/2$ dishonest parties. To see this, partition all the parties equally into two partitions. One may think of the leader election protocol as a two-party game where a partition of parties wins if the elected leader comes from this partition. Clearly, there is a winning strategy for one partition and, hence, by corrupting all the parties in this partition, the adversary is guaranteed to be elected as the leader.

In elegant work, Saks [Sak89] proposed the simple "*baton passing*" protocol. In a baton passing protocol, one party starts with a baton. In every round, the party that currently holds the baton shall pass the baton to a random new party that has not held the baton yet. The final party who holds the baton is the elected leader. By the tight analysis of [AL93], we know that baton passing protocol is resilient to $\Theta\left(\frac{n}{\log n}\right)$ dishonest parties.

After Saks' work, a sequence of highly influence works [AN90, BN93, RZ98, Fei99, RSZ99] have been trying to improve the resilience of leader election protocols to the maximal achievable $(1/2-\varepsilon)n$ parties, where $\varepsilon > 0$ is an arbitrary constant. In particular, Feige [Fei99] proposed this elegant "*lightest bin*" protocol, which proceeds as follows. For a round among $n$ parties, imagine there are some $m = n/\mathsf{polylog}\,(n)$ bins. Parties are instructed to selected a random bin. After all parties send the bin they select, those parties that come from the bin with the smallest number of parties (i.e., the lightest bin) will proceed to the next round. Parties coming from the remaining bins are eliminated. Feige [Fei99] proved that the

[12]↑Arbitrary product measure over $\{0,1\}^n$ could always be (approximately) simulated by a uniformly measure over $\Sigma^n$ for an appropriate finite alphabet $\Sigma$. But the converse is not true.

lightest bin protocol is resilient to $(1/2 - \varepsilon)n$ corruptions. Moreover, the round complexity of the lightest bin protocol is $\log^* n$,[13] which is optimal as proven by Russell, Saks, and Zuckerman [RSZ99].

**One-round Strong Adaptive Corruption.** Strong adaptive adversary [GKP15] is an adaptive adversary who gets to see the messages parties are about to send first and then may decide to corrupt some parties and change their messages. The problem of a one-round protocol against strong adaptive adversaries is closely related to the problem of *isoperimetric inequalities*.

For a one-round protocol where $f\colon \Sigma_1 \times \cdots \times \Sigma_n \to \{0,1\}$ determines the output. Let $S := f^{-1}(1)$ and $\overline{S} := f^{-1}(0)$. Consider the *vertex boundary* $\partial S$ of $S$. That is,

$$\partial S := \left\{ y \in \overline{S} \;\middle|\; \exists x \in S \text{ s.t. } \mathsf{HD}(x,y) = 1 \right\}.^{14}$$

Clearly, if the messages parties are about to send come from $\partial S$, then a strong adaptive adversary could corrupt one party to alter the output of the protocol from 0 to 1. Analogously, if the messages parties are about to send come from $\partial\overline{S}$, then the adversary may alter the output from 1 to 0. Therefore, the insecurity of the protocol is determined by the size of the boundaries $\partial S \bigcup \partial\overline{S}$. In other words, to find the optimal protocol, one needs to find the set $S$ (with density $1/2$) that has the smallest boundary. If the alphabets $\Sigma_i = \{0,1\}$ for all $i$, then Harper's theorem [Har66] states that Hamming ball is exactly the set that minimizes its vertex boundary. However, for general alphabets, the characterization of the set with minimal boundary is unknown [Har99].

Finally, if one considers how many corruption are needed to nearly fix the output of the protocol, then by Azuma's inequality [Azu67], $\mathcal{O}(\sqrt{n})$ corruptions suffice. This result holds for arbitrary alphabets.

**Adaptive Corruption.** In the adaptive corruption setting, Ben-Or and Linial [BL85] conjectured that, for any collective coin-tossing protocol, an adversary could corrupt $\widetilde{\Theta}(\sqrt{n})$

[13] $\log^* n$ is defined to be the smallest $k$ such that $\log^{(k)}(n) < 1$. Here, $\log^{(1)}(n) := \log n$ and $\log^{(k)}(n) := \log\left(\log^{(k-1)}(n)\right)$ for $k > 1$.

[14] $\mathsf{HD}(x,y)$ denotes the Hamming distance between $x$ and $y$.

processors and nearly fix the output of the protocol. A multiple-round protocol is said to be *single-turn* if every party speaks only once. Lichtenstein, Linial, and Saks [LLS89] proved that for a single-turn protocol where every party only sends a bit, majority protocol is the optimal protocol. Since one can fix $\sqrt{n}\log^2 n$ bits to nearly fix the output of the majority protocol, Ben-Or and Linial's conjecture is confirmed for this setting. Goldwasser, Kalai, and Park [GKP15] considered a one-round protocol where every party might send a long message. Additionally, they only consider protocols that are symmetric. That is, permuting the order of the messages does not change the output of the protocol. They proved that Ben-Or and Linial's conjecture is correct in this setting. Kalai, Komargodski, and Raz [KKR18] proved that for any single-turn protocol, corrupting $\widetilde{\Theta}(\sqrt{n})$ parties suffice to fix the output of the protocol, thus proving Ben-Or and Linial's conjecture for any single-turn protocol. Very recently, Haitner and Karidi-Heller [HKH20] finally fully resolved this conjecture in the positive. That is, for any $n$-party coin-tossing protocol in the full information model, an adaptive adversary could corrupt $\widetilde{\Theta}(\sqrt{n})$ parties and nearly fix the output of the protocol.

**Other types of attacks.** In the literature, other types of attacks have also been considered. For single-turn protocols, consider an adaptive adversary who may restart one party after seeing the message it is about to send. Cleve and Impagliazzo [CI93] proved that such adversary could deviate the expected output of the protocol by $\Omega(1/\sqrt{n})$. Khorasgani, Maji, and Mukherjee [KMM19] used a geometric approach to give a tight upper and lower bound on the optimal achievable insecurity in this setting. Their approach is naturally constructive and produces the optimal protocol. However, it is not clear if one can efficiently implement their protocols. Recently, Khorasgani et al. [KMMW21] presented an efficient protocol that approximates the optimal protocol by [KMM19].

For single-turn protocols, Aspnes [Asp97, Asp98] considered the setting where an adaptive adversary may replace a party's message by a default one after seeing the message it is about to send. He also used a similar geometric approach to prove that an adversary could corrupt $\widetilde{\Theta}(\sqrt{n})$ parties to nearly fix the output of the protocol.

# 3. EXPLICIT RATE-1 NON-MALLEABLE CODE FOR LOCAL TAMPERING

Dziembowski, Pietrzak, and Wichs [DPW10] introduced the notion of non-malleable codes as an extension of the standard objective of error-correction. Non-malleable codes provide message-integrity assurances even when error-detection, let alone error-correction, is impossible. Suppose a sender encodes a message $m \in \{0,1\}^\ell$ and transmits the codeword over a channel. If the channel adds an error that has a small Hamming weight, then the sender can encode the message using an error-correcting code and the receiver can error-correct and retrieve the original message. Algebraic Manipulation Detection codes [CDF+08] help the receiver detect if the transmitted codeword is tampered using algebraic operations. For more sophisticated classes of tampering function $\mathcal{F}$, demanding manipulation detection or error-correction might be far-fetched. For example, suppose the channel replaces the original codeword with a fixed valid codeword. In this case, error-correction or error-detection is impossible. Non-malleable codes provide a meaningful message integrity assurance against sophisticated tampering families.

Let us fix an encoding and a decoding scheme $(\mathsf{Enc}, \mathsf{Dec})$, and a tampering function family $\mathcal{F}$. Non-malleable codes ensure that for any message $m \in \{0,1\}^\ell$ and tampering function $f \in \mathcal{F}$, the tampered message $\mathsf{Dec}(f(\mathsf{Enc}(m)))$ is either identical to the original message $m$ or a simulator $\mathsf{Sim}(f)$ can simulate this distribution (that is, it is independent of the original message). Even such a weak message integrity assurance turns out to be cryptographically useful, for example, in storing secret-keys for cryptographic primitives [DPW10, KOS18] and non-malleable messaging [GK18a, GK18b]. Naturally, we measure the quality of non-malleable codes using the following two parameters.

1. **Rate.** The ratio of the length of the message to the length of its encoding.

2. **Sophistication of the tampering family.** The complexity of the tampering attacks captured by the tampering functions in this family.

Constructing explicit non-malleable codes with high rate against sophisticated tampering function families is the guiding principle for the research in non-malleable codes. However,

both these objectives, even independently, have been significantly non-trivial to achieve. Only recently, using elegant probabilistic arguments, [FMVW14, CG14a] constructed rate-1 non-malleable codes in the *CRS model* for tampering families of bounded size.[1]

In this chapter, for any positive constant $\xi < 1$, we present the first *rate-1* explicit non-malleable codes against any tampering function that has $\xi \lg n$ *output locality*, i.e., at most $\xi \lg n$ input-bits influence any output bit of the tampering function. Note that there is no bound on the input locality, i.e., the number of output positions one input bit can influence during tampering. Here lg represents the logarithm with base 2, and $n$ represents the length of the codeword. Notably, our construction is in the information-theoretic plain model. We emphasize that our construction does not rely on any computational hardness assumption or a CRS.

## 3.1 Our Contribution

Our work focuses on constructing non-malleable codes, in the information-theoretic plain model, against tampering functions that are $\delta$-local, i.e., at most $\delta$ input bits influence any output bit. We emphasize that $\delta$ can be a function of $n$, the size of the codeword. Our work, for any positive constant $\xi < 1$, constructs explicit rate-1 NMC against $\delta$-local tampering functions, where $\delta = \xi \lg n$, which is a tampering family of size $2^{n^{1+o(1)}}$. In our case, the locality $\delta = \omega(1)$ and, hence, the set of all $\delta$-local tampering functions subsumes the family of $\mathsf{NC}^0$ tampering functions.

We present a general black-box compiler that takes three ingredients as input and constructs a non-malleable code for local functions. At an intuitive level, we prove the following result.

**Informal Theorem 3.1.1.** *For any positive constant $\xi < 1$, there exists an explicit and efficient rate-1 NMC against $\xi \lg n$-local tampering functions using the following primitives in a black-box manner (refer to Figure 3.2).*

[1] ↑Tampering functions can access the CRS; however, they cannot tamper the CRS.

1. *Rate-1 linear error-correcting code*[2] *with (near) linear distance and dual-distance (see Definition 3.4.2),*

2. *Rate-*$1/\eta^{o(1)}$ *NMC against leaky input and output local tampering for message length* $\eta$ *(referred to as the base NMC) (see Definition 3.4.1), and*

3. *A pseudorandom generator for finite state machines with super-polynomial stretch (see Definition 3.4.4).*

The compiler (refer to Figure 3.1 for an outline) encodes the message $m$ using the error-correcting code. Then, it samples a few entries of the codeword (at a suitable rate) and adds errors at half of them. The compiler tabulates all the sampled entries (both the erroneous and unaltered ones) along with their respective locations. The erroneous codeword forms the primary payload of the message $m$. The list of tabulated entries is appropriately encoded using a combination of the base NMC and the PRG and is juxtaposed (at the end) for consistency checks during decoding. If the rate of subsampling is sufficiently low, then the overall construction is rate-1. The security argument proceeds by demonstrating that if the subsampling rate is sufficiently high, then any local function cannot change the payload without being inconsistent with the tabulated entries themselves. Section 3.2 provides an intuitive overview of our compiler's construction.

Finally, we instantiate the respective primitives using (1) Reed-Solomon Codes over characteristic 2 fields, (2) An appropriate encoding introduced by Ball et al. [BDKM16], and (3) Nisan's PRG [Nis90]. As a consequence, we construct explicit efficient rate-1 NMC against $\xi \lg n$-local tampering functions, for any positive constant $\xi < 1$, with negligible simulation error (refer Theorem 3.5.2).

**Remark.** We note that the resulting decoding function for our construction is randomized. However, the randomization stems solely from the randomized decoding function of the base NMC construction of [BDKM16]. Given an appropriate NMC against leaky input and output local tampering with deterministic decoding, our construction will have deterministic

[2]↑Error-correcting codes can be converted into error-correcting secret sharing schemes using standard share-packing techniques [Sha79, BM84, FY92].

decoding.

**Remark.** If the base NMC is only rate-1/$\mathsf{poly}(n)$, then our compiler with suitably modified parameters, as indicated in Appendix A.3, constructs an explicit rate-1 NMC against $o(\log n)$-local tampering functions.

### 3.2 Technical Overview

As a starting point, it is instructive to understand the construction of Agrawal et al. [AGM+15b] for a rate-1 NMC against tampering functions with input and output locality 1. The conceptual hurdles in generalizing this approach to $\delta$-local functions, we believe, motivates the components used in our construction.

**Construction of Agrawal et al. [AGM+15b].** The construction of Agrawal et al. [AGM+15b] encodes the message $m$ with an error correcting secret sharing (ECSS) scheme to obtain $a$. Then, it samples a small number of bits from $a$ indexed by $E$, which are represented by $a_E$, and replaces $a_E$ with a (uniformly random) error $e$. This creates an erroneous codeword $c$. Observe that half of the bits of $e$ match the original entries in $a_E$ and the remaining do not. Next, an NMC of rate-1/$\mathsf{poly}(\lambda)$ encodes the consistency checks $(E, e)$ as $c_{\text{err}}$, and the final encoding is $(c, c_{\text{err}})$. The decoding algorithm error-corrects $c$ to obtain $a$ (and hence, $m$) and checks the *consistency* between $a, c, c_{\text{err}}$. For an appropriately chosen size of the set $E$, the encoding $(c, c_{\text{err}})$ is non-malleable and has rate-1.

We represent the tampered codeword and error, respectively, by $\widetilde{c}$ and $\widetilde{c_{\text{err}}}$. The security argument proceeds, roughly, as follows.

(1) The tampering on $c_{\text{err}}$ is independent of the message $m$. This argument crucially relies on the output-locality of the tampering function. The independence[3] of the ECSS is sufficiently high to permit the simulation of the tampering on $c_{\text{err}}$ independent of the message $m$.

(2) The non-malleability of the encoding $c_{\text{err}}$ ensures that $\widetilde{c_{\text{err}}}$ encodes either (a) the original $(E, e)$, or (b) an entirely unrelated $(E^*, e^*)$. The case of the tampering function creating an invalid encoding is not particularly insightful.

[3]↑An ECSS of independence $t$ has the property that any $t$ shares are uniformly and independently random.

(3.a.) Consider the case where the tampering function preserves error; namely, the $\mathsf{same}^*$ case. In this case, they argue that the only way to get a valid tampered codeword is by keeping $\widetilde{c}$ identical to $c$ and that the probability of encoding being valid independent of the original message $m$. For this argument, they perform a case analysis based on the number of bits that the tampering function *does not* directly copy from the codeword (a.k.a., the *not-copied-bits*). The tampering function, by definition, directly *copies* the remaining bits from the codeword into the tampered codeword.

If the number of not-copied-bits in the tampering function is small, then the simulation proceeds as follows. Since the tampering function has a small number of not-copied-bits, most bits in $\widetilde{c}$ are identical to their corresponding bits in $c$. These copied bits define a unique codeword (using the high distance property of ECSS[4]). Decoding succeeds if every not-copied-bit of $\widetilde{c}$ matches the corresponding bit in $c$. Moreover, decoding fails if any not-copied bit of $\widetilde{c}$ does not match the corresponding input bit in $c$. Since, the number of the not-copied-bits is small and they have output locality 1, we can simulate this check independent of the original message $m$ by leveraging the (sufficiently large) independence of the ECSS.

On the other hand, if the number of not-copied-bits is large, then they argue that the tampered codeword is invalid (w.h.p.). The following intuition underlies their argument. Due to the input-locality 1 of the tampering functions, the error $c_{\mathrm{err}}$ can influence only a few bits in $\widetilde{c}$. Consequently, there still remains a large number of bits in $\widetilde{c}$ that are not-copied-bits and are not influenced by $c_{\mathrm{err}}$. Therefore, the subset of these bits that is sampled in $E$ is also large (over the random choice of $E$). Among these indices, leveraging the high independence of the ECSS and input locality 1 of the tampering function, there is a large subset where each indexed bit in the tampered codeword independently disagrees with the tabulated $(E, e)$ with probability (at least)[5] $1/2$. So, with high probability, the tampered codeword fails the consistence check.

(3.b.) Consider the case where the tampering function replaces the error with an unrelated $(E^*, e^*)$. In this case, they argue that the only way to get valid tampered codeword is

[4]↑An ECSS with distance $d$ ensures that, for two different secrets, at least $d$ secret shares are different.
[5]↑If the tampering function flips the input bit then the probability of disagreement is 1; otherwise, the probability of disagreement is $1/2$.

by replacing $c$ by an unrelated $c^*$ that is consistent with $(E^*, e^*)$. For this argument, they perform a case analysis based on the number of output-bits of the tampering function that are non-constant (a.k.a., the *non-constant-bits*). If the number of non-constant-bits is small, then the tampered message is simulatable independent of the message due to the high independence of the ECSS and output locality 1 of tampering function. On the other hand, if the number of non-constant-bits is large, then the decoding fails with high probability. In this case, each bit in $\widetilde{c}$ that is influenced by a bit in $c$ risks creating an independent inconsistency with $(E^*, e^*)$ with probability $1/2$. Hence, if there is a large number of these bits where each of them is inconsistent with $(E^*, e^*)$ independently with probability $1/2$, then the overall codeword will be invalid with high probability. Similar to case 3.a., this argument relies on leveraging the high independence of the ECSS, input locality 1 of the tampering function, and the fact that $E$ is randomly chosen.

To summarize, two key properties are crucial to our arguments.

(A) Being non-committal to the errors. We rely on randomness of errors to argue inconsistency with tabulated errors in $c_{\text{err}}$.

(B) Independence of failure. Our objective is to identify output bits that cause decoding failure independently.

Consequently, we have the following objective.

| "Find a *large subset* of bits in $\widetilde{c}$ that *independently* fail the consistency check" while, simultaneously, "remaining *noncommittal* to (most of) the error $(E, e)$" |
|---|

In the sequel, we elaborate the unique challenges to achieve this objective against $\delta$-local functions, with $\delta > 1$, and no a priori bound on the input-locality.

**Intuition underlying Our Construction.** For a tampering function with output locality $\delta$ (referred to as a $\delta$-local function), intuitively, every bit in the tampered codeword is influenced by some bits in $c$ and some bits in $c_{\text{err}}$. The 2-local tampering functions suffice to capture these two influences and we use these to illustrate some primary challenges and key components of our construction.

Using the output locality of the tampering function, we can argue that tampering on $c_{\text{err}}$ would be independent of the message $m$. Next, we use non-malleablity of encoding $c_{\text{err}}$ to

simulate whether $\widetilde{c_{\mathsf{err}}}$ encodes (a) the original errrors, (2) an unrelated $(E^*, e^*)$, or (3) $\perp$. Let us consider the case when the tampering function preserves the original errors. In this case, we perform a case analysis on the number of *not-copied-bits*. So the first (somewhat minor) hurdle is how to define not-copied-bits for $\delta$-local functions. Since a bit in $\widetilde{c}$ can be influenced by $\delta$ bits, it is a not-copied-bit if it is not a copy for (at least) 1 out of the $2^\delta$ possible inputs. Hence, in the final argument, this bit shall fail the consistency check with probability $1/2^\delta$. Thus, as $\delta$ increases, we need to find *exponentially more* bits that *independently* fail to be consistent.

The second hurdle is that, unlike Agrawal et al. [AGM+15b], our tampering functions are *not* input-local. So, for instance, one bit in the $(c, c_{\mathsf{err}})$ can influence every bit of the tampered codeword. Therefore, even though there might be many not-copied-bits, their probability of being inconsistent is possibly correlated. To resolve this challenge, Viola [Vio11] proposed a technique to fix the values of the *highly influential* input bits (sampled from an appropriate distribution) of the tampering function. This technique, intuitively, transforms an output local tampering function into a convex combination of tampering function that are both input and output local. We use this technique to fix the highly influential bits in $c$ to be uniform random bits (relying on output locality of tampering function and independence of ECSS). However, as we discuss below, many challenges remain related to the bits in $c_{\mathsf{err}}$ that are highly influential for $\widetilde{c}$.

Consider the following representative 2-local tampering function. Each bit is $\widetilde{c}$ is influenced by corresponding bit in $c$ and a bit in $c_{\mathsf{err}}$ while ensuring that all bits in $c_{\mathsf{err}}$ have an identical number of output neighbors.

(1) If the threshold to identify "highly influential" input bits is set too low, then the procedure mentioned above might fix the entire $c_{\mathsf{err}}$, because the size of $c_{\mathsf{err}}$ is very small. Consequently, the error $(E, e)$ gets fixed. Thereafter, it is unclear how to proceed and catch any non-trivial tampering of $c$. So, the threshold to identify "highly influential" *cannot* be too low. Therefore, in this case, it is possible that no bit in $c_{\mathsf{err}}$ is fixed and $c_{\mathsf{err}}$ cumulatively influences a lot of bits in $\widetilde{c}$.

(2) Ideally, we would like that the bits we pick from $\widetilde{c}$ to argue failure do not depend on $c_{\mathsf{err}}$. However, in this case, all the bits in $\widetilde{c}$ depend on $c_{\mathsf{err}}$.

(3) Furthermore, there is another subtle issue. Conditioning on the fact that the tampered $\widetilde{c_{\text{err}}}$ encodes the same error or a fixed $(E^*, e^*)$ distorts the distribution of $c_{\text{err}}$, which, in turn, influences the distribution of the tampered $\widetilde{c}$. To summarize, the distributions $\widetilde{c}$ and $(E, e)$ are correlated when conditioned on whether the $\widetilde{c_{\text{err}}}$ encodes the same $c_{\text{err}}$ or a fixed $c_{\text{err}}$.

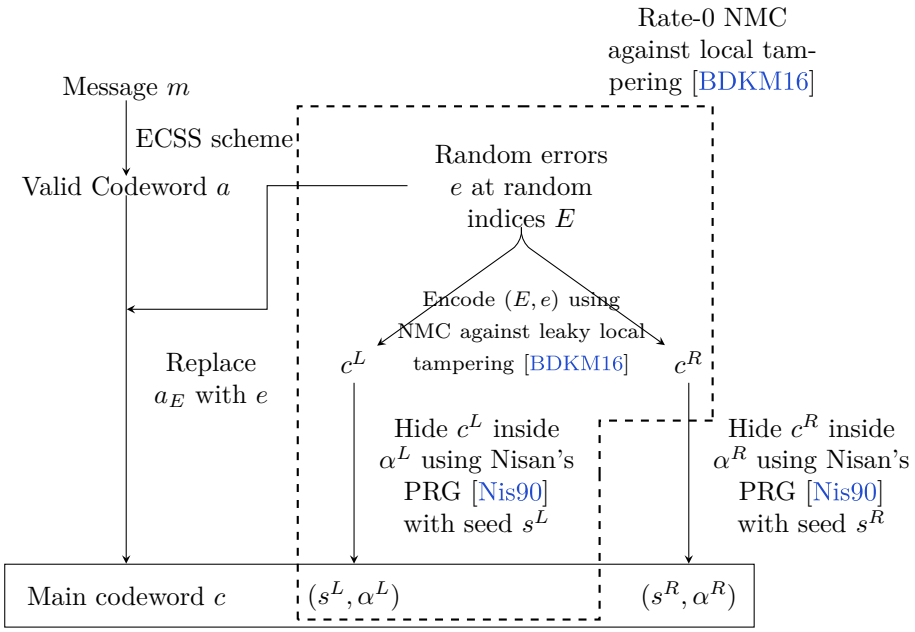



**Figure 3.1.** Block diagram of the compiler to construct NMC against local tampering.

To resolve these concerns simultaneously, the high level idea is to hide the informative bits about $(E, e)$, i.e., $c_{\text{err}}$, in a polynomially larger string, say $\alpha$ (refer to Figure 3.1 for a block diagram of our compiler). We use a PRG with a super-polynomial stretch to determine the positions with informative bits inside $\alpha$ and store the PRG seed $s$ along with $\alpha$ as the new payload. So our final codeword is $(c, s, \alpha)$.[6] We argue that for any tampering function, the number of bits from $c_{\text{err}}$ that are highly influential for $\widetilde{c}$ is small. To simulate these bits, we perform a small leakage on $c_{\text{err}}$. Since our base NMC from [BDKM18] is resilient to small leakage, we stay non-committal to $(E, e)$ even conditioned on this leakage. Note that the rest of the bits in $c_{\text{err}}$ have a bounded input locality onto $\widetilde{c}$ and hence, $c_{\text{err}}$ influences only a small subset of bits in $\widetilde{c}$.

[6]↑Similar to [BDG+18], hash function families with sufficiently high independence also suffice in this context.

Now, if we had a large number of not-copied-bits in $\widetilde{c}$, we have a large number of not-copied-bits in $\widetilde{c}$ that are not influenced by $c_{\mathrm{err}}$. But these bits might share input neighbors in $c$ and have correlated probability of failing consistency checks. Recall that we have already fixed the highly influential bits in $c$. Finally, we can use the bounded input and output locality to identify independent bits in $\widetilde{c}$ (using the greedy neighbor-of-neighbor argument of Viola [Vio11]).

This section presents only the intuitive rationale underlying the cryptographic primitives needed for our construction. There are further subtleties involved in the security arguments. Section 3.6.1 presents the full proof of our compiler using a hybrid argument.

**Remark: Limit of our approach.** We present a simple rationale for why our construction works for $\delta$-local functions, where $\delta = \xi \lg n$ and $\xi < 1$ is a positive constant. Note that in steps 3.a. and 3.b., the probability of inconsistency with the tabulated error was at least $1/2$ in a 1-local tampering function. However, the probability of inconsistency in a $\delta$-local tampering function can be as low as $2^{-\delta}$. The probability of $u$ independent consistency checks to simultaneously pass is $(1 - 2^{-\delta})^u$. We need $u = \omega\big(2^{\delta} \log n\big)$ for this quantity to be negligible. On the other hand, we have $u \leqslant n$. Consequently, we must have $2^{\delta} \ll n/\log n$, or, in particular, $\delta \ll \lg n$.

### 3.3 Preliminaries

We use $[n]$ to denote the set $\{1, 2, \ldots, n\}$. For $x = (x_1, x_2, \ldots, x_n)$ and $S \subseteq [n]$, we use $x_S$ to denote $(x_{s_1}, x_{s_2}, \ldots, x_{s_k})$, where $S = \{s_1, s_2, \ldots, s_k\}$ and $s_1 < s_2 < \cdots < s_k$. For brevity, we write $x_{-i}$ for $x_{[n]\setminus\{i\}}$. We use $U_S$ to represent the uniform distribution over the set $S$. If $\mathcal{D}$ is a distribution, we write $x \leftarrow \mathcal{D}$ to denote that $x$ is sampled according to distribution $\mathcal{D}$. The support of a distribution $\mathcal{D}$, represented by $\mathrm{Supp}(\mathcal{D})$, is the set $\{x\colon \Pr[\mathcal{D} = x] > 0\}$. For any binary strings $x, y \in \{0,1\}^n$, we use $\mathsf{HD}(x, y)$ to denote their Hamming distance defined by $\mathsf{HD}(x, y) := |\{i\colon x_i \neq y_i \text{ and } 1 \leqslant i \leqslant n\}|$.

### 3.3.1 Local Functions

Let $f\colon \{0,1\}^n \to \{0,1\}^n$ be a deterministic function. We write $f$ as $(f_1, f_2, \ldots, f_n)$ such that $f(x) = (f_1(x), f_2(x), \ldots, f_n(x))$, where each $f_i\colon \{0,1\}^n \to \{0,1\}$ and $1 \leqslant i \leqslant n$. We say that the *$i$-th bit (of the input) has influence on the $j$-th bit (of the output)* if there exists an $x^*_{-i}$ such that

$$f_j(x^*_1, \ldots, x^*_{i-1}, 0, x^*_{i+1}, \ldots, x^*_n) \neq f_j(x^*_1, \ldots, x^*_{i-1}, 1, x^*_{i+1}, \ldots, x^*_n)$$

For every output position $1 \leqslant j \leqslant n$, we define the input neighbors $\mathrm{Inp}_f(j)$ to be $\{i | 1 \leqslant i \leqslant n, i \text{ has influence on } j\}$. Similarly, for an input position $1 \leqslant i \leqslant n$, we define its output neighbors $\mathrm{Out}_f(i)$ to be $\{j | 1 \leqslant j \leqslant n, i \text{ has influence on } j\}$. We extend this notion naturally to a set of indices. We write $\mathrm{Inp}_f(S) = \cup_{s\in S} \mathrm{Inp}_f(s)$ and $\mathrm{Out}_f(S) = \cup_{s\in S} \mathrm{Out}_f(s)$.

A function $f$ has input locality $\delta$, if, for all $1 \leqslant i \leqslant n$, we have $|\mathrm{Out}_f(i)| \leqslant \delta$. Similarly, a function $f$ has output locality $\delta$, if for all $1 \leqslant j \leqslant n$, we have $|\mathrm{Inp}_f(j)| \leqslant \delta$.

**Definition 3.3.1** (Local Functions)**.** *A function $f : \{0,1\}^n \longrightarrow \{0,1\}^n$ is called a $\delta$-local function if it has output locality $\delta$.*

We use $\mathrm{Local}^\delta$ to represent the set of all such functions because $n$ shall be implicit from our context.

Recall that $\mathsf{NC}^0$ is the set of all functions $f$ such that for all $i$, $f_i$ can be computed by a circuit of fan-in 2 and constant depth. Trivially, $\mathsf{NC}^0 \subseteq \mathrm{Local}^{\mathcal{O}(1)}$.

We follow the convention in the literature and define the restriction of boolean functions as follows.

**Definition 3.3.2** (Restriction)**.** *Let $g : \{0,1\}^n \to \{0,1\}$ be a boolean function and $(I, \bar{I})$ be a partition of $[n]$. Let $x \in \{0,1\}^I$. Then, we write $g_{I|x} : \{0,1\}^n \to \{0,1\}$ for function $g$ with input of indices in $I$ being restricted to $x$. For function $f : \{0,1\}^n \to \{0,1\}^n$ such that $f = (f_1, f_2, \ldots, f_n)$ we write $f_{I|x}$ to denote $((f_1)_{I|x}, (f_2)_{I|x}, \ldots, (f_n)_{I|x})$. We say that $i \in \bar{I}$ has influence on $j$ if there exists a $x^*_{-i}$ such that $x^*_I = x$ and*

$$(f_{I|x})_j(x^*_1, \ldots, x^*_{i-1}, 0, x^*_{i+1}, \ldots, x^*_n) \neq (f_{I|x})_j(x^*_1, \ldots, x^*_{i-1}, 1, x^*_{i+1}, \ldots, x^*_n)$$

*Note that for all* $j \in [n]$, $\mathrm{Inp}_{f_{I|x}}(j) = \{i | 1 \leqslant i \leqslant n, i \text{ has influence on } j\} \subseteq \bar{I}$.

### 3.3.2 Hypergeometric Distribution

Consider a universe of size $N$ with $K$ success samples. An $(N, K, n)$-hypergeometric distribution is the probability distribution of number of success samples picked when $n$ random samples are picked from the universe without replacement. Specifically, we define the distribution as follows.

**Definition 3.3.3.** *A distribution* $\mathcal{D}$ *over the sample space* $[n]$ *is an* $(N, K, n)$*-hypergeometric distribution if, for any* $k \in [n]$*, we have*

$$\Pr[\mathcal{D} = k] = \binom{K}{k}\binom{N-K}{n-k}\binom{N}{n}^{-1}$$

Using standard coupling arguments, it is known that the hypergeometric distribution is more concentrated than the corresponding Bernoulli distribution. Consequently, we have the following tail bound.

**Lemma 3.3.1.** *([Hoe63, Chv79]) Let* $X$ *be a random variable sampled from a* $(N, K, n)$*-hypergeometric distribution. Then for any* $\varepsilon \in (0, \frac{K}{N})$*,*

$$\Pr\left[X \leqslant (K/N - \varepsilon) \cdot n\right] \leqslant \exp\left(-2\varepsilon^2 n\right)$$

The following corollary suffices for our proof.

**Corollary 3.3.1.** *Let* $A \subseteq [n]$ *be an arbitrary subset of size* $a$*. Let* $B \subseteq [n]$ *be a random subset of size* $b$*. Then*

$$\Pr\left[|A \cap B| \leqslant ab/2n\right] \leqslant \exp\left(-a^2 b/2n^2\right)$$

Note that $|A \cap B|$ is an $(n, a, b)$-hypergeometric distribution. The corollary follows from the previous lemma with $\varepsilon = a/2n$.

## 3.4 Building Blocks

In this section we describe the building blocks of our compiler.

### 3.4.1 Non-malleable Codes against Leaky Input and Output Local Tampering

Our construction relies on an encoding scheme that satisfies non-malleability against leaky input and output local tampering that we define below.

**Definition 3.4.1.** *Let* $(\mathsf{Enc}, \mathsf{Dec})$ *be a coding scheme such that* $\mathsf{Enc} : \{0,1\}^k \to \{0,1\}^{n_L} \times \{0,1\}^{n_R}$ *and* $\mathsf{Dec} : \{0,1\}^{n_L} \times \{0,1\}^{n_R} \to \{0,1\}^k$. *We call* $(\mathsf{Enc}, \mathsf{Dec})$ *a* $(\lambda, \mu, \ell_i, \ell_o)$-non-malleable code against leaky input and output local tampering *with simulation error* $\varepsilon$ *if it satisfies the following conditions.*

*Let* $\mathcal{L}^L \subseteq [n_L]$ *and* $\mathcal{L}^R \subseteq [n_R]$ *be arbitrary subsets of size at most* $\lambda n_L$ *and* $\lambda n_R$*, respectively. Consider any function* $F$ *with domain* $\{0,1\}^{|\mathcal{L}^L|} \times \{0,1\}^{|\mathcal{L}^R|}$ *that outputs a tampering function* $g : \{0,1\}^{n_L} \times \{0,1\}^{n_R} \to \{0,1\}^{n_L} \times \{0,1\}^{n_R}$ *such that for any* $x \in \{0,1\}^{|\mathcal{L}^L|}$, $y \in \{0,1\}^{|\mathcal{L}^R|}$, *and* $g = F(x, y)$

1. *The output locality of the tampering function* $g$ *is at most* $\ell_o$*, and*
2. *All but (at most)* $\mu n_L$ *input-bits of the first* $n_L$ *input-bits of* $g$ *have input locality (at most)* $\ell_i$*.*

*Then, there exists a distribution* $\mathsf{Sim}(\mathcal{L}^L, \mathcal{L}^R, F)$ *over* $\Big(\{0,1\}^k \cup \{\bot, \mathsf{same}^*\}\Big) \times \{0,1\}^{|\mathcal{L}^L|} \times \{0,1\}^{|\mathcal{L}^R|}$ *such that for any message* $m \in \{0,1\}^k$,

$$\mathsf{Tamper}^m_{\mathcal{L}^L,\mathcal{L}^R,F} \approx_\varepsilon \mathsf{Copy}(\mathsf{Sim}(\mathcal{L}^L, \mathcal{L}^R, F),\ m),\ \textit{where}$$

$$\mathsf{Tamper}^m_{\mathcal{L}^L,\mathcal{L}^R,F} := \left\{ \begin{array}{c} (c^L, c^R) \leftarrow \mathsf{Enc}(m), x := c^L_{\mathcal{L}^L}, y := c^R_{\mathcal{L}^R} \\ g := F(x, y) \\ (\widetilde{c^L}, \widetilde{c^R}) = g(c^L, c^R), \widetilde{m} = \mathsf{Dec}(\widetilde{c^L}, \widetilde{c^R}) \\ \textit{Output } (\widetilde{m}, x, y) \end{array} \right\}$$

Intuitively, leaky input and output local tampering allows the adversary to first pick a subset of indices and peek into the codeword at those places, then use this leakage as

an advice to select a output-local, (almost) input-local tampering function. Then, non-malleability against leaky input and output local tampering guarantees that the tampered message and the leakage are simulatable independent of the original message only given the position of leaked indices and the map $F$ from leakage to the tampering function. Ball et al. [BDKM16] construct this non-malleable code as an intermediate step toward their final rate-0 non-malleable codes against local tampering. As a corollary of their results, we have the following lemma, which suffices for our construction.

**Lemma 3.4.1** ([BDKM16])**.** *There exist constants $\lambda, \mu$ such that, for any $\ell_i, \ell_o = O(\log k)$, there exists an explicit and efficient $(\lambda, \mu, \ell_i, \ell_o)$-non-malleable code against leaky input and output local tampering with simulation error $\varepsilon = \mathsf{negl}(\lambda)(k)$ and rate $1/k^{o(1)}$, where $k$ is the length of the message.*

**Remark 3.4.1.** *Note that [BDKM16] reduces the problem of constructing non-malleable codes against leaky input and output local tampering to the problem of constructing non-malleable codes against 2-split-state tampering family. The rate of their final construction will be the product of the rate of the reduction, which is inverse of the locality (i.e., $1/\max(\ell_i, \ell_o)$) and the rate of the given 2-split-state non-malleable code. Instantiated with the state-of-the-art 2-split-state construction by Li [Li17, Li18], which has rate $\Omega(\log\log\log k/\log\log k)$, the final rate of [BDKM16]'s construction can be as high as $1/\mathsf{polylog}(k)$, which is $1/k^{o(1)}$ and satisfies this lemma.*

### 3.4.2 Error-Correcting Secret-Sharing Schemes

**Definition 3.4.2.** *An encoding scheme $(\mathsf{Enc}, \mathsf{Dec})$ with block length $n$ and message length $\ell$ is said to be an* $(n, \ell, d, t)$-error-correcting secret sharing scheme *(ECSS scheme) if it satisfies the following conditions.*

1. ***Distance** $d$. For any two codewords $c, c'$, $\mathsf{HD}(c, c') > d$.*
2. ***Independence** $t$. For any message $m \in \{0,1\}^\ell$ and a subset $S \subseteq [n]$ such that $|S| \leqslant t$, the distribution of $\mathsf{Enc}(m)_S$ is identical to the uniform distribution $U_{\{0,1\}^{|S|}}$.*

3. ***Error Correction*** $d/2$**.** *There exists an error-correcting function* ECorr *such that for any* $c \in \{0,1\}^n$, $\mathrm{ECorr}(c)$ *outputs a codeword* $c^*$ *such that* $\mathsf{HD}(c, c^*) \leqslant d/2$. *If no such codeword exists, then it outputs* $\perp$.

**Lemma 3.4.2.** *For every* $\zeta \in (0,1)$, *there exists an explicit* $(n, \ell, d, t)$*-ECSS scheme with* $n = (1 + o(1))\ell$ *and* $d, t \geqslant n^{1-\zeta}$.

For completeness, we provide such a construction in Appendix A.1.

### 3.4.3 Pseudorandom Generator for Finite State Machines

**Definition 3.4.3** (Finite State Machine)**.** *A finite state machine (FSM)* $Q$ *with space* $w$ *over the alphabet* $\Sigma$ *satisfies the following properties.*

1. *There exists a state-transition function* $q\colon \{0,1\}^w \times \Sigma \to \{0,1\}^w$ *that takes as input the current state* $s \in \{0,1\}^w$ *and an alphabet* $x \in \Sigma$, *and outputs the new state* $q(s, x)$.

2. *There exists a subset* $S \subseteq \{0,1\}^w$ *such that if the final state* $s \in S$ *then the FSM accepts the input and outputs 1. Otherwise, it outputs 0.*

**Definition 3.4.4.** *A function* $G\colon \{0,1\}^p \to \Sigma^u$ *is a* pseudorandom generator for FSMs *with space* $w$ *and alphabet* $\Sigma$ *with error* $\varepsilon$ *if for any distinguisher FSM* $Q$ *with space* $w$ *and alphabet* $\Sigma$ *we have*

$$\left|\Pr\left[Q\left(U_{\Sigma^u}\right) = 1\right] - \Pr\left[Q\left(G(U_{\{0,1\}^p})\right) = 1\right]\right| \leqslant \varepsilon$$

**Lemma 3.4.3** ([Nis90])**.** *There exists a constant* $\kappa > 0$ *such that for all integers* $d > 0$ *and* $u \leqslant \kappa d$, *there is an explicit pseudorandom generator* $G\colon \Sigma^{3u} \to \Sigma^{2^u}$ *for FSMs with alphabet* $\Sigma = \{0,1\}^d$ *and space* $\kappa d$ *with error* $2^{-\kappa d}$.

## 3.5 Our Compiler

In this section, we will present our compiler. That is, for all constants $\xi < 1$, given a rate-1 ECSS scheme, a rate $1/\eta^{o(1)}$ non-malleable code against leaky input and output local tampering (for $\eta$ length messages) and a PRG secure against finite state machines with appropriate parameters, we construct a rate-1 non-malleable coding scheme against all $\delta$-local

Building blocks:

- $(\mathrm{ECSS.Enc}, \mathrm{ECSS.Dec})$ is an $(n, \ell, d, t)$ ECSS scheme.
- $(\mathrm{NMEnc}_0, \mathrm{NMDec}_0)$ is a $(\lambda, \mu, \ell_i, \ell_o)$-non-malleable code against leaky input and output local tampering.
- $G : (\{0,1\}^{\log^2 n})^{3\Lambda \log n} \longrightarrow (\{0,1\}^{\log^2 n})^{n^\Lambda}$ is a PRG that fools all FSMs with space $\kappa \log^2 n$. We set $\Lambda$ below.

---

$\mathrm{NMEnc}_1(\mathsf{msg})$:

1. Sample a random $E \subseteq [n]$ of size $n^{1-\varepsilon_1}$, where $\varepsilon_1$ is a small constant.
2. For all $i \in E$, sample $e_i \leftarrow U_{\{0,1\}}$.
3. Sample $a \leftarrow \mathrm{ECSS.Enc}(\mathsf{msg})$
4. Define $c$ as $c_i = \begin{cases} a_i, & i \notin E \\ e_i, & i \in E \end{cases}$
5. Let $(c^L, c^R) \leftarrow \mathrm{NMEnc}_0(E, e)$
6. Pick seeds $s^L, s^R \xleftarrow{\$} \{0,1\}^{3\Lambda \cdot \log^3 n}$.
7. Let $\mathrm{Embed}^L, \mathrm{Embed}^R$ be as below.
   - $\alpha^L = \mathrm{Embed}^L(s^L, c^L)$
   - $\alpha^R = \mathrm{Embed}^R(s^R, c^R)$
8. Output $(c, s^L, \alpha^L, s^R, \alpha^R)$

$\mathrm{NMDec}_1(\widetilde{c}, \widetilde{s^L}, \widetilde{\alpha^L}, \widetilde{s^R}, \widetilde{\alpha^R})$:

1. Let $\mathrm{Recover}^L, \mathrm{Recover}^R$ be as below.
   - $\widetilde{c^L} = \mathrm{Recover}^L(\widetilde{s^L}, \widetilde{\alpha^L})$
   - $\widetilde{c^R} = \mathrm{Recover}^R(\widetilde{s^R}, \widetilde{\alpha^R})$
2. If $\mathrm{NMDec}_0(\widetilde{c^L}, \widetilde{c^R}) = \perp$, output $\perp$
3. (Else) $(\widetilde{E}, \widetilde{e}) = \mathrm{NMDec}_0(\widetilde{c^L}, \widetilde{c^R})$
4. If $\mathrm{ECSS.ECorr}(\widetilde{c}) = \perp$, output $\perp$
5. (Else) $\widetilde{a} = \mathrm{ECSS.ECorr}(\widetilde{c})$
6. Define $c'$ as $c'_i = \begin{cases} \widetilde{a}_i, & i \notin \widetilde{E} \\ \widetilde{e}_i, & i \in \widetilde{E} \end{cases}$
7. if $c' \neq \widetilde{c}$, output $\perp$
8. (Else) $\widetilde{\mathsf{msg}} = \mathrm{ECSS.Dec}(\widetilde{a})$
9. Output $\widetilde{\mathsf{msg}}$

---

Let lengths of $c^L$ and $c^R$ be $n^{\beta_1}$ and $n^{\beta_2}$, respectively. First, pick[a] a constant $\gamma$ s.t. $\max(\beta_1, \beta_2) < \gamma < 1$. Next, let $\tau > 0$ be a constant s.t. $\Lambda = \gamma + 2\tau < 1$.

$\underline{\mathrm{Embed}^L, \mathrm{Recover}^L}$**:** Let $\rho^L : \{0,1\}^{\log^2 n} \longrightarrow \{0,1\}$ be any function with bias[b] $2n^{-(\Lambda - \beta_1)}$. First, compute $G(s^L) = (y_1, y_2, \ldots, y_{n^\Lambda})$ s.t. each $y_i \in \{0,1\}^{\log^2 n}$ and $\mathrm{Adv}^L = (\rho^L(y_1), \rho^L(y_2), \ldots, \rho^L(y_{n^\Lambda}))$. Then, $\alpha^L = \mathrm{Embed}^L(s^L, c^L)$ is defined as:

$$\alpha^L_i := \begin{cases} c^L_j & \text{If } \mathrm{Adv}^L_i \text{ is the } j^{th} \text{ 1 in } \mathrm{Adv}^L \\ 0 & \text{Otherwise.} \end{cases}$$

To recover during decoding, compute $G(\widetilde{s^L}) = (\widetilde{y_1}, \widetilde{y_2}, \ldots, \widetilde{y_{n^\Lambda}})$ and $\widetilde{\mathrm{Adv}}^L = (\rho^L(\widetilde{y_1}), \ldots, \rho^L(\widetilde{y_{n^\Lambda}}))$. Then, if $\widetilde{\mathrm{Adv}}^L$ does not contain $\geqslant n^{\beta_1}$ many 1's, quit decoding by outputing $\perp$. Otherwise, $\widetilde{c^L} = \mathrm{Recover}^L(\widetilde{s^L}, \widetilde{\alpha^L})$ is defined as:

$$\widetilde{c^L_j} := \widetilde{\alpha^L_i} \quad \text{where } \widetilde{\mathrm{Adv}}^L_i \text{ is the } j^{th} \text{ 1 in } \widetilde{\mathrm{Adv}}^L$$

$\underline{\mathrm{Embed}^R, \mathrm{Recover}^R}$**:** Let $\rho^R : \{0,1\}^{\log^2 n} \longrightarrow \{0,1\}$ be any function with bias $2n^{-(\Lambda - \beta_2)}$. Now $\mathrm{Embed}^R, \mathrm{Recover}^R$ are defined analogously to above using $\rho^R$.

---

[a]↑This is possible because $(E, e)$ has length $\eta = n^{1-\varepsilon_1}(\log n + 1)$ and $(\mathrm{NMEnc}_0, \mathrm{NMDec}_0)$ is a $1/\eta^{o(1)}$ rate coding scheme.

[b]↑Bias of a function is the probability that output is 1 for a uniformly sampled input.

**Figure 3.2.** Our Rate-1 Non-malleable Codes against $\delta$-Local Functions

tampering family $\text{Local}^{\delta}$ for $\delta = \xi \cdot \lg n$. Here $n$ is the length of the codeword. We begin by giving some notation, specifying the building blocks used followed by our contruction overview.

**Notation:** Throughout our construction and proof, we use the notation that after the tampering is done, any variable of original codeword, for example, $a$, will have a tilde on it, i.e., $\widetilde{a}$. For example, $c$ is the original main codeword and $\widetilde{c}$ would be the tampered version of the main codeword. Thus, when we talk about bits from $c$, it refers to the input-bits of the tampering function and on the other hand, bits from $\widetilde{c}$ are output-bits of the tampering function.

**Building blocks used.** We use the following three building blocks. Let $\delta = \xi \cdot \lg n$ for $\xi < 1$ be the locality of the tampering function.

1. An $(n, \ell, d, t)$-ECSS scheme with $d, t \geqslant n^{1-\zeta}$ and $n = (1 + o(1))\ell$ for an appropriate constant $\zeta$ to be fixed later. This is provided by Lemma 3.4.2.
2. For any constant $\lambda, \mu$ and $\eta = n^{\Theta(1)}$, a $(\lambda, \mu, \ell_i, \ell_o)$-NMC against leaky input and output local tampering for messages in $\{0,1\}^{\eta}$, rate $1/\eta^{o(1)}$, $\ell_o = \delta = O(\log \eta)$, $\ell_i = 4\delta/\mu = O(\log \eta)$, simulation error negligible in $\eta$. This is provided by Lemma 3.4.1. We denote the corresponding simulator by $\text{Sim}_0$.
3. A PRG $G : (\{0,1\}^{\log^2 n})^{3\Lambda \log n} \longrightarrow (\{0,1\}^{\log^2 n})^{n^{\Lambda}}$ that is secure against all FSMs with alphabet $\Sigma = \{0,1\}^{\log^2 n}$ and space $\kappa \log^2 n$ with error $2^{-\kappa \log^2 n}$ for an appropriate constant $\Lambda$ to be fixed later. Here, $\kappa$ is a constant provided by Lemma 3.4.3 for $u = \Lambda \log n$ and $d = \log^2 n$.

**Construction Overview.** Our construction starts with encoding the message $m \in \{0,1\}^{\ell}$ using ECSS scheme $a \leftarrow \text{ECSS.Enc}(m)$ such that $a \in \{0,1\}^n$. Next, we sample a random subset $E \subseteq [n]$ of size $n^{1-\varepsilon_1}$ for a small constant $\varepsilon_1$ specified later. Next, for each index $i \in E$, we sample a random bit $e_i$. These will be our planted errors. Then, all bits at $E$ in codeword $a$ are replaced by these random bits $e_i$ to produce $c$. We refer to this an erroneous codeword $c$ as the main codeword. We note that a bit at index in $E$ has probability $1/2$ of being an error.

Next, for the second part of our codeword, we record the error indices $E$ as well as planted errors $e = (e_1, e_2, \ldots, e_{|E|})$ using (poor-rate) non-malleable codes against leaky input and output local tampering. We sample $(c^L, c^R) \leftarrow \mathrm{NMEnc}_0(E, e)$. Finally, we hide the codeword $(c^L, c^R)$ inside a larger code $(\alpha^L, \alpha^R)$ at pseudorandom locations as follows: We will sample two seeds $s^L, s^R$ of appropriate length (See Figure 3.2). And invoke our pseudorandom generator $G$ on $s^L$ (resp., $s^R$) and use appropriate bias function $\rho^L$ (resp., $\rho^R$) to generate advice string $\mathrm{Adv}^L$ (resp., $\mathrm{Adv}^R$). At a high level, positions having a 1 in the advice string will store an actual bit of the code, and positions with 0 will store a redundant 0. Intuitively, this step ensures that when bits from $\alpha^L$ or $\alpha^R$ are used for tampering, most of these bits would be redundant 0's. Our final codeword is $(c, s^L, \alpha^L, s^R, \alpha^R)$.

Conversely, to decode, we use seeds $\widetilde{s^L}, \widetilde{s^R}$ to determine the indices of $\widetilde{c^L}, \widetilde{c^R}$ in $\widetilde{\alpha^L}, \widetilde{\alpha^R}$. Then, we decode $(\widetilde{c^L}, \widetilde{c^R})$ to get the error index set $\widetilde{E}$ and error bits $\widetilde{e}$. Next, we compare $\widetilde{c}$ with planted errors $(\widetilde{E}, \widetilde{e})$ to check (1) whether all the bits from $\widetilde{c}$ with index in $\widetilde{E}$ and $\widetilde{e}$ are equal; (2) we error correct $\widetilde{c}$ to obtain correct codeword $\widetilde{a}$ and check whether all the errors in $\widetilde{c}$ were recorded in $\widetilde{E}$. If both conditions are satisfied, we will consider the codeword valid and output the decoding of $\widetilde{a}$ as the decoded message.

**Setting the parameters.** Next, we will set the various constants used in our construction (as well as proof of non-malleability).

- $\lambda, \mu$: We pick constants $\lambda, \mu$ arbitrarily.
- $\Lambda, \gamma, \tau$: Let $|c^L| = n^{\beta_1}$ and $|c^R| = n^{\beta_2}$. Since $\eta = |(E, e)| = n^{1-\varepsilon_1}(\log n)$ and rate of $\mathrm{NMEnc}_0$ is $1/\eta^{o(1)}$, we have that $\max(\beta_1, \beta_2) < 1$. We pick positive constants $\gamma, \tau$ such that $\max(\beta_1, \beta_2) < \gamma < 1$ and $\gamma + 2\tau < 1$. Set $\Lambda = \gamma + 2\tau$.
- $\varepsilon_1, \varepsilon_2$: The number of erroneous indices $|E| = n^{1-\varepsilon_1}$. In our security hybrids, we have another small constant $\varepsilon_2$ and we require $\varepsilon_1 + 2\varepsilon_2 < 1 - \xi$, where $\xi$ is defined by the tampering family. Hence, given $\xi$, we pick two positive constants satisfying the condition.
- $\zeta$: In our construction, we use an $(n, \ell, d, t)$-ECSS scheme with $d, t \geqslant n^{1-\zeta}$. In our security proof, we require $\zeta < \min(\varepsilon_1, \varepsilon_2, \tau, 1 - \Lambda)$ and hence, $\zeta$ can be picked as a sufficiently small positive constant satisfying the constraint.

**Theorem 3.5.1.** *Let $\{0,1\}^\ell$ be the message space and $\delta = \xi \cdot \lg n$, for some constant $\xi < 1$. There exists an explicit and efficient rate-1 NMC against $\mathrm{Local}^\delta$ with simulation error that is negligible in $n$ and uses the following primitives in a black-box manner.*

1. *For appropriate $\zeta > 0$, an $(n, \ell, d, t)$-ECSS scheme with $d, t \geqslant n^{1-\zeta}$ and $n = (1 + o(1))\ell$.*
2. *For some constant $\lambda, \mu$ and $\eta = n^{\Theta(1)}$, a $(\lambda, \mu, \ell_i, \ell_o)$-NMC against leaky input and output local tampering for messages in $\{0,1\}^\eta$, rate $1/\eta^{o(1)}$, $\ell_o = O(\log \eta)$, $\ell_i = O(\log \eta)$, simulation error negligible in $\eta$.*
3. *For some constant $\Lambda > 0$, a PRG $G : (\{0,1\}^{\log^2 n})^{3\Lambda \log n} \longrightarrow (\{0,1\}^{\log^2 n})^{n^\Lambda}$ that is secure against FSM with alphabet size $\log^2 n$ and space $\Theta(\log^2 n)$ with error that is negligible in $n$.*

The above theorem when instantiated with Lemma 3.4.2, Lemma 3.4.1 and Lemma 3.4.3 gives following theorem.

**Theorem 3.5.2.** *For all constants $\xi < 1$, there exists an explicit rate-1 non-malleable code against $\mathrm{Local}^{\xi \cdot \lg n}$ with negligible in $n$ simulation error, where $n$ is the length of the codeword.*

In particular, this implies an explicit rate-1 non-malleable code against $\mathsf{NC}^0$ tampering.

### 3.5.1 Proof of Theorem 3.5.1

Here, we will prove that the our construction has rate-1 and perfect correctness. We provide proof of non-malleability in the next section.

**Rate of our construction.** Our codeword is $(c, s^L, \alpha^L, s^R, \alpha^R)$. Note that our main codeword $c$ has length $n = \ell + o(\ell)$. Next, $|s^L| = |s^R| = 3\Lambda \log^3 n$. And, $—\alpha^L| = |\alpha^R| = n^\Lambda$. Since, $\Lambda = \gamma + 2\tau < 1$ (see parameter setting above), the overall codeword has length $\ell + o(\ell)$.

**Correctness.** We first argue that our scheme has statistical correctness, and then show how the scheme in Figure 3.2 can be tweaked slightly to give perfect correctness. It is easy to see that the correctness of our scheme in Figure 3.2 is broken only when $\mathrm{Adv}^L$ does not have enough number of 1's to store all of $c^L$ in $\alpha^L$ or similarly, when $\mathrm{Adv}^R$ does not have enough number of 1's to store all of $c^R$ in $\alpha^R$. If this happens, the decoding algorithm would output $\perp$. Note that whether this event happens or not depends on the choice of seeds $s^L$ and $s^R$

only. We prove the following lemma that states that probability of this event happening is negligible.

**Lemma 3.5.1.** *With probability at least* $1 - 2^{-\Omega(\log^2 n)}$ *over the random choice of* $s_L$ *and* $s_R$, $\alpha^L$ *and* $\alpha^R$ *will contain all the bits from* $c^L$ *and* $c^R$.

*Proof.* We will prove the lemma for $(s^L, \alpha^L)$ and same argument holds for $(s^R, \alpha^R)$. We first show that the lemma holds when $G$ is a random function. Next, we argue that if lemma does not hold for a PRG $G$, then there exists a distinguisher FSM $Q$ with space $\kappa \log^2 n$ that breaks PRG security with non-negligible probability in $n$.

Firstly, when $G(s^L)$ outputs uniform random string, the expected number of 1's in $\mathrm{Adv}^L$ is $n^{\Lambda} \cdot 2n^{-(\Lambda - \beta_1)} = 2n^{\beta_1}$. Next, using Chernoff bound, with probability at least $1 - \exp(-\Theta(n^{\beta_1}))$, there are at least $n^{\beta_1}$ many 1's in $\mathrm{Adv}^L$ and hence, $\alpha^L$ will contain all the bits from $c^L$.

Now suppose that the lemma does not hold when we use PRG $G$ that fools FSMs with space $\kappa \log^2 n$. Consider the following FSM $Q$ that takes $(y_1, y_2, \ldots, y_{n^{\Lambda}})$ as input and a state in $Q$ stores $\mathsf{ctr}$, which denotes number of indices $i$ for which $\rho^L(y_i)$ output 1. The final output of $Q$ is 1 when $\mathsf{ctr} \geqslant n^{\beta_1}$. Clearly, by our argument above, on a true uniform string, $Q$ will output 1 with probability at least $1 - \exp(-\Theta(n^{\beta_1}))$. If this lemma is incorrect for a PRG $G$, $Q$ will output 1 with probability at most $1 - 2^{-\Omega(\log^2 n)}$ and hence $Q$ will break the underlying PRG with success probability greater than $2^{-\Omega(\log^2 n)}$. Finally, note that $Q$ only needs $\Lambda \log n < \kappa \log^2 n$ space to record $A$. This completes the proof. □

**Getting perfect correctness.** We can tweak our scheme slightly to give perfect correctness as follows: If $s^L$ or $s^R$ is bad, i.e., $(\alpha^L, \alpha^R)$ will not contain all bits in $(c^L, c^R)$, then we ignore $\mathrm{Adv}^L, \mathrm{Adv}^R$ and store the codeword in default location. More precisely, we store $c^L$ in first $|c^L|$ locations in $\alpha^L$ and similarly for $c^R$. It is easy to see that this gives perfect correctness. In the proof of non-malleability, our simulator can simply give up when this case happens. (Since $s^L, s^R$ are uniform seeds independent of the message, it is easy to check for this case.) This would increase the simulation error by the probability of this event occurring. But, above Lemma 3.5.1 proves that this happens with negligible probability. Hence, this only increases the simulation error by $\mathsf{negl}(\lambda)(n)$.

### 3.6 Proof of Non-malleability of Our Compiler

**Non-malleability.** Recall that to prove non-malleability of the resulting scheme against $\delta$-local tampering family $\mathrm{Local}^{\delta}$, we need to show that for any $f \in \mathrm{Local}^{\delta}$, there exists a simulator $\mathsf{Sim}_1(f)$ such that, for all message $m \in \{0,1\}^{\ell}$, we have the following

$$\left\{\begin{array}{c}(c, s^L, \alpha^L, s^R, \alpha^R) \leftarrow \mathrm{NMEnc}_1(m) \\ (\widetilde{c}, \widetilde{s^L}, \widetilde{\alpha^L}, \widetilde{s^R}, \widetilde{\alpha^R}) = f(c, s^L, \alpha^L, s^R, \alpha^R) \\ \widetilde{m} = \mathrm{NMDec}_1(\widetilde{c}, \widetilde{s^L}, \widetilde{\alpha^L}, \widetilde{s^R}, \widetilde{\alpha^R}) \\ \text{Output } \widetilde{m}\end{array}\right\} = \mathsf{Tamper}_f^m \approx_{\varepsilon} \mathsf{Copy}(\mathsf{Sim}_1(f), m)$$

Our simulator is formally defined in Figure 3.3. In the simulator and the hybrids, $n_e = |(s^L, \alpha^L, s^R, \alpha^R)|$. A detailed proof using a sequence of indistinguishable hybrids is presented in the next section. We shall use the following lemma in our hybrid argument. We present the proof of Lemma 3.6.1 in Appendix A.2.

**Lemma 3.6.1.** *For any $\delta$-local tampering function, with probability at least $1 - 2^{-\Omega(\log^2 n)}$ over the random choice of $s_L$ and $s_R$, the following conditions hold.*
*(1) At most $\mu n^{\beta_1}$ bits from $c^L$ will have input locality higher than $4\delta/\mu$ onto $\widetilde{\alpha^R}$;*
*(2) Number of bits in $c^L$ and $c^R$ that have greater than $n^{1-\gamma-\tau}$ input locality onto $\widetilde{c}$ are bounded by $4\delta n^{\beta_1-\tau}$ and $4\delta n^{\beta_2-\tau}$, respectively.*
*And as a consequence, we have*
*(3) Number of bits in $\widetilde{c}$ that are influenced by low input locality bits from $c^L$ and $c^R$ are bounded by $n^{\beta_1} \cdot n^{1-\gamma-\tau} = o(n^{1-\tau})$ and $n^{\beta_2} \cdot n^{1-\gamma-\tau} = o(n^{1-\tau})$, respectively.*

#### 3.6.1 Detailed hybrid argument

In this section, we are going to use a series of statistically close hybrids to prove that $\mathsf{Tamper}_f^{\mathsf{msg}}$ and $\mathsf{Copy}\big(\mathsf{Sim}_1(f), \mathsf{msg}\big)$ are indistinguishable. Throughout this subsection, we use the following color/highlight notation. In a current hybrid, the text in red denotes the changes from the previous hybrid. The text in shaded part represents the steps that will

1. Let $P = \{i | i \in [n], |\mathrm{Out}_f(i) \cap [n]| \geqslant n^{\varepsilon_2}\}$
2. Let $Q = \{i | i \in [n], \mathrm{Out}_f(i) \backslash [n] \neq \emptyset\}$
3. Let $X = P \cup Q$. Sample $a_X \leftarrow U_{\{0,1\}^{|X|}}$
4. Sample a random $E_1 \subseteq X$ s.t. $|E_1| \leftarrow (n, |X|, n^{1-\varepsilon_1})$-hypergeometric distribution.
5. For all $i \in E_1$, sample $e_i \leftarrow U_{\{0,1\}}$.
6. For all $i \in E_1$, replace $a_i$ with $e_i$, we get $c_X$.
7. Sample seeds $s^L, s^R$ uniformly from $\{0,1\}^{3\Lambda \log^3 n}$.
8. Given $s^L$ (resp., $s^R$), indices of $c^L$ (resp., $c^R$) in $\alpha^L$ (resp., $\alpha^R$) are determined.
   Let $\mathrm{Bad}^L$={Indices of $c^L$ with more than $n^{1-\gamma-\tau}$ output neighbors in $\widetilde{c}$},
   $\mathrm{Leak}^L$={Indices of $c^L$ with output neighbors in either $\widetilde{s^L}$ or $\widetilde{s^R}$},
   $\mathrm{Bad}^R$={Indices in $c^R$ with more than $n^{1-\gamma-\tau}$ output neighbors in $\widetilde{c}$}, and
   $\mathrm{Leak}^R$={Indices in $c^R$ with output neighbors in either $\widetilde{s^L}$ or $\widetilde{s^R}$}
9. Let $\mathcal{L}^L = \mathrm{Bad}^L \cup \mathrm{Leak}^L$ and $\mathcal{L}^R = \mathrm{Bad}^R \cup \mathrm{Leak}^R$.
10. Let $f_0$ be the following mapping from leakage at $(\mathcal{L}^L, \mathcal{L}^R)$ to tampering function $g$ for $\mathrm{NMEnc}_0$: First, use $(s^L, s^R)$, leakage at $(\mathrm{Leak}^L, \mathrm{Leak}^R)$ and $c_Q$ to compute $\widetilde{s^L}$ and $\widetilde{s^R}$. These determine indices of $\widetilde{c^L}$ and $\widetilde{c^R}$ in $\widetilde{\alpha^L}$ and $\widetilde{\alpha^R}$. Then, define $g$ to be the tampering function from indices of $(c^L, c^R)$ to indices of $(\widetilde{c^L}, \widetilde{c^R})$.
11. If $(|\mathcal{L}^L| \geqslant \lambda n^{\beta_1})$ or $(|\mathcal{L}^R| \geqslant \lambda n^{\beta_2})$ or ($f_0$ does not satisfy Definition 3.4.1), output $\perp$
12. (Else) $(\mathrm{ans}, x, y) = \mathrm{Sim}_0(\mathcal{L}^L, \mathcal{L}^R, f_0)$.
13. Let $S^L, S^R$ denote indices of $s^L, s^R$. Define function $h$ as a restriction of $f_1$:

$$h := (f_1)_{(X, S^L, S^R, \mathcal{L}^L, \mathcal{L}^R) | (c_X, s^L, s^R, x, y)} \qquad \text{(See Definition 3.3.2)}$$

14. $V := \{i | i \in [n], \mathrm{Inp}_h(i) \neq \emptyset\}$
15. $W := \{i | i \in [n], \mathrm{Inp}_h(i) \backslash [n] \neq \emptyset\}$
16. $Z := \{i \in [n] | \exists z \in \{0,1\}^{n+n_e}, z_{(X, S^L, S^R, \mathcal{L}^L, \mathcal{L}^R)} = (c_X, s^L, s^R, x, y), h_i(\mathrm{z}) \neq z_i\}$
17. Sample $a \leftarrow \mathrm{ECSS.Enc}(0^\ell) | (\mathrm{ECSS.Enc}(0^\ell))_X = a_X$
18. Sample a random $E_2 \subseteq [n] \backslash X$ of size $n^{1-\varepsilon_1} - |E_1|$, let $E = E_1 \cup E_2$
19. For all $i \in E_2$, sample $e_i \leftarrow U_{\{0,1\}}$
20. Define $c$ as $c_i = \begin{cases} a_i, & i \notin E \\ e_i, & i \in E \end{cases}$
21. $(\widetilde{E}, \widetilde{e}) = \mathsf{Copy}(\mathrm{Sim}_0(\mathcal{L}^L, \mathcal{L}^R, f_0), (E, e))$
22. $(c^L, c^R) \leftarrow \mathrm{NMEnc}_0(E, e)$ s.t. $\mathrm{NMDec}_0(g(c^L, c^R)) = (\widetilde{E}, \widetilde{e})$ and $c^L_{\mathcal{L}^L} = x$, $c^R_{\mathcal{L}^R} = y$
23. $\alpha^L = \mathrm{Embed}^L(s^L, c^L)$, $\alpha^R = \mathrm{Embed}^R(s^R, c^R)$
24. $\widetilde{c} = f_1(c, s^L, \alpha^L, s^R, \alpha^R)$
25. If ans =
    - $\perp$: Output $\perp$
    - $\mathsf{same}^*$: If $|Z \backslash (W \cup X)| \geqslant n^{1-\varepsilon_2}$, output $\perp$
      (Else) If $\widetilde{c}_Z = c_Z$, output $\mathsf{same}^*$; (Else) Output $\perp$.
    - $(E^*, e^*)$: If $|V \backslash W| \geqslant n^{1-\varepsilon_2}$, output $\perp$
      (Else) If $\mathrm{ECSS.ECorr}(\widetilde{c}) \perp$, output $\perp$;
      (Else) $\widetilde{a} = \mathrm{ECSS.ECorr}(\widetilde{c})$
      Define $c'$ as $c'_i = \begin{cases} \widetilde{a_i}, & i \notin \widetilde{E} \\ \widetilde{e_i}, & i \in \widetilde{E} \end{cases}$
      If $c' \neq \widetilde{c}$, output $\perp$; (Else) Output $\widetilde{m} = \mathrm{ECSS.Dec}(\widetilde{a})$

**Figure 3.3.** Simulator $\mathsf{Sim}_1(f)$

be replaced by red part of the next hybrid. We call $c$ (resp., $\widetilde{c}$) the main codeword and $(s^L, \alpha^L, s^R, \alpha^R)$ (resp., $(\widetilde{s^L}, \widetilde{\alpha^L}, \widetilde{s^R}, \widetilde{\alpha^R})$) the error codeword.

$H_1(f, m)$ : Our first hybrid is the real world $\mathsf{Tamper}_f^m$, we simply open up the definition of $\mathrm{NMEnc}_1$ and $\mathrm{NMDec}_1$ and write tampering function $f$ as $(f_1, f_2)$. Both functions are given as input the entire codeword and $f_1$ is doing the tampering on the main codeword, i.e., outputs $\widetilde{c}$, while $f_2$ is doing the tampering on the error codeword, i.e., outputs $(\widetilde{s^L}, \widetilde{\alpha^L}, \widetilde{s^R}, \widetilde{\alpha^R})$. This way of writing $f$ would be useful in later hybrids.

$H_1(f, m)$:

1. Sample a random $E \subseteq [n]$ of size $n^{1-\varepsilon_1}$
2. For all $i \in E$, sample $e_i \leftarrow U_{\{0,1\}}$
3. Sample $a \leftarrow \mathrm{ECSS.Enc}(m)$
4. Define $c$ as $c_i = \begin{cases} a_i, & i \notin E \\ e_i, & i \in E \end{cases}$
5. Let $(c^L, c^R) \leftarrow \mathrm{NMEnc}_0(E, e)$
6. Sample seeds $s^L, s^R$ uniformly from $\{0,1\}^{3\Lambda \log^3 n}$
7. $\alpha^L = \mathrm{Embed}^L(s^L, c^L)$ and $\alpha^R = \mathrm{Embed}^R(s^R, c^R)$
8. $\widetilde{c} = f_1(c, s^L, \alpha^L, s^R, \alpha^R)$
9. $(\widetilde{s^L}, \widetilde{\alpha^L}, \widetilde{s^R}, \widetilde{\alpha^R}) = f_2(c, s^L, \alpha^L, s^R, \alpha^R)$
10. $\widetilde{c^L} = \mathrm{Recover}^L(\widetilde{s^L}, \widetilde{\alpha^L})$ and $\widetilde{c^R} = \mathrm{Recover}^R(\widetilde{s^R}, \widetilde{\alpha^R})$
11. If $\mathrm{NMDec}_0(\widetilde{c^L}, \widetilde{c^R}) = \perp$, output $\perp$; (Else) $(\widetilde{E}, \widetilde{e}) = \mathrm{NMDec}_0(\widetilde{c^L}, \widetilde{c^R})$
12. If $\mathrm{ECSS.ECorr}(\widetilde{c}) = \perp$, output $\perp$; (Else) $\widetilde{a} = \mathrm{ECSS.ECorr}(\widetilde{c})$
13. Define $c'$ as $c'_i = \begin{cases} \widetilde{a}_i, & i \notin \widetilde{E} \\ \widetilde{e}_i, & i \in \widetilde{E} \end{cases}$
14. If $c' \neq \widetilde{c}$, output $\perp$; (Else) $\widetilde{m} = \mathrm{ECSS.Dec}(\widetilde{a})$
15. Output $\widetilde{m}$

$H_2(f, m)$ : In the next hybrid $H_2$, we change the way we sample ECSS codeword of $m$. We define two subsets of indices $P$ and $Q$. Intuitively, $P$ is the *popular* input bits of the main codeword, i.e., bits in $c$ that influence more than $n^{\varepsilon_2}$ bits of $\widetilde{c}$. And $Q$ is the set of bits in main codeword $c$ that influence the error codeword $(\widetilde{s^L}, \widetilde{\alpha^L}, \widetilde{s^R}, \widetilde{\alpha^R})$. Now, let $X = P \cup Q$. We first sample a uniform string $a_X$ of length $|X|$ and then sample $a \leftarrow \mathrm{ECSS.Enc}(m)$ condition

on that $\mathrm{ECSS.Enc}(m)_X = a_X$. We argue that this does not change the distribution of $a$ and hence it is identical to the previous hybrid.

To argue this we use the independence property of our ECSS scheme. In particular, since $t \geqslant n^{1-\zeta}$, the distribution of $a_X$ is indeed uniform as long as long as $|X| = o(n^{1-\zeta})$. Now, $|P|$ can be bound as follows: The total number of input neighbors of $\widetilde{c}$ is $\delta n$ and at most $\delta n^{1-\varepsilon_2}$ many bits in $c$ can influence more than $n^{\varepsilon_2}$ bits from $\widetilde{c}$. Hence $|P| = o(n^{1-\zeta})$ as long as we pick $\boxed{\zeta < \varepsilon_2}$. Next, the length of the error codeword is $|s^L| + |\alpha^L| + |s^R| + |\alpha^R| = O(n^\Lambda)$ and hence, by output locality $\delta$, the size of $Q$ is at most $\delta \cdot \mathcal{O}\left(n^\Lambda\right) = \mathrm{o}\left(n^{1-\zeta}\right)$ as long as $\boxed{\zeta < 1 - \Lambda}$.

$H_2(f, m)$:

1. Let $P = \{i | i \in [n], |\mathrm{Out}_f(i) \cap [n]| \geqslant n^{\varepsilon_2}\}$
2. Let $Q = \{i | i \in [n], \mathrm{Out}_f(i) \backslash [n] \neq \emptyset\}$
3. Let $X = P \cup Q$. Sample $a_X \leftarrow U_{\{0,1\}^{|X|}}$
4. Sample $a \leftarrow \mathrm{ECSS.Enc}(m) | (\mathrm{ECSS.Enc}(m))_X = a_X$
5. Sample a random $E \subseteq [n]$ of size $n^{1-\varepsilon_1}$
6. For all $i \in E$, sample $e_i \leftarrow U_{\{0,1\}}$
7. Define $c$ as $c_i = \begin{cases} a_i, & i \notin E \\ e_i, & i \in E \end{cases}$
8. Let $(c^L, c^R) \leftarrow \mathrm{NMEnc}_0(E, e)$
9. Sample seeds $s^L, s^R$ uniformly from $\{0,1\}^{3\Lambda \log^3 n}$
10. $\alpha^L = \mathrm{Embed}^L(s^L, c^L)$ and $\alpha^R = \mathrm{Embed}^R(s^R, c^R)$
11. $\widetilde{c} = f_1(c, s^L, \alpha^L, s^R, \alpha^R)$
12. $(\widetilde{s^L}, \widetilde{\alpha^L}, \widetilde{s^R}, \widetilde{\alpha^R}) = f_2(c, s^L, \alpha^L, s^R, \alpha^R)$
13. $\widetilde{c^L} = \mathrm{Recover}^L(\widetilde{s^L}, \widetilde{\alpha^L})$ and $\widetilde{c^R} = \mathrm{Recover}^R(\widetilde{s^R}, \widetilde{\alpha^R})$
14. If $\mathrm{NMDec}_0(\widetilde{c^L}, \widetilde{c^R}) = \perp$, output $\perp$; (Else) $(\widetilde{E}, \widetilde{e}) = \mathrm{NMDec}_0(\widetilde{c^L}, \widetilde{c^R})$
15. If $\mathrm{ECSS.ECorr}(\widetilde{c}) = \perp$, output $\perp$; (Else) $\widetilde{a} = \mathrm{ECSS.ECorr}(\widetilde{c})$
16. Define $c'$ as $c'_i = \begin{cases} \widetilde{a_i}, & i \notin \widetilde{E} \\ \widetilde{e_i}, & i \in \widetilde{E} \end{cases}$
17. If $c' \neq \widetilde{c}$, output $\perp$; (Else) $\widetilde{m} = \mathrm{ECSS.Dec}(\widetilde{a})$
18. Output $\widetilde{m}$

$H_3(f, m)$ : In the next hybrid $H_3$, we rewrite the way how $(\widetilde{E}, \widetilde{e})$ is generated from $(E, e)$ given seeds $s^L$ and $s^R$. Here, we would generate $(\widetilde{E}, \widetilde{e})$ as output of a tampering experiment on $(E, e)$ with an appropriate tampering function from the leaky input and output local tampering family. Note that $(E, e)$ is first encoded to $(c^L, c^R)$ and then is hidden among $(\alpha^L, \alpha^R)$ using seeds $s^L, s^R$. We note that if we are given the seed $s^L$ and $s^R$, the places where $c^L$ are $c^R$ are stored among $\alpha^L$ and $\alpha^R$ is known. Similarly, if we know $\widetilde{s^L}$ and $\widetilde{s^R}$, the places where $\widetilde{c^L}$ and $\widetilde{c^R}$ are stored among $\widetilde{\alpha^L}$ and $\widetilde{\alpha^R}$ are also known. Therefore, we define $\mathrm{Leak}^L$ and $\mathrm{Leak}^R$ as the input neighbors of both $\widetilde{s^L}$ and $\widetilde{s^R}$ from $c^L$ and $c^R$ respectively. Now let $f_0$ be the mapping that given the leakage $\mathrm{Leak}^L$ and $\mathrm{Leak}^R$, first computes[7] $\widetilde{s^L}$ and $\widetilde{s^R}$, and then outputs the tampering function $g$. Now that we know indices of $(c^L, c^R)$ and $(\widetilde{c^L}, \widetilde{c^R})$, function $g$ maps $(c^L, c^R)$ to $(\widetilde{c^L}, \widetilde{c^R})$.[8] We note that leaking bits at $\mathrm{Bad}^L$ and $\mathrm{Bad}^R$ from $c^L$ and $c^R$ would be used in later hybrids. So the total leakage from $c^L$ and $c^R$ are $\mathcal{L}^L = \mathrm{Leak}^L \cup \mathrm{Bad}^L$ and $\mathcal{L}^R = \mathrm{Leak}^R \cup \mathrm{Bad}^R$.

Now we need to argue that the tampering $f_0$ and leakage $\mathcal{L}^L, \mathcal{L}^R$ forms a valid tampering experiment onto our base NMC against leaky input and output local tampering. It is easy to see that if it is valid, then the two hybrids are identical. When they are not valid we output $\perp$ in this hybrid and we need to argue that probability of output $\perp$ due this is negligible.

Firstly, $f_0$ might not satisfy Definition 3.4.1 if one of the following happens: (i) Not all the bits from $c^L$, $c^R$ are contained in $\alpha^L$ and $\alpha^R$, respectively and thus, $f_0$ cannot produce function $g$; (ii) $g$ has output locality higher than $\ell_o = \delta$; (iii) under $g$, more than $\mu n^{\beta_1}$ many bits from $c^L$ have input locality higher than $\ell_i = 4\delta/\mu$ to $\widetilde{c^R}$. Note that our tampering function $f$ is $\delta$-local and therefore, the output function $g$ will also be $\delta$-local, thus (ii) will never happen. And the probability of (i) or (iii) happening is negligibe as guaranteed by Lemma 3.5.1 and (1) from Lemma 3.6.1, respectively.

We bound the size of the leakage $|\mathcal{L}^L| = |\,\mathrm{Leak}^L \cup \mathrm{Bad}^L\,|$ by $o(n^{\beta_1})$. First, we observe that our seeds $s^L$ and $s^R$ are of length $\mathcal{O}\left(\log^3 n\right)$ and hence $|\mathrm{Leak}^L|$ is at most $\mathcal{O}\left(\delta \log^3 n\right) = \mathrm{o}\left(n^{\beta_1}\right)$. And the size of $\mathrm{Bad}^L$ is $\mathrm{o}\left(n^{\beta_1}\right)$ is guaranteed by (2) of Lemma 3.6.1. The argument

[7]↑Note that at this point, the original seed $s^L$ and $s^R$ and their input neighbors $c_Q$ from main codeword $c$ is already fixed.

[8]↑If $\widetilde{c^L}$ or $\widetilde{c^R}$ are not contained in $\widetilde{\alpha^L}$ or $\widetilde{\alpha^R}$, $f_0$ will simply set $g$ to be a $\perp$ function.

for $\mathcal{L}^R$ is analogous to $\mathcal{L}^L$. This proves that this hybrid and the previous one are $2^{-\Omega(\log^2 n)}$-close.

Note that we still need the error codeword $(s^L, \alpha^L, s^R, \alpha^R)$ to do the tampering $f_1$ onto $\widetilde{c}$. Hence, we sample $c^L$ and $c^R$ under the condition that the tampering experiment outputs $(\widetilde{E}, \widetilde{e}, x, y)$ and construct the error codeword as defined by our compiler.

$H_3(f, m)$:

1. Let $P = \{i | i \in [n], |\mathrm{Out}_f(i) \cap [n]| \geqslant n^{\varepsilon_2}\}$
2. Let $Q = \{i | i \in [n], \mathrm{Out}_f(i) \backslash [n] \neq \emptyset\}$
3. Let $X = P \cup Q$. Sample $a_X \leftarrow U_{\{0,1\}^{|X|}}$
4. Sample $a \leftarrow \mathrm{ECSS.Enc}(m) | (\mathrm{ECSS.Enc}(m))_X = a_X$
5. Sample a random $E \subseteq [n]$ of size $n^{1-\varepsilon_1}$
6. For all $i \in E$, sample $e_i \leftarrow U_{\{0,1\}}$
7. Define $c$ as $c_i = \begin{cases} a_i, & i \notin E \\ e_i, & i \in E \end{cases}$
8. Sample seeds $s^L, s^R$ uniformly from $\{0,1\}^{3\Lambda \log^3 n}$
9. Given $s^L$ (resp., $s^R$), indices of $c^L$ (resp., $c^R$) in $\alpha^L$ (resp., $\alpha^R$) are determined.
   Let $\mathrm{Bad}^L$={Indices of $c^L$ with more than $n^{1-\gamma-\tau}$ output neighbors in $\widetilde{c}$},
   $\mathrm{Leak}^L$={Indices of $c^L$ with output neighbors in either $\widetilde{s^L}$ or $\widetilde{s^R}$},
   $\mathrm{Bad}^R$={Indices in $c^R$ with more than $n^{1-\gamma-\tau}$ output neighbors in $\widetilde{c}$}, and
   $\mathrm{Leak}^R$={Indices in $c^R$ with output neighbors in either $\widetilde{s^L}$ or $\widetilde{s^R}$}
10. Let $\mathcal{L}^L = \mathrm{Bad}^L \cup \mathrm{Leak}^L$ and $\mathcal{L}^R = \mathrm{Bad}^R \cup \mathrm{Leak}^R$.
11. Let $f_0$ be the following mapping from leakage at $(\mathcal{L}^L, \mathcal{L}^R)$ to tampering function $g$ for $\mathrm{NMEnc}_0$: First, use $(s^L, s^R)$, leakage at $(\mathrm{Leak}^L, \mathrm{Leak}^R)$ and $c_Q$ to compute $\widetilde{s^L}$ and $\widetilde{s^R}$. These determine indices of $\widetilde{c^L}$ and $\widetilde{c^R}$ in $\widetilde{\alpha^L}$ and $\widetilde{\alpha^R}$. Then, define $g$ to be the tampering function from indices of $(c^L, c^R)$ to indices of $(\widetilde{c^L}, \widetilde{c^R})$.
12. If $(|\mathcal{L}^L| \geqslant \lambda n^{\beta_1})$ or $(|\mathcal{L}^R| \geqslant \lambda n^{\beta_2})$ or ($f_0$ does not satisfy Definition 3.4.1), output $\perp$
13. (Else) $(\widetilde{E}, \widetilde{e}, x, y) = \mathsf{Tamper}^{(E,e)}_{\mathcal{L}^L, \mathcal{L}^R, f_0}$
14. If $(\widetilde{E}, \widetilde{e}) = \perp$, output $\perp$
15. $(c^L, c^R) \leftarrow \mathrm{NMEnc}_0(E, e)$ s.t. $\mathrm{NMDec}_0(g(c^L, c^R)) = (\widetilde{E}, \widetilde{e})$ and $c^L_{\mathcal{L}^L} = x$, $c^R_{\mathcal{L}^R} = y$
16. $\alpha^L = \mathrm{Embed}^L(s^L, c^L)$, $\alpha^R = \mathrm{Embed}^R(s^R, c^R)$
17. $\widetilde{c} = f_1(c, s^L, \alpha^L, s^R, \alpha^R)$
18. If $\mathrm{ECSS.ECorr}(\widetilde{c}) = \perp$, output $\perp$; (Else) $\widetilde{a} = \mathrm{ECSS.ECorr}(\widetilde{c})$

19. Define $c'$ as $c'_i = \begin{cases} \widetilde{a}_i, & i \notin \widetilde{E} \\ \widetilde{e}_i, & i \in \widetilde{E} \end{cases}$
20. If $c' \neq \widetilde{c}$, output $\perp$; (Else) $\widetilde{m} = \mathrm{ECSS.Dec}(\widetilde{a})$
21. Output $\widetilde{m}$

$H_4(f, m)$ : In the next hybrid $H_4$, we simply replace the tampering experiment onto our base non-malleable codes with its corresponding simulator $\mathrm{Sim}_0$ and incur a negligible error by Lemma 3.4.1.

$H_4(f, m)$:

1. Let $P = \{i | i \in [n], |\mathrm{Out}_f(i) \cap [n]| \geqslant n^{\varepsilon_2}\}$
2. Let $Q = \{i | i \in [n], \mathrm{Out}_f(i) \backslash [n] \neq \emptyset\}$
3. Let $X = P \cup Q$. Sample $a_X \leftarrow U_{\{0,1\}^{|X|}}$
4. Sample $a \leftarrow \mathrm{ECSS.Enc}(m) | (\mathrm{ECSS.Enc}(m))_X = a_X$
5. Sample a random $E \subseteq [n]$ of size $n^{1-\varepsilon_1}$
6. For all $i \in E$, sample $e_i \leftarrow U_{\{0,1\}}$
7. Define $c$ as $c_i = \begin{cases} a_i, & i \notin E \\ e_i, & i \in E \end{cases}$
8. Sample seeds $s^L, s^R$ uniformly from $\{0,1\}^{3\Lambda \log^3 n}$
9. Given $s^L$, define: $\mathrm{Bad}^L, \mathrm{Leak}^L$ as in $H_3(f, m)$
   Given $s^R$, define: $\mathrm{Bad}^R, \mathrm{Leak}^R$ as in $H_3(f, m)$
10. Let $\mathcal{L}^L = \mathrm{Bad}^L \cup \mathrm{Leak}^L$ and $\mathcal{L}^R = \mathrm{Bad}^R \cup \mathrm{Leak}^R$.
11. Define mapping $f_0$ and its output $g$ as in $H_3(f, m)$
12. If $(|\mathcal{L}^L| \geqslant \lambda n^{\beta_1})$ or $(|\mathcal{L}^R| \geqslant \lambda n^{\beta_2})$ or ($f_0$ does not satisfy Definition 3.4.1), output $\perp$
13. (Else) $(\mathrm{ans}, x, y) = \mathrm{Sim}_0(\mathcal{L}^L, \mathcal{L}^R, f_0)$
14. If $\mathrm{ans} = \perp$, output $\perp$; (Else) $(\widetilde{E}, \widetilde{e}) = \mathsf{Copy}(\mathrm{ans}, (E, e))$
15. $(c^L, c^R) \leftarrow \mathrm{NMEnc}_0(E, e)$ s.t. $\mathrm{NMDec}_0(g(c^L, c^R)) = (\widetilde{E}, \widetilde{e})$ and $c^L_{\mathcal{L}^L} = x$, $c^R_{\mathcal{L}^R} = y$
16. $\alpha^L = \mathrm{Embed}^L(s^L, c^L)$, $\alpha^R = \mathrm{Embed}^R(s^R, c^R)$
17. $\widetilde{c} = f_1(c, s^L, \alpha^L, s^R, \alpha^R)$
18. If $\mathrm{ECSS.ECorr}(\widetilde{c}) = \perp$, output $\perp$; (Else) $\widetilde{a} = \mathrm{ECSS.ECorr}(\widetilde{c})$
19. Define $c'$ as $c'_i = \begin{cases} \widetilde{a}_i, & i \notin \widetilde{E} \\ \widetilde{e}_i, & i \in \widetilde{E} \end{cases}$

20. If $c' \neq \widetilde{c}$, output $\perp$; (Else) $\widetilde{m} = \text{ECSS.Dec}(\widetilde{a})$
21. Output $\widetilde{m}$

$H_5(f, m)$ : In this hybrid, we break the error indices $E$ into two parts: $E_1 = E \cap X$ and $E_2 = E \backslash E_1$. Next, we note that $c_Q$ is needed to define the tampering on the error codeword. Hence, we sample $E_1$ and the error bits from $E_1$ early before defining tampering on error codeword. However, rest of errors, i.e., $E_2$ is not used before we invoke simulator $\text{Sim}_0$. Based on these observations, we re-arrange parts of the hybrids and this hybrid is identical to the previous one. Note that the size of $E_1$ and $E_2$ are distributed according to $(n, |X|, n^{1-\varepsilon_1})$-hypergemetric distribution and $(n, n-|X|, n^{1-\varepsilon_1})$-hyper geometric distribution, respectively. By Corollary 3.3.1, it is easy to see that with probability $1 - \exp(-\Theta(n^{1-\varepsilon_1}))$, the size of $E_2$ is at least $n^{1-\varepsilon_1}/2$.

$H_5(f, m)$:

1. Let $P = \{i | i \in [n], |\text{Out}_f(i) \cap [n]| \geqslant n^{\varepsilon_2}\}$
2. Let $Q = \{i | i \in [n], \text{Out}_f(i) \backslash [n] \neq \emptyset\}$
3. Let $X = P \cup Q$. Sample $a_X \leftarrow U_{\{0,1\}^{|X|}}$
4. Sample a random $E_1 \subseteq X$ s.t. $|E_1| \leftarrow (n, |X|, n^{1-\varepsilon_1})$-hypergeometric distribution
5. For all $i \in E_1$, sample $e_i \leftarrow U_{\{0,1\}}$
6. For all $i \in E_1$, replace $a_i$ with $e_i$, we get $c_X$
7. Sample seeds $s^L, s^R$ uniformly from $\{0,1\}^{3\Lambda \log^3 n}$
8. Given $s^L$, define: $\text{Bad}^L, \text{Leak}^L$ as in $H_3(f, m)$
   Given $s^R$, define: $\text{Bad}^R, \text{Leak}^R$ as in $H_3(f, m)$
9. Let $\mathcal{L}^L = \text{Bad}^L \cup \text{Leak}^L$ and $\mathcal{L}^R = \text{Bad}^R \cup \text{Leak}^R$.
10. Define mapping $f_0$ and its output $g$ as in $H_3(f, m)$
11. If $(|\mathcal{L}^L| \geqslant \lambda n^{\beta_1})$ or $(|\mathcal{L}^R| \geqslant \lambda n^{\beta_2})$ or ($f_0$ does not satisfy Definition 3.4.1), output $\perp$
12. (Else) $(\text{ans}, x, y) = \text{Sim}_0(\mathcal{L}^L, \mathcal{L}^R, f_0)$.
13. If $\text{ans} = \perp$, output $\perp$
14. Sample $a \leftarrow \text{ECSS.Enc}(m) | (\text{ECSS.Enc}(m))_X = a_X$
15. Sample a random $E_2 \subseteq [n] \backslash X$ of size $n^{1-\varepsilon_1} - |E_1|$, Let $E = E_1 \cup E_2$
16. For all $i \in E_2$, sample $e_i \leftarrow U_{\{0,1\}}$
17. Define $c$ as $c_i = \begin{cases} a_i, & i \notin E \\ e_i, & i \in E \end{cases}$

18. $(\widetilde{E}, \widetilde{e}) = \mathsf{Copy}(\text{ans}, (E, e))$
19. $(c^L, c^R) \leftarrow \text{NMEnc}_0(E, e)$ s.t. $\text{NMDec}_0(g(c^L, c^R)) = (\widetilde{E}, \widetilde{e})$ and $c^L_{\mathcal{L}^L} = x$, $c^R_{\mathcal{L}^R} = y$
20. $\alpha^L = \text{Embed}^L(s^L, c^L)$, $\alpha^R = \text{Embed}^R(s^R, c^R)$
21. $\widetilde{c} = f_1(c, s^L, \alpha^L, s^R, \alpha^R)$
22. If $\text{ECSS.ECorr}(\widetilde{c}) = \perp$, output $\perp$; (Else) $\widetilde{a} = \text{ECSS.ECorr}(\widetilde{c})$
23. Define $c'$ as $c'_i = \begin{cases} \widetilde{a}_i, & i \notin \widetilde{E} \\ \widetilde{e}_i, & i \in \widetilde{E} \end{cases}$
24. If $c' \neq \widetilde{c}$, output $\perp$; (Else) $\widetilde{m} = \text{ECSS.Dec}(\widetilde{a})$
25. Output $\widetilde{m}$

$H_6(f, m)$ : In this hybrid, we only introduce some new notation to be used in later hybrids and hence, this hybrid is identical to the previous one.

We focus on the tampering of the main codeword using function $f_1$. Note that so far in the previous hybrid, we have already fixed certain bits in the input main codeword $c$ (that is, $c_X$), picked PRG seeds $s^L, s^R$ and also leaked certain parts of $c^L, c^R$, i.e., $\mathcal{L}^L, \mathcal{L}^R$.[9] Using this information, we define a restriction $h$ of function $f_1$ that fixes all the above bits in the input.

We next define three subsets of $[n]$ corresponding to $h$, namely, $V$, $W$ and $Z$ as follows. $V$ is the subset of bits $i$ such that $\widetilde{c}_i$ is not fixed given the fixing of bits done so far. And $W$ is the subset of bits that are influenced by some bits in the error codeword (that have not been leaked and fixed so far). And $Z$ is the subset of bits $i$, such that the output of $h_i$ is not always the $i$-th input bit (In the definition of $Z$, recall that $n_e = |(s^L, \alpha^L, s^R, \alpha^R)|$).

Intuitively, $Z$ is the set of bits that are not-copied-bits under the tampering function $h$, $V$ is the set of non-constant-bits and $W$ is the set of bits that are influenced by the error codeword. As we explained in technical overview Section 3.2, if $\text{ans} = \mathsf{same}^*$ and the size of $Z \backslash W$ is large or if $\text{ans} = (E^*, e^*)$ and the size of $V \backslash W$ is large, then the tampered codeword will be invalid with probability $1 - \mathsf{negl}(\lambda)(n)$. This intuition is formally proved in the next hybrid.

[9] ↑Note that those places in $\alpha^L, \alpha^R$ that are not used to store $c^L$ and $c^R$ are also fixed (to be 0 by the compiler).

$H_6(f, m)$:

1. Let $P = \{i | i \in [n], |\mathrm{Out}_f(i) \cap [n]| \geqslant n^{\varepsilon_2}\}$
2. Let $Q = \{i | i \in [n], \mathrm{Out}_f(i) \backslash [n] \neq \emptyset\}$
3. Let $X = P \cup Q$. Sample $a_X \leftarrow U_{\{0,1\}^{|X|}}$
4. Sample a random $E_1 \subseteq X$ s.t. $|E_1| \leftarrow (n, |X|, n^{1-\varepsilon_1})$-hypergeometric distribution.
5. For all $i \in E_1$, sample $e_i \leftarrow U_{\{0,1\}}$
6. For all $i \in E_1$, replace $a_i$ with $e_i$, we get $c_X$
7. Sample seeds $s^L, s^R$ uniformly from $\{0,1\}^{3\Lambda \log^3 n}$
8. Given $s^L$, define: $\mathrm{Bad}^L, \mathrm{Leak}^L$ as in $H_3(f, m)$
   Given $s^R$, define: $\mathrm{Bad}^R, \mathrm{Leak}^R$ as in $H_3(f, m)$
9. Let $\mathcal{L}^L = \mathrm{Bad}^L \cup \mathrm{Leak}^L$ and $\mathcal{L}^R = \mathrm{Bad}^R \cup \mathrm{Leak}^R$.
10. Define mapping $f_0$ and its output $g$ as in $H_3(f, m)$
11. If $(|\mathcal{L}^L| \geqslant \lambda n^{\beta_1})$ or $(|\mathcal{L}^R| \geqslant \lambda n^{\beta_2})$ or ($f_0$ does not satisfy Definition 3.4.1), output $\perp$
12. (Else) $(\mathrm{ans}, x, y) = \mathrm{Sim}_0(\mathcal{L}^L, \mathcal{L}^R, f_0)$.
13. Let $S^L, S^R$ denote indices of $s^L, s^R$. Define function $h$ as a restriction of $f_1$ (Definition 3.3.2): $h := (f_1)_{(X, S^L, S^R, \mathcal{L}^L, \mathcal{L}^R)|(c_X, s^L, s^R, x, y)}$
14. $V := \{i \in [n] | \mathrm{Inp}_h(i) \neq \emptyset\}$.
15. $W := \{i \in [n] | \mathrm{Inp}_h(i) \backslash [n] \neq \emptyset\}$.
16. $Z := \{i \in [n] | \exists z \in \{0,1\}^{n+n_e}, z_{(X, S^L, S^R, \mathcal{L}^L, \mathcal{L}^R)} = (c_X, s^L, s^R, x, y), h_i(z) \neq z_i\}$.
17. If ans $= \perp$, output $\perp$
18. Sample $a \leftarrow \mathrm{ECSS.Enc}(m) | (\mathrm{ECSS.Enc}(m))_X = c_X$
19. Sample a random $E_2 \subseteq [n] \backslash X$ of size $n^{1-\varepsilon_1} - |E_1|$, let $E = E_1 \cup E_2$
20. For all $i \in E_2$, sample $e_i \leftarrow U_{\{0,1\}}$
21. Define $c$ as $c_i = \begin{cases} a_i, & i \notin E \\ e_i, & i \in E \end{cases}$
22. $(\widetilde{E}, \widetilde{e}) = \mathsf{Copy}(\mathrm{Sim}_0(\mathcal{L}^L, \mathcal{L}^R, f_0), (E, e))$
23. $(c^L, c^R) \leftarrow \mathrm{NMEnc}_0(E, e)$ s.t. $\mathrm{NMDec}_0(g(c^L, c^R)) = (\widetilde{E}, \widetilde{e})$ and $c^L_{\mathcal{L}^L} = x$, $c^R_{\mathcal{L}^R} = y$
24. $\alpha^L = \mathrm{Embed}^L(s^L, c^L)$, $\alpha^R = \mathrm{Embed}^R(s^R, c^R)$
25. $\widetilde{c} = f_1(c, s^L, \alpha^L, s^R, \alpha^R)$
26. If $\mathrm{ECSS.ECorr}(\widetilde{c}) = \perp$, output $\perp$; (Else) $\widetilde{a} = \mathrm{ECSS.ECorr}(\widetilde{c})$
27. Define $c'$ as $c'_i = \begin{cases} \widetilde{a}_i, & i \notin \widetilde{E} \\ \widetilde{e}_i, & i \in \widetilde{E} \end{cases}$
28. If $c' \neq \widetilde{c}$, output $\perp$; (Else) $\widetilde{m} = \mathrm{ECSS.Dec}(\widetilde{a})$
29. Output $\widetilde{m}$

$H_7(f, m)$ :

In the next hybrid $H_7$, we add a sanity check right after we define $V, W, Z$. (a) When ans $= \perp$, we will output $\perp$ immediately. This is the same as the previous hybrid. (b) When ans $= \mathsf{same}^*$, we check the size of $Z \backslash (W \cup X)$. If it is larger than $n^{1-\varepsilon_2}$, we directly output $\perp$ without any further computation. On the other hand, if it is less than $n^{1-\varepsilon_2}$, we only compare $c$ and $\widetilde{c}$ at locations $Z$. If they are the same, we output $\mathsf{same}^*$, otherwise, we output $\perp$. (c) When ans $= (E^*, e^*)$, we check the size of $V \backslash W$. If $|V \backslash W| \geqslant n^{1-\varepsilon_2}$, we directly output $\perp$ without further computation. Below, we prove that the previous hybrid $H_6(f, m)$ and $\mathsf{Copy}(H_7(f, m), m)$ are statistically close. We break the proof into two parts: ans $= \mathsf{same}^*$ case and ans $= (E^*, e^*)$ case.

**Case** ans $= \mathsf{same}^*$**:** Let us first look at that the case when $|Z \backslash (W \cup X)| < n^{1-\varepsilon_2}$. Note that by the definition of $Z$, all the bits of $\widetilde{c}$ in $[n] \backslash Z$ are identical to those in $c$. Recall $c$ is obtained by planting $|E| = n^{1-\varepsilon_1}$ errors into a valid ECSS codeword $a$. We have $\mathsf{HD}(\widetilde{c}, a) \leqslant \mathsf{HD}(\widetilde{c}, c) + \mathsf{HD}(c, a) = |Z| + |E| \leqslant (|Z \backslash (W \cup X)| + |W| + |X|) + |E|$. Using $|W| = o(n^{1-\tau})$ from (3) of Lemma 3.6.1, $|X| = o(n^{1-\zeta})$ from hybrid 2, and $|E| = n^{1-\varepsilon_1}$, we get $\mathsf{HD}(\widetilde{c}, a) \leqslant n^{1-\varepsilon_2} + o(n^{1-\tau}) + o(n^{1-\zeta}) + n^{1-\varepsilon_1} = o(n^{1-\zeta})$ by setting $\boxed{\zeta < \varepsilon_2}$, $\boxed{\zeta < \tau}$ and $\boxed{\zeta < \varepsilon_1}$. Hence, using the fact that the distance of the ECSS scheme, $d \geqslant n^{1-\zeta}$, we get $\mathsf{ECSS.ECorr}(\widetilde{c}) = a$. Consequently, if we error-correct $\widetilde{c}$ and plant in the original errors $(E, e)$, we get $c$. Hence, experiment would output $\perp$ iff $\widetilde{c} \neq c$. This happens only when $\widetilde{c}_Z \neq c_Z$.

Now consider the case when $|Z \backslash (W \cup X)| \geqslant n^{1-\varepsilon_2}$. We begin by computing a lower bound on number of error indices in $Z \backslash (W \cup X)$, i.e., size of set $A = (Z \backslash (W \cup X)) \cap E_2$. First, note that $E_2$ is a random subset of $[n] \backslash X$ of size at least $n^{1-\varepsilon_1}/2$ with probability $1 - \exp(-\Omega(n^{1-\varepsilon_1}))$ by Corollary 3.3.1. Next, we observe that sets $Z, W, X$ are defined independent of $E_2$ and hence, by Corollary 3.3.1, $|A| \geqslant \frac{1}{4} \cdot n^{1-\varepsilon_1-\varepsilon_2}$ with probability at least $1 - \exp(-\Omega(n^{1-\varepsilon_1-2\varepsilon_2}))$.

Next, we pick a subset $A' \subseteq A$ such that bits in $A'$ have disjoint input neighbors. That is, $\forall i, j \in A', \mathrm{Inp}_h(i) \cap \mathrm{Inp}_h(j) = \emptyset$. We use following two properties to ensure that we can pick $A'$ of sufficiently large size. First, for every bit $i \in A$, $\mathrm{Inp}_h(i) \subseteq [n]$ (because $A \cap W = \emptyset$). Second, all the bits in $[n]$ with more than $n^{\varepsilon_2}$ output neighbors in $[n]$ belong to subset $P$ and have already been fixed. This implies that for any bit $i \in A$, all bits in $\mathrm{Inp}_h(i)$ have at most $n^{\varepsilon_2}$ output neighbours in $[n]$. Therefore, it is guaranteed that we can pick a set $A' \subseteq A$ s.t. $|A'| \geqslant \frac{|A|}{\delta n^{\varepsilon_2}} = \frac{n^{1-\varepsilon_1-2\varepsilon_2}}{4\delta}$. (This can be done greedily by picking an arbitrary index $i \in A$ and discarding all the bits in $A$ that are influenced by $\mathrm{Inp}_h(i)$. Since $h$ has at output locality $\delta$ and each bit in $\mathrm{Inp}_h(i)$ influences at most $n^{\varepsilon_2}$-many bits in $A$, we discard at most $\delta n^{\varepsilon_2}$ indices from $A$ for picking one index in $A'$. Now, we recurse on the remaining indices in $A$.)

For the rest of the proof, we consider such a set $A'$ of size exactly $\frac{n^{1-\varepsilon_1-2\varepsilon_2}}{4\delta}$. We note that for all indices $i \in A'$ following conditions are satisfied (1) $c_i$ is a planted error $e_i$ ($A' \subseteq E_2$); (2) $h_i$ does not always output $e_i$ ($A' \subseteq Z$); (3) the input neighbors of $i$ are all in $[n]$ ($A' \cap W = \emptyset$). For the tampered main codeword to be consistent with recorded errors, we need that for all $i \in A'$, The $i$-th bit after tampering, i.e. $\widetilde{c_i}$ needs to be equal to $e_i$. We show that this happens with probability at most $(1 - 1/2^{\delta})^{\frac{n^{1-\varepsilon_1-2\varepsilon_2}}{8\delta}}$, which is negligible for $\delta = \xi \cdot \lg n$ when $\boxed{\varepsilon_1 + 2\varepsilon_2 < 1 - \xi}$. Hence, it suffices to output $\perp$ always.

We first argue that all of $A'$ input neighbors are independent uniform bits. We use the fact that $A'$ is of size $\frac{n^{1-\varepsilon_1-2\varepsilon_2}}{4\delta}$ and its (at most $\delta \cdot \frac{n^{1-\varepsilon_1-2\varepsilon_2}}{4\delta}$-many) input neighbors are all from our ECSS codeword with planted errors. Since we have only fixed $X$ of size $o(t)$ from $c$ so far and our ECSS has independence $t \geqslant n^{1-\zeta}$ and $n^{1-\varepsilon_1-2\varepsilon_2} = o(t)$, all the input neighbors of $A'$ are indeed independent uniform bits. Given the uniformly random input, we examine the bits from $A'$ one by one. For any $i \in A'$, there are following two possibilities

- If $c_i$ (i.e., $e_i$) is the input neighbor of $\widetilde{c_i}$, then since $h_i$ does not always output $c_i$, there exists a setting of the other (at most) $\delta - 1$ neighbors, such that $\widetilde{c_i}$ is either fixed 0, fixed 1, or flipped $e_i$. Because of uniformity of value at input neighbors, this setting happens with probability at least $\frac{1}{2^{\delta-1}}$ and when it happens, with probability at least $1/2$, $\widetilde{c_i} \neq e_i$. Hence, $\widetilde{c_i} = e_i$ with probability at most $1 - \frac{1}{2^{\delta}}$. We remove $i$ from $A'$ and recurse on remaining bits.

- If $c_i$ (i.e., $e_i$) is not the input neighbor of $\widetilde{c_i}$, then since all of the input neighbors are independent of uniform bit $e_i$, the probability $\widetilde{c_i} = e_i$ is at most $1/2$. However, we need to address a small subtlety here. Since $e_i$ is not the input neighbor of itself, it can be in the input neighbor of another bit in $A'$. To keep failure probabilities independent, if such a bit $j$ exists (s.t. $e_i$ is an input neighbor of $\widetilde{c_j}$), we only include $i$ in our witness set of failed indices but we remove both indices $i$ and $j$ before recursing to remaining bits in $A'$.

Now, we have shown that either a bit has probability at most $1 - \frac{1}{2^\delta}$ to be consistent or two bits have probability at most $1/2$ to be consistent at the same time. And all of those events are independent, hence, the probability that all the bits are consistent with errors $(E, e)$ is at most $(1 - \frac{1}{2^\delta})^{|A'|/2}$.

**Case** ans $= (E^*, e^*)$**:** For the case when ans $= (E^*, e^*)$, this hybrid is only different from previous one when $|V \backslash W| \geqslant n^{1-\varepsilon_2}$. We show that if this happens, the output of previous hybrid is not $\perp$ with only negligible probability.

We first pick a $B \subseteq (V \backslash W)$ such that $\forall i, j \in B, \mathrm{Inp}_h(i) \cap \mathrm{Inp}_h(j) = \emptyset$. Similar to above, all the input neighbors of $V \backslash W$ are contained in $[n]$ and have output locality at most $n^{\varepsilon_2}$ in $[n]$. Hence, it is guaranteed that we could pick $B$ such that $|B| = \frac{n^{1-2\varepsilon_2}}{\delta}$. (Similar to $\mathsf{same}^*$ case, this can be done greedily by picking an arbitrary index from $V \backslash W$ into $B$ and removing all the bits its input neighbors have influence on. We only discard at most $\delta n^{\varepsilon_2}$ bits for picking one bit.)

Note that $B \subseteq V$ implies that for all $i \in B$, $\mathrm{Inp}_h(i) \neq \emptyset$ and since all the bits in $B$ has disjoint input neighbors, we have $|\mathrm{Inp}_h(B)| \geqslant |B|$. Now, consider a subset $B' \subseteq B$ such that each bit in $B'$ has an input neighbour in errors $E_2$. That is,

$$B' = \left\{ i \middle| i \in B, \mathrm{Inp}_h(i) \cap E_2 \neq \emptyset \right\}$$

Again $E_2$ is a random subset of size at least $n^{1-\varepsilon_1}/2$ with probability $1 - \exp(-\Omega(n^{1-\varepsilon_1}))$ and is independent of $B$. Thus, by Corollary 3.3.1, with probability at least $1 - \exp(-\Omega(n^{1-\varepsilon_1-4\varepsilon_2}))$, $|\mathrm{Inp}_h(B) \cap E_2| \geqslant \frac{1}{4\delta} n^{1-\varepsilon_1-2\varepsilon_2}$. Hence, $|B'| \geqslant \frac{n^{1-\varepsilon_1-2\varepsilon_2}}{4\delta^2}$ (Because of $\delta$-locality).

For the rest of proof, we consider such a set $B'$ of size exactly $\frac{n^{1-\varepsilon_1-2\varepsilon_2}}{4\delta^2}$. Next, we argue that input neighbors of $B'$ (at most $\delta \cdot n^{1-\varepsilon_1-2\varepsilon_2}/(4\delta^2)$ in number) are independently uniformly distributed. This is because they are all from our ECSS codeword with planted errors. Since we have only fixed $X$ of size $o(t)$ from $c$ so far and our ECSS has independence $t = n^{1-\zeta}$ with $\boxed{\zeta < \varepsilon_1 + 2\varepsilon_2}$, all the input neighbors of $B'$ are indeed independent uniform bits. So, bits in $B'$ satisfy the following conditions: its input neighbors (1) are disjoint; (2) contain at least one bit from $E_2$; (3) are contained in $[n]$; (4) are independently uniform bits.

Next, we define $M = \mathrm{Out}_h(\mathrm{Inp}_h(B'))$. This is the set of all indices that is being influenced by the input neighbor of $B'$. Obviously $B' \subseteq M$. And the size of $M$ is bounded by $n^{\varepsilon_2} \cdot \delta \cdot n^{1-\varepsilon_1-2\varepsilon_2}/(4\delta^2) = n^{1-\varepsilon_1-\varepsilon_2}/(4\delta)$. We first observe that fix any $c^*_{[n]\setminus M}$, there is at most one $c^*_M$ that is consistent with $c^*_{[n]\setminus M}$ and the fixed errors $E^*, e^*$. This is because if there exist two $c^{(1)}$, $c^{(2)}$ s.t. $c^{(1)}_{[n]\setminus M} = c^{(2)}_{[n]\setminus M}$, their distance is bounded by $n^{1-\varepsilon_1-\varepsilon_2}/(4\delta)$ which is smaller than the distance $d \geqslant n^{1-\zeta}$ as long as $\boxed{\zeta < \varepsilon_1 + \varepsilon_2}$. Therefore, those two codewords will be error-corrected to the same correct codeword and after being reconstructed from errors $(E^*, e^*)$, they will be the same. Therefore, for every fixing $c^*_{[n]\setminus M}$, there is at most one codeword $c^*$ (equivalently, one $c^*_M$), which is consistent with $(E^*, e^*)$. Since $B' \subseteq M$, there is at most one choice for $c^*_{B'}$ as well.

Finally, we prove that the probability that $c^*_{B'}$ takes the fixed value needed to be consistent is negligible. Now,for any $i \in B'$, we know some bit $E_j$ is the input neighbors of $i$. Therefore, at least one out of at most $2^{\delta-1}$ possible settings of all the other input neighbprs $\mathrm{Inp}_h(i)\setminus[j]$, flipping the value of $e_j$ will flip the output of $h_i$. Note that by definition of $M$, $E_j$ cannot be the input neighbors of any bits in $[n]\setminus M$, hence $e_j$ is independent of $c^*_{[n]\setminus M}$. And thus, whenever this setting happens, with probability $1/2$, the output at $i$ will not be consistent with $(E^*, e^*)$. Therefore, since the input neighbors of $i$ are uniformly distributed, the probability that $\widetilde{c_i}$ is not consistent with fixed errors $(E^*, e^*)$ is at least $\frac{1}{2^\delta}$. Since all the input neighbors of $B'$ are all independent uniform bits, the probability that all the bits from $B'$ are consistent is at most $(1 - \frac{1}{2^\delta})^{n^{1-\varepsilon_1-2\varepsilon_2}/(4\delta^2)}$, which is negligible when $\delta = \xi \cdot \lg n$ with $\boxed{\varepsilon_1 + 2\varepsilon_2 < 1 - \xi}$.

$H_7(f, m)$:

1. Let $P = \{i | i \in [n], |\mathrm{Out}_f(i) \cap [n]| \geqslant n^{\varepsilon_2}\}$
2. Let $Q = \{i | i \in [n], \mathrm{Out}_f(i) \backslash [n] \neq \emptyset\}$
3. Let $X = P \cup Q$. Sample $a_X \leftarrow U_{\{0,1\}^{|X|}}$
4. Sample a random $E_1 \subseteq X$ s.t. $|E_1| \leftarrow (n, |X|, n^{1-\varepsilon_1})$-hypergeometric distribution
5. For all $i \in E_1$, sample $e_i \leftarrow U_{\{0,1\}}$
6. For all $i \in E_1$, replace $a_i$ with $e_i$, we get $c_X$
7. Sample seeds $s^L, s^R$ uniformly from $\{0,1\}^{3\Lambda \log^3 n}$
8. Given $s^L$, define: $\mathrm{Bad}^L, \mathrm{Leak}^L$ as in $H_3(f, m)$
   Given $s^R$, define: $\mathrm{Bad}^R, \mathrm{Leak}^R$ as in $H_3(f, m)$
9. Let $\mathcal{L}^L = \mathrm{Bad}^L \cup \mathrm{Leak}^L$ and $\mathcal{L}^R = \mathrm{Bad}^R \cup \mathrm{Leak}^R$.
10. Define mapping $f_0$ and its output $g$ as in $H_3(f, m)$
11. If $(|\mathcal{L}^L| \geqslant \lambda n^{\beta_1})$ or $(|\mathcal{L}^R| \geqslant \lambda n^{\beta_2})$ or ($f_0$ does not satisfy Definition 3.4.1), output $\perp$
12. (Else) $(\mathrm{ans}, x, y) = \mathrm{Sim}_0(\mathcal{L}^L, \mathcal{L}^R, f_0)$.
13. Let $S^L, S^R$ denote indices of $s^L, s^R$. Then, $h := (f_1)_{(X, S^L, S^R, \mathcal{L}^L, \mathcal{L}^R)|(c_X, s^L, s^R, x, y)}$
14. $V := \{i | i \in [n], \mathrm{Inp}_h(i) \neq \emptyset\}$
15. $W := \{i | i \in [n], \mathrm{Inp}_h(i) \backslash [n] \neq \emptyset\}$
16. $Z := \{i \in [n] | \exists z \in \{0,1\}^{n+n_e}, z_{(X, S^L, S^R, \mathcal{L}^L, \mathcal{L}^R)} = (c_X, s^L, s^R, x, y), h_i(\mathrm{z}) \neq z_i\}$
17. If $\mathrm{ans} = \perp$, output $\perp$
    If $\mathrm{ans} = \mathsf{same}^*$ and $|Z \backslash (W \cup X)| \geqslant n^{1-\varepsilon_2}$, output $\perp$
    If $\mathrm{ans} = (E^*, e^*)$ and $|V \backslash W| \geqslant n^{1-\varepsilon_2}$, output $\perp$
18. Sample $a \leftarrow$ ECSS.Enc(m) $|(\mathrm{ECSS.\,Enc}(m))_X = c_X$
19. Sample a random $E_2 \subseteq [n] \backslash X$ of size $n^{1-\varepsilon_1} - |E_1|$, let $E = E_1 \cup E_2$
20. For all $i \in E_2$, sample $e_i \leftarrow U_{\{0,1\}}$
21. Define $c$ as $c_i = \begin{cases} a_i, & i \notin E \\ e_i, & i \in E \end{cases}$
22. $(\widetilde{E}, \widetilde{e}) = \mathsf{Copy}(\mathrm{Sim}_0(\mathcal{L}^L, \mathcal{L}^R, f_0), (E, e))$
23. $(c^L, c^R) \leftarrow \mathrm{NMEnc}_0(E, e)$ s.t. $\mathrm{NMDec}_0(g(c^L, c^R)) = (\widetilde{E}, \widetilde{e})$ and $c^L_{\mathcal{L}^L} = x$, $c^R_{\mathcal{L}^R} = y$
24. $\alpha^L = \mathrm{Embed}^L(s^L, c^L)$, $\alpha^R = \mathrm{Embed}^R(s^R, c^R)$
25. $\widetilde{c} = f_1(c, s^L, \alpha^L, s^R, \alpha^R)$
26. If ans =

- $\mathsf{same}^*$: If $\widetilde{c}_Z = c_Z$, output $\mathsf{same}^*$
  (Else) Output $\perp$.

- $(E^*, e^*)$: If $\mathrm{ECSS.ECorr}(\widetilde{c}) = \perp$, output $\perp$; (Else) $\widetilde{a} = \mathrm{ECSS.ECorr}(\widetilde{c})$

  Define $c'$ as $c'_i = \begin{cases} \widetilde{a}_i, & i \notin \widetilde{E} \\ \widetilde{e}_i, & i \in \widetilde{E} \end{cases}$

  If $c' \neq \widetilde{c}$, output $\perp$; (Else) $\widetilde{m} = \mathrm{ECSS.Dec}(\widetilde{a})$

  Output $\widetilde{m}$

$H_8(f, m)$ : In the final hybrid, we simply switch message $m$ with $0^\ell$. For completeness, we provide this hybrid in supplementary material.

Note that the only bits from $\mathrm{ECSS.Enc}(m)$ that affect the output of the hybrid is (1) the neighbors of $Z$ and also $c_Z$ when ans $= \mathsf{same}^*$ and $|Z \backslash (W \cup X)| < n^{1-\varepsilon_2}$; (2) the neighbors of $V$ , when ans $\notin \{\mathsf{same}^*, \perp\}$ and $|V \backslash W| < n^{1-\varepsilon_2}$.[10] For (1), as shown in hybrid 7, the size of $Z$ is $\mathrm{o}(t)$ when $|Z \backslash (W \cup X)| < n^{1-\varepsilon_2}$ and hence the neighbor of $|Z|$ is of size at most $\delta \cdot |Z| = \mathrm{o}(t)$. For (2), $|V| \leqslant |V \backslash W| + |W|$. Both are $\mathrm{o}(t)$ as require in hybrid 7 and therefore so is $|V|$ and the size of the neighbors of $V$. Hence the number of bits in $c$ that influence the hybrid output is at most $\mathrm{o}(t)$. Any $\mathrm{o}(t)$ bits from $\mathrm{ECSS.Enc}(m)$ condition on $c_X$ is uniformly distributed. Hence, we can switch the encoding of $m$ with encoding of $0^\ell$. And this final hybrid is identical to our simulator in Figure 3.3.

$H_8(f)$:

1. Let $P = \{i | i \in [n], |\mathrm{Out}_f(i) \cap [n]| \geqslant n^{\varepsilon_2}\}$
2. Let $Q = \{i | i \in [n], \mathrm{Out}_f(i) \backslash [n] \neq \emptyset\}$
3. Let $X = P \cup Q$. Sample $a_X \leftarrow U_{\{0,1\}^{|X|}}$
4. Sample a random $E_1 \subseteq X$ s.t. $|E_1| \leftarrow (n, |X|, n^{1-\varepsilon_1})$-hypergeometric distribution
5. For all $i \in E_1$, sample $e_i \leftarrow U_{\{0,1\}}$
6. For all $i \in E_1$, replace $a_i$ with $e_i$, we get $c_X$
7. Sample seeds $s^L, s^R$ uniformly from $\{0,1\}^{3\Lambda \log^3 n}$
8. Given $s^L$, define: $\mathrm{Bad}^L, \mathrm{Leak}^L$ as in $H_3(f, m)$

   Given $s^R$, define: $\mathrm{Bad}^R, \mathrm{Leak}^R$ as in $H_3(f, m)$
9. Let $\mathcal{L}^L = \mathrm{Bad}^L \cup \mathrm{Leak}^L$ and $\mathcal{L}^R = \mathrm{Bad}^R \cup \mathrm{Leak}^R$.

[10] ↑Note that, by the definition of $V$, all the output bits from $[n] \backslash V$ are fixed to some values with no input neighbors. Hence, it suffices to have the neighbor of $V$ to finish the hybrid completely.

10. Define mapping $f_0$ and its output $g$ as in $H_3(f, m)$
11. If $(|\mathcal{L}^L| \geqslant \lambda n^{\beta_1})$ or $(|\mathcal{L}^R| \geqslant \lambda n^{\beta_2})$ or ($f_0$ does not satisfy Definition 3.4.1), output $\perp$
12. (Else) $(\text{ans}, x, y) = \text{Sim}_0(\mathcal{L}^L, \mathcal{L}^R, f_0)$.
13. Let $S^L, S^R$ denote indices of $s^L, s^R$. Then, $h := (f_1)_{(X,S^L,S^R,\mathcal{L}^L,\mathcal{L}^R)|(c_X,s^L,s^R,x,y)}$
14. $V := \{i | i \in [n], \text{Inp}_h(i) \neq \emptyset\}$
15. $W := \{i | i \in [n], \text{Inp}_h(i) \backslash [n] \neq \emptyset\}$
16. $Z := \{i \in [n] | \exists z \in \{0,1\}^{n+n_e}, z_{(X,S^L,S^R,\mathcal{L}^L,\mathcal{L}^R)} = (c_X, s^L, s^R, x, y), h_i(\text{z}) \neq z_i\}$
17. If ans $= \perp$, output $\perp$

    If ans $= \mathsf{same}^*$ and $|Z \backslash (W \cup X)| \geqslant n^{1-\varepsilon_2}$, output $\perp$

    If ans $= (E^*, e^*)$ and $|V \backslash W| \geqslant n^{1-\varepsilon_2}$, output $\perp$
18. Sample $a \leftarrow \text{ECSS.Enc}(0^\ell) | (\text{ECSS.Enc}(0^\ell))_X = c_X$
19. Sample a random $E_2 \subseteq [n] \backslash X$ of size $n^{1-\varepsilon_1} - |E_1|$, let $E = E_1 \cup E_2$
20. For all $i \in E_2$, sample $e_i \leftarrow U_{\{0,1\}}$
21. Define $c$ as $c_i = \begin{cases} a_i, & i \notin E \\ e_i, & i \in E \end{cases}$
22. $(\widetilde{E}, \widetilde{e}) = \mathsf{Copy}(\text{Sim}_0(\mathcal{L}^L, \mathcal{L}^R, f_0), (E, e))$
23. $(c^L, c^R) \leftarrow \text{NMEnc}_0(E, e)$ s.t. $\text{NMDec}_0(g(c^L, c^R)) = (\widetilde{E}, \widetilde{e})$ and $c^L_{\mathcal{L}^L} = x$, $c^R_{\mathcal{L}^R} = y$
24. $\alpha^L = \text{Embed}^L(s^L, c^L)$, $\alpha^R = \text{Embed}^R(s^R, c^R)$
25. $\widetilde{c} = f_1(c, s^L, \alpha^L, s^R, \alpha^R)$
26. If ans $=$

    - $\mathsf{same}^*$: If $\widetilde{c}_Z = c_Z$, output $\mathsf{same}^*$

      (Else) Output $\perp$.
    - $(E^*, e^*)$: If $\text{ECSS.ECorr}(\widetilde{c}) = \perp$, output $\perp$;

      (Else) $\widetilde{a} = \text{ECSS.ECorr}(\widetilde{c})$

      Define $c'$ as $c'_i = \begin{cases} \widetilde{a}_i, & i \notin \widetilde{E} \\ \widetilde{e}_i, & i \in \widetilde{E} \end{cases}$

      If $c' \neq \widetilde{c}$, output $\perp$; (Else) $\widetilde{m} = \text{ECSS.Dec}(\widetilde{a})$

      Output $\widetilde{m}$

# 4. EXPLICIT RATE-1/3 NON-MALLEABLE CODE FOR TWO-LOOKAHEAD AND THREE-SPLIT-STATE TAMPERING

**Lookahead Tampering & Non-malleable Messaging.** Consider the motivating application of non-malleable message transmission, where the high-speed network switches routing the communication between parties shall forward their data packets at several gigabits per second. An adversary, who is monitoring the communication at a network switch, cannot block or slow the information stream, which would outrightly signal her intrusion. So, the adversary is naturally left to innocuously substituting data packets based on all the information that she has seen so far, namely, the *lookahead tampering* model [ADKO15, CGM+16]. This restricts the tampering power of the adversary as she cannot tamper the encoding arbitrarily.

**Split-State Tampering.** A widely studied setting is $k$-split-state tampering [DKO13, ADL14, CZ14, ADKO15, Li17, KOS17]. Here, message is encoded into $k$ states and the adversary can only tamper each of the states independently (and arbitrarily). More formally, the message $m \in \{0,1\}^\ell$ is encoded as $c = (c_1, c_2, \ldots, c_k) \in \{0,1\}^{n_1} \times \{0,1\}^{n_2} \times \cdots \times \{0,1\}^{n_k}$. A tampering function is a $k$-tuple of functions $f = (f_1, f_2, \ldots, f_k)$ s.t. the function $f_i : \{0,1\}^{n_i} \to \{0,1\}^{n_i}$ is an arbitrary function. Note that the tampering function only sees single states locally, and decoding requires aggregating information across all states.

**Our Objective.** Motivated by applications like non-malleable message transmission over high-speed networks, our work studies the limits of the efficiency of constructing non-malleable codes in the $k$-split-state model where a lookahead adversary tampers each state independently, i.e., the *$k$-lookahead model*. We know that constructing non-malleable codes against single state, i.e., $k = 1$, lookahead adversary is impossible [CGM+16]. So, we consider the next best setting of 2-split-state lookahead tampering, where the message is encoded into 2 states and transmitted using 2 independent paths. Each of these states is tampered independently using lookahead tampering. Since split-state lookahead tampering is a sub-class of split-state tampering, a conservative approach is to use generic non-malleable codes in the $k$-split-state, which protect against arbitrary split-state tamperings. Prior to our work,

the most efficient non-malleable codes achieved rate $R = \log\log\ell/\log\ell$ for $k = 2$ [Li18], and rate $R = 1/3$ for $k = 4$ [KOS17]. In a concurrent and independent work, [KOS18] achieves rate $R = 1/3$ for $k = 3$.

As illustrated above, there are natural cryptographic applications where lookahead attacks appropriately model the adversarial threat. We ask the following question: Can we leverage the structure of the lookahead tampering to construct a constant rate non-malleable code that requires establishing least number of, i.e., only 2, independent communication routes between the sender and the receiver?

**Our Results.** We first prove an upper-bound that the rate of any non-malleable code in the 2-split-state lookahead model is at most 1/2. Next, we construct a non-malleable code for the 2-lookahead model with rate $R = 1/3$, which is 2/3-close to the above mentioned optimal upper-bound. En route, we also independently construct a 3-split-state non-malleable code that achieves rate $R = 1/3$. The starting point of all our non-malleable code constructions is the recent construction of [KOS17] in the 4-split-state model.

Finally, we interpret our results in the context of the original motivating example of non-malleable message transmission. It is necessary to establish at least two independent routes of communication to facilitate non-malleable message transmission between two parties. We show that the cumulative size of the encoding of the message sent by the sender must be at least twice the message length when the sender transmits the shares of the encoded message over two independent routes. For this setting, we provide a construction where the encoding of the message is (roughly) three-times the size of the message (1.5x the optimal solution).

### 4.1 Our Contribution

Let $\mathcal{S}_n$ represent the set of all functions from $\{0,1\}^n$ to $\{0,1\}^n$. We call any subset $\mathcal{F} \subseteq \mathcal{S}_n$ a *tampering family* on $\{0,1\}^n$. We denote $k$-split-state tampering families on $\{0,1\}^{n_1+n_2+\cdots+n_k}$ by $\mathcal{F}_1 \times \mathcal{F}_2 \times \cdots \times \mathcal{F}_k$, where $\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_k$ are tampering families on $\{0,1\}^{n_1}, \{0,1\}^{n_2}, \ldots, \{0,1\}^{n_k}$. Here, the codeword is distributed over $k$ states of size $n_1, n_2, \ldots, n_k$.

**(Split-State) Lookahead Tampering.** Motivated by the example in the introduction, instead of considering an arbitrary tampering function for each state, we consider tam-

pering functions that encounter the information as a stream. Let $\mathcal{LA}_{n_1,n_2,\ldots,n_B}$ be the set of all functions $f\colon \{0,1\}^{n_1+n_2+\cdots+n_B} \to \{0,1\}^{n_1+n_2+\cdots+n_B}$ such that there exists functions $f^{(1)}, f^{(2)}, \ldots, f^{(B)}$ with the following properties.

1. For each $1 \leqslant i \leqslant B$, we have $f^{(i)}\colon \{0,1\}^{n_1+n_2+\cdots+n_i} \to \{0,1\}^{n_i}$, and
2. The function $f(x_1, x_2, \ldots, x_B)$ is the concatenation of $f^{(i)}(x_1, x_2, \ldots, x_i)$, i.e., $f(x_1, x_2, \ldots, x_B) = f^{(1)}(x_1)||f^{(2)}(x_1, x_2)||\cdots||f^{(B)}(x_1, x_2, \ldots, x_B)$

Intuitively, the codeword arrives as $B$ blocks of information, and the $i$-th block is tampered based on all the blocks so far $\{1, 2, \ldots, i\}$. In the $k$-split-state lookahead tampering, denoted by *$k$-lookahead*, the tampering function for each state is a lookahead function. The $k$-lookahead tampering family was introduced in [ADKO15] for the purpose of constructing non-malleable codes in the 2-split-state model. A similar notion called block-wise tampering function was introduced by [CGM+16]. Our first result is the hardness result. We give a more precise statement of this result in Theorem 4.3.2.

**Theorem 4.1.1.** *For $k$-lookahead tampering family, the best achievable rate is $1 - 1/k$.*

In fact, we prove the above upper bound for the weakest tampering family in this class where each block in lookahead tampering is a single bit, i.e., $\mathcal{LA}_{n_1,n_2,\ldots,n_B}$ s.t. $B = n$ and $n_i = 1$. For brevity, we represent this function by $\mathcal{LA}_{1^{\otimes n}}$. Surprisingly, analogous to the result of Cheraghchi and Guruswami [CG14a] for the $k$-split-state model, we prove that even against significantly more restricted $k$-lookahead tampering $\mathcal{LA}_{1^{\otimes n_1}} \times \cdots \times \mathcal{LA}_{1^{\otimes n_k}}$, the rate of any non-malleable code is at most $1 - 1/k$ (see Subsection 4.3.1).

We use Figure 4.1 to summarize our positive results in $k$-lookahead and $k$-split-state model and position our results relative to relevant prior works. Intuitively, lower the $k$, the more powerful is the tampering family, and the harder it is to construct the non-malleable codes. The state-of-the-art in non-malleable code construction against $k$-lookahead coincides with the general $k$-split-state model. In particular, no constant-rate non-malleable codes are known even against the restricted 2-lookahead model. We resolve this open question in the positive (with 2/3 the optimal rate).

**Theorem 4.1.2** (Rate-1/3 NMC against 2-Lookahead)**.** *There exists a computationally efficient non-malleable code, with negligible simulation error, against the 2-lookahead tampering*

$\mathcal{LA}_{n_1,n_2} \times \mathcal{LA}_{n_3,n_4}$, *where* $n_1 = (2+o(1))\ell$, $n_2 = o(\ell)$, $n_3 = o(\ell)$, $n_4 = \ell$, *where* $\ell$ *is the length of the message.*

We start from the construction of 4-split-state non-malleable codes by Kanukruthi et al. [KOS17] and leverage a unique characteristic of the (rate-0) 2-split-state code of Aggarwal, Dodis, and Lovett [ADL14], namely *augmented non-malleability* that was identified by [AAG+16].

By manipulating the way we store information in the construction of Theorem 4.1.2, we also obtain the first constant-rate non-malleable codes in 3-split-state.[1]

**Theorem 4.1.3** (Rate-1/3 NMC in 3-Split-State)**.** *There exists a computationally efficient non-malleable code, with negligible simulation error, in the 3-split-state model* $\mathcal{S}_{n_1} \times \mathcal{S}_{n_2} \times \mathcal{S}_{n_3}$, *where* $n_1 = \ell$, $n_2 = (2 + o(1))\ell$, $n_3 = o(\ell)$, *where* $\ell$ *is the length of the message.*

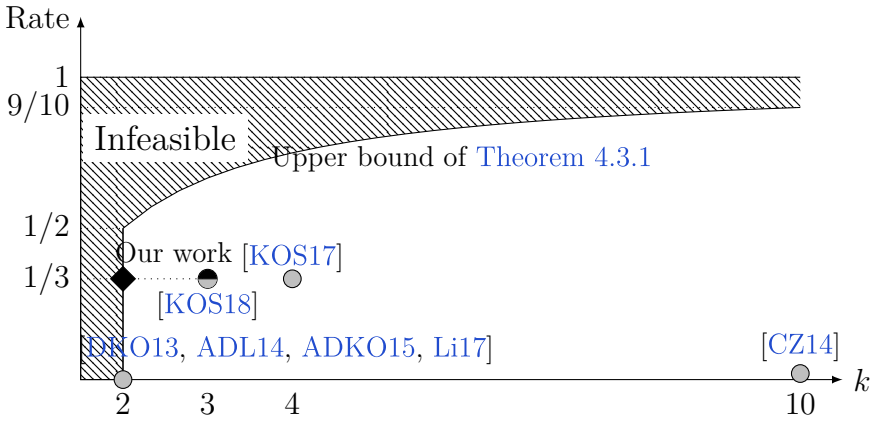



**Figure 4.1.** A comparison of the efficiency of our 2-lookahead non-malleable code with the efficiency of generic $k$-split-state non-malleable codes in the information-theoretic setting. The diamond represents a $k$-lookahead result, and the circles represent $k$-split-state results. Black color represents our results, and gray color represents other known results (includes both prior and concurrent works).

Lastly, [ADKO15] motivated constant-rate construction achieving non-malleability against 2-lookahead tampering along with another particular family of functions (namely, *forgetful functions*) as an intermediate step to constructing constant-rate non-malleable codes in the 2-split-state model. We achieve partial progress towards this goal, and Theorem 4.5.1 summarizes this result.

[1]↑Concurrent and independent work of [KOS18] obtained similar result.

## 4.2 Preliminaries

For any natural number $n$, the symbol $[n]$ denotes the set $\{1, 2, \ldots, n\}$. For a probability distribution $A$ over a finite sample space $\Omega$, $A(x)$ denotes the probability of sampling $x \in \Omega$ according to the distribution $A$ and $x \leftarrow A$ denotes that $x$ is sampled from $\Omega$ according to $A$. For any $n \in \mathbb{N}$, $U_n$ denotes the uniform distribution over $\{0,1\}^n$. Similarly, for a set $S$, $U_S$ denotes the uniform distribution over $S$.

Let $f : \{0,1\}^p \times \{0,1\}^q \longrightarrow \{0,1\}^p \times \{0,1\}^q$. For any $x \in \{0,1\}^p, y \in \{0,1\}^q$, let $(\widetilde{x}, \widetilde{y}) = f(x, y)$. Then, we define $f_x(y) = \widetilde{y}$ and $f_y(x) = \widetilde{x}$. Note that $f_x : \{0,1\}^q \to \{0,1\}^q$ and $f_y : \{0,1\}^p \to \{0,1\}^p$.

### 4.2.1 Augmented Non-malleable codes

Our constructions rely on leveraging a unique characteristic of the non-malleable code in 2-split-state ($\mathcal{S}_{n_1} \times \mathcal{S}_{n_2}$ s.t. $n_1 + n_2 = n$) provided by Aggarwal, Dodis, and Lovett [ADL14], namely *augmented non-malleability*, which was identified by [AAG+16]. We formally define this notion next. Below, we denote the two states of the codeword as $(L, R) \in \{0,1\}^{n_1} \times \{0,1\}^{n_2}$.

**Definition 4.2.1** ($(n_1, n_2, \ell, \varepsilon)$-Augmented Non-malleable Codes against 2-split-state tampering family)**.** *A coding scheme* $(\mathsf{Enc}, \mathsf{Dec})$ *with message length* $\ell$ *is said to be an augmented non-malleable coding scheme against tampering family* $\mathcal{S}_{n_1} \times \mathcal{S}_{n_2}$ *with* $n_1 + n_2 = n$ *and error* $\varepsilon$ *if for all functions* $(f, g) \in \mathcal{S}_{n_1} \times \mathcal{S}_{n_2}$*, there exists a distribution* $\mathrm{SimPlus}_{f,g}$ *over* $\{0,1\}^{n_1} \times (\{0,1\}^{\ell} \cup \{\bot\} \cup \{\mathsf{same}^*\})$ *such that for all* $\mathsf{msg} \in \{0,1\}^{\ell}$*,*

$$\mathrm{TamperPlus}^{\mathsf{msg}}_{f,g} \approx_{\varepsilon} \mathsf{Copy}\left(\mathrm{SimPlus}_{f,g}, \mathsf{msg}\right)$$

*where* $\mathrm{TamperPlus}^{\mathsf{msg}}_{f,g}$ *stands for the following augmented tampering distribution*

$$\mathrm{TamperPlus}^{\mathsf{msg}}_{f,g} := \left\{ \begin{array}{c} (L, R) \leftarrow \mathsf{Enc}(m),\ \widetilde{L} = f(L),\ \widetilde{R} = g(R) \\ \textit{Output}\left(L, \mathsf{Dec}(\widetilde{L}, \widetilde{R})\right) \end{array} \right\}$$

Note that above we abuse notation for $\mathsf{Copy}\,(\mathrm{SimPlus}_{f,g}, \mathsf{msg})$. Formally, it is defined as follows: $\mathsf{Copy}\,(\mathrm{SimPlus}_{f,g}, \mathsf{msg}) = (L, \mathsf{msg})$ when $\mathrm{SimPlus}_{f,g} = (L, \mathsf{same}^*)$ and $\mathrm{SimPlus}_{f,g}$ otherwise.

It was shown in [AAG+16] that the construction of Aggarwal et al. [ADL14] satisfies this stronger definition of augmented non-malleability with rate $1/\mathsf{poly}(\ell)$ and negligible error $\varepsilon$. More formally, the following holds.

**Imported Theorem 4.2.1** ([AAG+16])**.** *For any message length $\ell$, there is a coding scheme $(\mathrm{Enc}^+, \mathrm{Dec}^+)$ of block length $n = p(\ell)$ (where $p$ is a polynomial) that satisfies augmented non-malleability against 2-split-state tampering functions with error that is negligible in $\ell$.*

### 4.2.2 Building Blocks

Next, we describe average min-entropy seeded extractors with small seed and one-time message authentication codes that we use in our construction.

**Definition 4.2.2** (Average conditional min-entropy)**.** *The average conditional min-entropy of a distribution $A$ conditioned on distribution $L$ is defined to be*

$$\widetilde{H}_\infty(A|L) = -\log\left(\mathbb{E}_{\ell \leftarrow L}\left[2^{-H_\infty(A|L=\ell)}\right]\right)$$

Following lemma holds for average conditional min-entropy in the presence of leakage.

**Lemma 4.2.1** ([DORS08])**.** *Let $L$ be an arbitrary $\kappa$-bit leakage on $A$, then $\widetilde{H}_\infty(A|L) \geqslant H_\infty(A) - \kappa$.*

**Definition 4.2.3** (Seeded Average Min-entropy Extractor)**.** *We say* $\mathrm{Ext} : \{0,1\}^n \times \{0,1\}^d \longrightarrow \{0,1\}^\ell$ *is a $(k, \varepsilon)$-average min-entropy strong extractor if for every joint distribution $(A, L)$ such that $\widetilde{H}_\infty(A|L) \geqslant k$, we have that $(\mathrm{Ext}(A, U_d), U_d, L) \approx_\varepsilon (U_\ell, U_d, L)$.*

It is proved in [Vad12] that any extractor is also a average min-entropy extractor with only a loss of constant factor on error. Also, [GUV07] gave strong extractors with small seed length that extract arbitrarily close to $k$ uniform bits. We summarize these in the following lemma.

Combining these results with the following known construction for extractors, we have that there exists average min-entropy extractor that require seed length $\mathcal{O}(\log n + \log(1/\varepsilon))$ and extracts uniform random strings of length arbitrarily close to the conditional min-entropy of the source.

**Lemma 4.2.2** ([GUV07, Vad12]). *For all constants $\alpha > 0$ and all integers $n \geqslant k$, there exists an efficient $(k, \varepsilon)$-average min-entropy strong extractor* $\mathrm{Ext} : \{0,1\}^n \times \{0,1\}^d \longrightarrow \{0,1\}^\ell$ *with seed length $d = \mathcal{O}(\log n + \log(1/\varepsilon))$ and $\ell = (1-\alpha)k - \mathcal{O}(\log(n) + \log(1/\varepsilon))$.*

Next, we define one-time message authentication codes.

**Definition 4.2.4** (Message authentication code). *A $\mu$-secure one-time message authentication code (MAC) is a family of pairs of function*

$$\left\{\mathrm{Tag}_k : \{0,1\}^\alpha \longrightarrow \{0,1\}^\beta,\ \mathrm{Verify}_k : \{0,1\}^\alpha \times \{0,1\}^\beta \longrightarrow \{0,1\}\right\}_{k \in K}$$

*such that*

*(1) For all $m, k$,* $\mathrm{Verify}_k(m, \mathrm{Tag}_k(m)) = 1$.

*(2) For all $m \neq m'$ and $t, t'$,* $\Pr_{k \leftarrow U_K}[\,\mathrm{Tag}_k(m) = t \mid \mathrm{Tag}_k(m') = t'] \leqslant \mu$.

Message authentication code can be constructed from $\mu$-almost pairwise hash function family with the key length $2\log(1/\mu)$. For completeness, we give a construction in Appendix B.1.

### 4.3 Non-malleable Codes against $k$-Lookahead

In this section, we study the $k$-lookahead tampering family. We first prove an upper-bound on the maximum rate that can be achieved for any non-malleable code against $k$-lookahead tampering family. For this, Theorem 4.3.2 states that the maximum rate that can be achieved is roughly $1 - 1/k$. Surprisingly, this matches the upper bound on the rate non-malleable codes against much stronger tampering family of $k$-split-state by [CG14a]. Our upper bound as well as the impossibility result by [CGM+16] rules the information theoretic construction against single state lookahead tampering. On the constructive side, for 2-

lookahead model, the technically most challenging setting among $k$-lookahead tampering families, we construct a non-malleable code that achieves rate 1/3.

**Notation.** Recall that $\mathcal{LA}_{n_1,n_2,\ldots,n_B} \subseteq (\{0,1\}^n)^{\{0,1\}^n}$, where $n = \sum_{i\in[B]} n_i$, denotes the family of lookahead tampering functions $f = (f^{(1)}, f^{(2)}, \ldots, f^{(B)})$ for $f^{(i)} : \{0,1\}^{\sum_{j\in[i]} n_j} \to \{0,1\}^{n_i}$ such that

$$\tilde{c} := f(c) = f^{(1)}(c_1)||f^{(2)}(c_1, c_2)|| \ldots ||f^{(i)}(c_1, \ldots, c_i)|| \ldots ||f^{(B)}(c_1, \ldots, c_B)$$

for $c = c_1||c_2|| \ldots ||c_B$ and for all $i \in [B]$, $c_i \in \{0,1\}^{n_i}$. That is, if $c$ consists of $B$ parts such that $i^{th}$ part has length $n_i$, then $i^{th}$ tampered part depends on first $i$ parts of $c$. We also use $\mathcal{LA}_{m^{\otimes B}}$ to denote the family of lookahead tampering functions $\mathcal{LA}_{\underbrace{m, m, \ldots, m}_{B\text{-times}}}$, i.e., the codeword has $B$ parts of length $m$ each.

### 4.3.1 Impossibility Results for the Split-State Lookahead Model

In this section, we first prove an upper-bound on the rate of any non-malleable encoding against 2-lookahead tampering function, where each bit is treated as a block, i.e., $\mathcal{LA}_{1^{\otimes n/2}} \times \mathcal{LA}_{1^{\otimes n/2}}$. In our proof, we use ideas similar to [CG14a] and the following imported lemma is used in their proof of theorem 5.3 (see [CG13]).[2]

**Imported Lemma 4.3.1.** *For any constant $0 < \delta < \alpha$ and any encoding scheme $(\mathsf{Enc}, \mathsf{Dec})$ with block length $n$ and rate $1 - \alpha + \delta$, the following holds. Let the codeword $c$ be written as $(c_1, c_2) \in \{0,1\}^{\alpha n} \times \{0,1\}^{(1-\alpha)n}$. Let $\eta = \frac{\delta}{4\alpha}$. Then, there exists a set $X_\eta \subseteq \{0,1\}^{\alpha n}$ and two messages $\mathsf{msg}_0, \mathsf{msg}_1$ such that*

$$\Pr\left[c_1 \in X_\eta | \mathsf{Dec}(c) = \mathsf{msg}_0\right] \geqslant \eta$$
$$\Pr\left[c_1 \in X_\eta | \mathsf{Dec}(c) = \mathsf{msg}_1\right] \leqslant \eta/2$$

[2]↑Specifically, in their proof of Theorem 5.3, they picked two messages $s_0$, $s_1$ along with $X_\eta$ that satisfy the property we require for $m_0$, $m_1$ in the imported lemma. Also, we stress that their proof not only showed $s_0$ and $s_1$ exist, but there are *multiple choices* for the pair. This gives us the freedom when we pick our $m_0$ and $m_1$. We make use of this in our proof.

**Theorem 4.3.1.** *Let* $(\mathsf{Enc}, \mathsf{Dec})$ *be any encoding scheme that is non-malleable against the family of tampering functions* $\mathcal{LA}_{1^{\otimes n/2}} \times \mathcal{LA}_{1^{\otimes n/2}}$ *and achieves rate* $1/2+\delta$*, for any constant* $\delta > 0$ *and simulation error* $\varepsilon$*. Then,* $\varepsilon > \delta/8$*.*

*Proof.* Note that any codeword $c$ in support of $\mathsf{Enc}$ consists of two states $c_1$ and $c_2$, each of length $n/2$. We use $c_{i,j}$ for $i \in \{1,2\}$ and $j \in \{1,\ldots,n/2\}$ to denote the $j^{th}$ bit in state $i$. Any tampering function $f = (f_1, f_2)$ generates a tampered codeword $\widetilde{c} = (\widetilde{c_1}, \widetilde{c_2}) = (f_1(c_1), f_2(c_2))$. Below, we will construct a tampering function $f^*$ such that any simulated distribution $\mathsf{Sim}_{f^*}$ will be $\varepsilon$ far from tampering distribution $\mathsf{Tamper}_{f^*}$.

Next, we fix a message $\widehat{\mathsf{msg}}$ and its codeword $\widehat{c}^{(0)} = (\widehat{c}_1^{(0)}, \widehat{c}_2^{(0)}) \in \mathsf{Enc}(\widehat{\mathsf{msg}})$ such that the following holds. Let $\widehat{c}^{(1)} \in \{0,1\}^n$ be such that for all $j \in \{1,\ldots,n/2-1\}$, $\widehat{c}_{1,j}^{(0)} = \widehat{c}_{1,j}^{(1)}$, $\widehat{c}_{1,n/2}^{(0)} \neq \widehat{c}_{1,n/2}^{(1)}$ and $\widehat{c}_2^{(0)} = \widehat{c}_2^{(1)}$. Moreover, we require that $\mathsf{Dec}(\widehat{c}^{(1)}) \neq \widehat{\mathsf{msg}}$. That is, the two codewords are identical except the last bit of first block and the second codeword does not encode the same message[3] $\widehat{\mathsf{msg}}$. Above condition is still satisfied if $\mathsf{Dec}(\widehat{c}^{(1)}) = \perp$.

Since the rate of the given scheme $(\mathsf{Enc}, \mathsf{Dec})$ is $1-1/2+\delta$ (with a constant $\delta$), by Imported Lemma 4.3.1, we have that there exist special messages $\mathsf{msg}_0, \mathsf{msg}_1$ and set $X_\eta$ with the above guarantees where $c_1$ corresponds to the first state. In fact, Imported Lemma 4.3.1 gives many such pair of messages and we will pick such that $\widehat{\mathsf{msg}}, \mathsf{msg}_0, \mathsf{msg}_1$ are all unique.

Now, our tampering function $f^* = (f^*{}_1, f^*{}_2)$ is as follows: $f^*$ tampers a codeword $c = (c_1, c_2)$ to $\widetilde{c} = (\widetilde{c_1}, \widetilde{c_2})$ such that for all $j \in \{1,\ldots,n/2-1\}$, $\widetilde{c}_{1,j} = \widehat{c}_{1,j}^{(0)}$, $\widetilde{c}_{1,n/2} = \widehat{c}_{1,n/2}^{(0)}$ if $c_1 \in X_\eta$, else $\widehat{c}_{1,n/2}^{(1)}$ and $\widetilde{c}_2 = \widehat{c}_2^{(0)}$. That is, if $c_1 \in X_\eta$, the resulting codeword is $\widehat{c}^{(0)}$, else it is $\widehat{c}^{(1)}$. Note that the above tampering attack can be done using a split-state lookahead tampering function.

Finally, it is evident that for message $\mathsf{msg}_0$, the tampering experiment results in $\widehat{\mathsf{msg}}$ with probability at least $\eta$. On the other hand, for message $\mathsf{msg}_1$, the tampering experiment results in $\widehat{\mathsf{msg}}$ with probability at most $\eta/2$. Hence, probability assigned by $\mathsf{Tamper}_{f^*}^{\mathsf{msg}_0}$ and $\mathsf{Tamper}_{f^*}^{\mathsf{msg}_1}$ to message $\widehat{\mathsf{msg}}$ differs by at least $\eta/2$. Since $\widehat{\mathsf{msg}}$ is different from $\mathsf{msg}_0, \mathsf{msg}_1$, it holds that $\varepsilon$, the simulation error of non-malleable code, is at least $\eta/4$ by triangle inequality.

□

[3]↑We note that such codewords would exist otherwise we can show that the last bit of the first state is redundant for decoding. This way we can obtain a smaller encoding. Then, w.l.o.g., we can apply our argument on this new encoding.

The above result can be extended to $k$-lookahead tampering as follows:

**Theorem 4.3.2.** *Let* $(\mathsf{Enc}, \mathsf{Dec})$ *be any encoding scheme that is non-malleable against the family of tampering functions* $\mathcal{LA}_{1^{\otimes n_1}} \ldots \times \ldots \mathcal{LA}_{1^{\otimes n_k}}$ *and achieves rate* $1 - 1/k + \delta$*, for any constant* $\delta > 0$ *and simulation error* $\varepsilon$*. Then,* $\varepsilon > k\delta/16$*.*

*Proof Outline.* The proof follows by doing a similar analysis as above for the largest state. Without loss of generality, let the first state be the largest state, i.e., $n_1 \geqslant n_i$ for all $i \in \{2, \ldots, k\}$. By averaging argument it holds that $n_1 \geqslant n/k$, where $n$ is the block length. Now, the theorem follows along the same lines as the proof of 2-lookahead tampering above when we consider the code for the first state as $c_1$ and rest of the code as $c_2$. We note that the above proof does not require $c_1$ and $c_2$ to have the same size.

### 4.3.2 Rate-1/3 Non-malleable Code in 2-Lookahead Model

In this section, we present our construction for non-malleable codes against 2-lookahead tampering functions. Our construction relies on the following tools. Let $(\mathsf{Tag}, \mathsf{Verify})$ (resp., $(\mathsf{Tag}', \mathsf{Verify}')$) be a $\mu$ (resp., $\mu'$) secure message authentication code with message length $\ell$ (resp., $n$), tag length $\beta$ (resp., $\beta'$) and key length $\gamma$ (resp., $\gamma'$). Let $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^\ell$ be a $(k, \varepsilon_1)$ average min-entropy strong extractor. We define $k$ later during parameter setting. Finally, let $(\mathsf{Enc}^+, \mathsf{Dec}^+)$ be $(n_1^+, n_2^+, \ell^+, \varepsilon^+)$-augmented 2-split-state non-malleable code (see Definition 4.2.1), where $\ell^+ = \gamma + \gamma' + \beta + \beta' + d$. We denote the codewords of this scheme as $(L, R)$ and given a tampering function, we denote the output of the simulator SimPlus as $(L, \mathsf{Ans})$.

**Construction Overview.** We define our encoding and decoding functions formally in Figure 4.2. In our encoding procedure, we first sample a uniform source $w$ of $n$ bits and a uniform seed $s$ of $d$ bits. Next, we extract a randomness $r$ from $(w, s)$ using the strong extractor Ext. We hide the message $\mathsf{msg}$ using $r$ as the one-time pad to obtain a ciphertext $c$. Next, we sample random keys $k_1, k_2$ and authenticate the ciphertext $c$ using $\mathsf{Tag}_{k_1}$ and the source $w$ using $\mathsf{Tag}'_{k_2}$ to obtain tags $t_1$ and $t_2$, respectively. Now, we think of $(k_1, k_2, t_1, t_2, s)$ as the digest and protect it using an augmented 2-state non-malleable encoding $\mathsf{Enc}^+$ to

obtain $(L, R)$. Finally, our codeword is $\Big((c_1, c_2), (c_3, c_4)\Big)$ where $c_1 = w$, $c_2 = R$, $c_3 = L$ and $c_4 = c$.

We also note that $n_1 := |c_1| = |w| = n$, $n_2 := |c_2| = |R| = n_2^+$, $n_3 := |c_3| = |L| = n_1^+$ and $n_4 := |c_4| = |c| = \ell$. From Figure 4.2, it is evident that our construction satisfies perfect correctness.

| $\mathsf{Enc}(m)$: | $\mathsf{Dec}\Big((c_1, c_2), (c_3, c_4)\Big)$: |
|---|---|
| 1. Sample $w \leftarrow U_n$, $s \leftarrow U_d$, $k_1 \leftarrow U_\gamma$, $k_2 \leftarrow U_{\gamma'}$ | 1. Let the tampered states be $\widetilde{w} := c_1$, $\widetilde{R} := c_2$, $\widetilde{L} := c_3$, $\widetilde{c} := c_4$ |
| 2. Compute $r = \mathrm{Ext}(w, s)$, $c = m \oplus r$ | 2. Decrypt $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s}) = \mathsf{Dec}^+(\widetilde{L}, \widetilde{R})$ |
| 3. Compute the tags $t_1 = \mathrm{Tag}_{k_1}(c)$, $t_2 = \mathrm{Tag}'_{k_2}(w)$ | 3. If $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s}) = \perp$, output $\perp$ |
| 4. Compute the 2-state non-malleable encoding $(L, R) \leftarrow \mathsf{Enc}^+(k_1, k_2, t_1, t_2, s)$ | 4. (Else) If $\mathrm{Verify}_{\widetilde{k_1}}(\widetilde{c}, \widetilde{t_1}) = 0$ or $\mathrm{Verify}'_{\widetilde{k_2}}(\widetilde{w}, \widetilde{t_2}) = 0$, output $\perp$ |
| 5. Output the states $\Big((w, R), (L, c)\Big)$ | 5. (Else) Output $\widetilde{c} \oplus \mathrm{Ext}(\widetilde{w}, \widetilde{s})$ |

**Figure 4.2.** Non-malleable coding scheme against $\mathcal{LA}_{n_1,n_2} \times \mathcal{LA}_{n_3,n_4}$, where $n_1 = |w|$, $n_2 = |R|$, $n_3 = |L|$, and $n_4 = |c|$.

**Proof of Non-malleability against 2-lookahead tampering.** Given a tampering function $(f, g) \in \mathcal{LA}_{n_1,n_2} \times \mathcal{LA}_{n_3,n_4}$, where $f = (f^{(1)}, f^{(2)})$ and $g = (g^{(1)}, g^{(2)})$, we formally describe our simulator in Figure 4.3.

Our simulator describes a leakage function $\mathcal{L}(w)$ that captures the leakage required on the source $w$ in order to simulate the tampering experiment. This leakage has five parts $(L, \mathrm{Ans}, \mathrm{flag}_1, \mathrm{flag}_2, \mathrm{mask})$. The values $L$ and Ans are the outputs of simulator SimPlus on tampering function $(g^{(1)}, f_w^{(2)})$, where $f_w^{(2)}$ represents the tampering function on $R$ given $w$. Next, for the case when $\mathrm{Ans} = \mathsf{same}^*$, $\mathrm{flag}_1$ denotes the bit $\widetilde{w} = w$. When $\mathrm{Ans} = (\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s})$, $\mathrm{flag}_2$ captures the bit $\mathrm{Verify}'_{\widetilde{k_2}}(\widetilde{w}, \widetilde{t_2})$, i.e., whether the new key $\widetilde{k_2}$ and tag $\widetilde{t_2}$ are valid authentication on new source $\widetilde{w}$. In this case, the value mask is the extracted output of tampered source $\widetilde{w}$ using tampered seed $\widetilde{s}$.

We give the formal proof on indistinguishability between simulated and tampering distributions in Subsection 4.3.3 using a series of statistically close hybrids.

1. $w \leftarrow U_n$
2. Define leakage function $\mathcal{L}(w) : \{0,1\}^n \longrightarrow \{0,1\}^{n_1^+} \times \{0,1\}^{\beta+\beta'+\gamma+\gamma'+d+1} \times \{0,1\} \times \{0,1\} \times \{0,1\}^\ell$ as the following function:
   (a) $(L, \text{Ans}) \leftarrow \text{SimPlus}_{g^{(1)}, f_w^{(2)}}$, $\widetilde{w} = f^{(1)}(w)$
   (b) If Ans =
   - Case $\perp$: $\text{flag}_1 = 0$, $\text{flag}_2 = 0$, $\text{mask} = 0^\ell$
   - Case $\mathsf{same}^*$: If $(\widetilde{w} = w)$, $\text{flag}_1 = 1$; Else $\text{flag}_1 = 0$
     $\text{flag}_2 = 0$, $\text{mask} = 0^\ell$
   - Case $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s})$: $\text{flag}_1 = 0$, Let $\text{mask} = \text{Ext}(\widetilde{w}, \widetilde{s})$
     If $\left(\text{Verify}'_{\widetilde{k_2}}(\widetilde{w}, \widetilde{t_2})\right) = 1$, $\text{flag}_2 = 1$; Else $\text{flag}_2 = 0$

   (c) $\mathcal{L}(w) := (L, \text{Ans}, \text{flag}_1, \text{flag}_2, \text{mask})$
3. $r \leftarrow U_\ell$, $c = 0^\ell \oplus r$, $\widetilde{c} = g_L^{(2)}(c)$
4. If Ans =
   - Case $\perp$: Output $\perp$
   - Case $\mathsf{same}^*$: If $\left(\widetilde{c} = c \text{ and } \text{flag}_1\right) = 1$, output same*
     Else output $\perp$
   - Case $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s})$: If $\left(\text{Verify}_{\widetilde{k_1}}(\widetilde{c}, \widetilde{t_1})=0 \text{ or } \text{flag}_2 = 0\right)$, output $\perp$
     Else output $\widetilde{c} \oplus \text{mask}$

**Figure 4.3.** The simulator $\mathsf{Sim}_{f,g}$ for the non-malleable code against 2-lookahead tampering family.

**Rate analysis.** We will use $\lambda$ as our security parameter. By Corollary B.1.1, we will let $k_1, k_2$ be of length $2\lambda$, i.e. $\gamma = \gamma' = 2\lambda$ and $t_1, t_2$ will have length $\lambda$, i.e. $\beta = \beta' = \lambda$ and both $(\text{Tag}, \text{Verify})$ and $(\text{Tag}', \text{Verify}')$ will have error $2^{-\lambda}$.

Since we will need to extract $\ell$ bits as a one-time pad to mask the message, by Lemma 4.2.2, we will set min-entropy $k$ to be $(1+\alpha')\ell$ for some constant $\alpha'$ and let Ext be a $((1+\alpha')\ell, 2^{-\lambda})$-strong average min-entropy extractor that extract $\ell$-bit randomness with seed length $\mathcal{O}(\log n + \lambda)$. By our analysis in Subsection 4.3.3, it suffices to have $n - (\ell + n_1^+ + \ell^+ + 3) = n - \ell - p(\log n + \lambda) \geqslant (1+\alpha')\ell$. Hence, we will set $n = (2+\alpha)\ell$ for some constant $\alpha > \alpha'$.

Now the message length for our augmented 2-state non-malleable code will be $2\lambda + 2\lambda + \lambda + \lambda + \mathcal{O}(\log n + \lambda) = \mathcal{O}(\log n + \lambda)$. Now by Theorem 4.2.1, we will let $\zeta$ be the constant such that $p(n^\zeta) = \text{o}(n)$ and set $\lambda = \mathcal{O}\left(n^\zeta\right)$. Hence, the length of $(L, R)$ will be $\text{o}(n)$.

Therefore, the total length of our coding scheme will be $\ell + (2+\alpha)\ell + o(n)$ and the rate is $\frac{1}{3+\alpha}$ with error $\mathcal{O}\left(2^{-n^{\zeta}}\right)$. This completes the proof for Theorem 4.1.2.

### 4.3.3 Proof of Non-Malleability against 2-lookahead (Theorem 4.1.2)

In this section, we prove that our code scheme Figure 4.2 is secure against the tampering family $\mathcal{LA}_{n_1,n_2} \times \mathcal{LA}_{n_3,n_4}$. In order to prove the non-malleability, we need to show that for all tampering functions $(f,g) \in \mathcal{LA}_{n_1,n_2} \times \mathcal{LA}_{n_3,n_4}$, where $f = (f^{(1)}, f^{(2)})$ and $g = (g^{(1)}, g^{(2)})$, our simulator as defined in Figure 4.3 satisfies that, for all $m$, we have

$$\left\{\begin{array}{c} \Big((w,R),(L,c)\Big) \leftarrow \mathsf{Enc}(m) \\ \widetilde{w} = f^{(1)}(w),\ \widetilde{R} = f^{(2)}(w,R) \\ \widetilde{L} = g^{(1)}(L),\ \widetilde{c} = g^{(2)}(L,c) \\ \text{Output: } \tilde{m} = \mathsf{Dec}\Big((\widetilde{w},\widetilde{R}),(\widetilde{L},\widetilde{c})\Big) \end{array}\right\} = \mathsf{Tamper}^{m}_{f,g} \approx \mathsf{Copy}\Big(\mathsf{Sim}_{f,g}\,,\, m\Big)$$

The following sequence of hybrids will lead us from tampering experiment to the simulator. Throughout this section, we use the following color/highlight notation. In a current hybrid, the text in red denotes the changes from the previous hybrid. The text in shaded part represents the steps that will be replaced by red part of the next hybrid.

The initial hybrid represents the tampering experiment $\mathsf{Tamper}^{m}_{f,g}$ and the last hybrid represents $\mathsf{Copy}(\mathsf{Sim}_{f,g}, m)$.

$H_0(f,g,m)$:

1. $w \leftarrow U_n,\ s \leftarrow U_d,\ k_1 \leftarrow U_\gamma,\ k_2 \leftarrow U_{\gamma'}$
2. $r = \mathrm{Ext}(w,s)$, $c = m \oplus r$, $t_1 = \mathrm{Tag}_{k_1}(c)$, $t_2 = \mathrm{Tag}'_{k_2}(w)$
3. $(L,R) \leftarrow \mathsf{Enc}^{+}(k_1,k_2,t_1,t_2,s)$
4. $\widetilde{w} = f^{(1)}(w)$, $\widetilde{R} = f^{(2)}(w,R)$, $\widetilde{L} = g^{(1)}(L)$, $\widetilde{c} = g^{(2)}(L,c)$
5. $(\widetilde{k_1},\widetilde{k_2},\widetilde{t_1},\widetilde{t_2},\widetilde{s}) = \mathsf{Dec}^{+}(\widetilde{L},\widetilde{R})$
6. If $(\widetilde{k_1},\widetilde{k_2},\widetilde{t_1},\widetilde{t_2},\widetilde{s}) = \perp$, output $\perp$
7. Else If $\Big(\mathrm{Verify}_{\widetilde{k_1}}(\widetilde{c},\widetilde{t_1})=0$ or $\mathrm{Verify}'_{\widetilde{k_2}}(\widetilde{w},\widetilde{t_2}) = 0\Big)$, output $\perp$
8. Else Output $\widetilde{c} \oplus \mathrm{Ext}(\widetilde{w},\widetilde{s})$

Next, we rewrite $\widetilde{R} = f^{(2)}(w,R)$ and $\widetilde{c} = g^{(2)}(L,c)$ as $\widetilde{R} = f^{(2)}_{w}(R)$ and $\widetilde{c} = g^{(2)}_{L}(c)$. Now, rearrange the steps leads us to the next hybrid.

$H_1(f, g, m)$:

1. $w \leftarrow U_n,\ s \leftarrow U_d,\ k_1 \leftarrow U_\gamma,\ k_2 \leftarrow U_{\gamma'}$
2. $r = \mathrm{Ext}(w, s)$, $c = m \oplus r$, $t_1 = \mathrm{Tag}_{k_1}(c)$, $t_2 = \mathrm{Tag}'_{k_2}(w)$
3. $\widetilde{w} = f^{(1)}(w)$
4. $(L, R) \leftarrow \mathsf{Enc}^+(k_1, k_2, t_1, t_2, s)$
5. $\widetilde{L} = g^{(1)}(L)$, $\widetilde{R} = f_w^{(2)}(R)$
6. $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s}) = \mathsf{Dec}^+(\widetilde{L}, \widetilde{R})$
7. $\widetilde{c} = g_L^{(2)}(c)$
8. If $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s}) = \perp$, output $\perp$
9. Else If $\Big(\mathrm{Verify}_{\widetilde{k_1}}(\widetilde{c}, \widetilde{t_1})=0$ or $\mathrm{Verify}'_{\widetilde{k_2}}(\widetilde{w}, \widetilde{t_2}) = 0\Big)$, output $\perp$
10. Else Output $\widetilde{c} \oplus \mathrm{Ext}(\widetilde{w}, \widetilde{s})$

Note that shaded steps in the previous hybrid formulate a 2-state tampering experiment onto $(L, R)$. Therefore, we could use the augmented simulator to replace the tampering experiment of augmented two-state non-malleable codes.

$H_2(f, g, m)$:

1. $w \leftarrow U_n,\ s \leftarrow U_d,\ k_1 \leftarrow U_\gamma,\ k_2 \leftarrow U_{\gamma'}$
2. $r = \mathrm{Ext}(w, s)$, $c = m \oplus r$, $t_1 = \mathrm{Tag}_{k_1}(c)$, $t_2 = \mathrm{Tag}'_{k_2}(w)$
3. $\widetilde{w} = f^{(1)}(w)$
4. $(L, \mathrm{Ans}) \leftarrow \mathrm{SimPlus}_{g^{(1)}, f_w^{(2)}}$
5. $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s}) = \mathsf{Copy}\Big(\mathrm{Ans}, (k_1, k_2, t_1, t_2, s)\Big)$.
6. $\widetilde{c} = g_L^{(2)}(c)$
7. If $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s}) = \perp$, output $\perp$
8. Else If $\Big(\mathrm{Verify}_{\widetilde{k_1}}(\widetilde{c}, \widetilde{t_1})=0$ or $\mathrm{Verify}'_{\widetilde{k_2}}(\widetilde{w}, \widetilde{t_2}) = 0\Big)$, output $\perp$
9. Else Output $\widetilde{c} \oplus \mathrm{Ext}(\widetilde{w}, \widetilde{s})$

Now in hybrid $H_3(f, g, m)$, instead of doing $\mathsf{Copy}()$, we do a case analysis on Ans. We note that the hybrids $H_2(f, g, m)$ and $H_3(f, g, m)$ are identical.

$H_3(f, g, m)$:

1. $w \leftarrow U_n,\ s \leftarrow U_d,\ k_1 \leftarrow U_\gamma,\ k_2 \leftarrow U_{\gamma'}$
2. $r = \text{Ext}(w, s)$, $c = m \oplus r$, $t_1 = \text{Tag}_{k_1}(c)$, $t_2 = \text{Tag}'_{k_2}(w)$
3. $\widetilde{w} = f^{(1)}(w)$
4. $(L, \text{Ans}) \leftarrow \text{SimPlus}_{g^{(1)}, f_w^{(2)}}$
5. $\widetilde{c} = g_L^{(2)}(c)$
6. If Ans =

- Case $\perp$: Output $\perp$
- Case same*: If $\Big(\text{Verify}_{k_1}(\widetilde{c}, t_1){=}0 \text{ or } \text{Verify}'_{k_2}(\widetilde{w}, t_2) = 0\Big)$, output $\perp$
  Else output $\widetilde{c} \oplus \text{Ext}(\widetilde{w}, s)$
- Case $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s})$: If $\Big(\text{Verify}_{\widetilde{k_1}}(\widetilde{c}, \widetilde{t_1}){=}0 \text{ or } \text{Verify}'_{\widetilde{k_2}}(\widetilde{w}, \widetilde{t_2}) = 0\Big)$, output $\perp$
  Else output $\widetilde{c} \oplus \text{Ext}(\widetilde{w}, \widetilde{s})$

Next, in hybrid $H_3(f, g, m)$ we change the case when Ans = same*. Note that Ans = same* says that the both the authentication keys $k_1, k_2$ as well as the tags are unchanged. Hence, with probability at least $(1 - \mu - \mu')$, both authentications would verify only if $w$ and $c$ are unchanged. Hence, in $H_4(f, g, m)$, we check if the ciphertext $c$ and source $w$ are the same.

Given that $(\text{Tag}, \text{Verify})$ and $(\text{Tag}', \text{Verify}')$ are $\mu$ and $\mu'$-secure message authentication codes, $H_3(f, g, m) \approx_{\mu+\mu'} H_4(f, g, m)$.

$H_4(f, g, m)$:

Copy

$\Big($

1. $w \leftarrow U_n,\ s \leftarrow U_d,\ k_1 \leftarrow U_\gamma,\ k_2 \leftarrow U_{\gamma'}$
2. $r = \text{Ext}(w, s)$, $c = m \oplus r$, $t_1 = \text{Tag}_{k_1}(c)$, $t_2 = \text{Tag}'_{k_2}(w)$
3. $\widetilde{w} = f^{(1)}(w)$
4. $(L, \text{Ans}) \leftarrow \text{SimPlus}_{g^{(1)}, f_w^{(2)}}$
5. $\widetilde{c} = g_L^{(2)}(c)$
6. If Ans =

- Case $\perp$: Output $\perp$
- Case same*: If $\Big(\widetilde{c} = c \text{ and } \widetilde{w} = w\Big) = 1$, output same*
  Else output $\perp$
- Case $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s})$: If $\Big(\text{Verify}_{\widetilde{k_1}}(\widetilde{c}, \widetilde{t_1}) = 0 \text{ or } \text{Verify}'_{\widetilde{k_2}}(\widetilde{w}, \widetilde{t_2}) = 0\Big)$, output $\perp$
  Else output $\widetilde{c} \oplus \text{Ext}(\widetilde{w}, \widetilde{s})$

$, m\Big)$

We note that the variables $k_1, k_2, t_1, t_2$ are no longer used in the hybrid. Hence, we remove the sampling of these in the next hybrid. It is clear that the two hybrids $H_4(f, g, m)$ and $H_5(f, g, m)$ are identical.

$H_5(f, g, m)$:

Copy

$\Big($

1. $w \leftarrow U_n,\ s \leftarrow U_d,\ r = \mathrm{Ext}(w, s),\ c = m \oplus r$
2. $(L, \mathrm{Ans}) \leftarrow \mathrm{SimPlus}_{g^{(1)}, f_w^{(2)}}$
3. $\widetilde{w} = f^{(1)}(w),\ \widetilde{c} = g_L^{(2)}(c)$
4. If Ans =
   - Case $\perp$: Output $\perp$
   - Case $\mathsf{same}^*$: If $\Big(\widetilde{c} = c$ and $\widetilde{w} = w\Big) = 1$, output same*
     Else output $\perp$
   - Case $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s})$: If $\Big(\mathrm{Verify}_{\widetilde{k_1}}(\widetilde{c}, \widetilde{t_1})$=0 or $\mathrm{Verify}'_{\widetilde{k_2}}(\widetilde{w}, \widetilde{t_2}) = 0\Big)$, output $\perp$
     Else output $\widetilde{c} \oplus \mathrm{Ext}(\widetilde{w}, \widetilde{s})$

$, m\Big)$

Now, we wish to use the property of average min-entropy extractor to remove the dependence between $c$ and $w$. Before we do the trick, we shall first rearrange the steps in $H_5(f, g, m)$ to get $H_6(f, g, m)$. We process all the leakage we need at the first part of our hybrid and use only the leakage of $w$ in the remaining. Intuitively, when $\mathrm{Ans} = \mathsf{same}^*$, $\mathrm{flag}_1$ records whether $\widetilde{w} = w$ and when $\mathrm{Ans} = (\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s})$, $\mathrm{flag}_2$ records whether $\widetilde{w}$ can pass the MAC verification under new key and tag and mask is the new one-time pad we need for decoding the tampered message. We note that the hybrids $H_5(f, g, m)$ and $H_6(f, g, m)$ are identical.

$H_6(f, g, m)$:

Copy

1. $w \leftarrow U_n$
2. $(L, \text{Ans}) \leftarrow \text{SimPlus}_{g^{(1)}, f_w^{(2)}},\ \widetilde{w} = f^{(1)}(w)$
3. If Ans =
   - Case same*: If $(\widetilde{w} = w)$, $\text{flag}_1 = 1$; Else $\text{flag}_1 = 0$
   - Case $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s})$: If $\left(\text{Verify}'_{\widetilde{k_2}}(\widetilde{w}, \widetilde{t_2})\right) = 1$, $\text{flag}_2 = 1$;
     Else $\text{flag}_2 = 0$.
     Let $\text{mask} = \text{Ext}(\widetilde{w}, \widetilde{s})$

$\left(\right.$ $\left., m\right)$

4. $s \leftarrow U_d,\ r = \text{Ext}(w, s),\ c = m \oplus r,\ \widetilde{c} = g_L^{(2)}(c)$
5. If Ans =
   - Case $\perp$: Output $\perp$
   - Case same*: If $\left(\widetilde{c} = c \text{ and } \text{flag}_1\right) = 1$, output same*
     Else output $\perp$
   - Case $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s})$: If $\left(\text{Verify}_{\widetilde{k_1}}(\widetilde{c}, \widetilde{t_1})=0 \text{ or } \text{flag}_2 = 0\right)$, output $\perp$
     Else output $\widetilde{c} \oplus \text{mask}$

In the next hybrid, we formalize $(L, \text{Ans}, \text{flag}_1, \text{flag}_2, \text{mask})$ as the leakage on source $w$. Note that the hybrids $H_6(f, g, m)$ and $H_7(f, g, m)$ are identical.

$H_7(f, g, m)$:

Copy

1. $w \leftarrow U_n$
2. Leakage function $\mathcal{L}(w) : \{0,1\}^n \longrightarrow \{0,1\}^{n_1^+} \times \{0,1\}^{\beta+\beta'+\gamma+\gamma'+d+1} \times \{0,1\} \times \{0,1\} \times \{0,1\}^\ell$ be the following function:
   (a) $(L, \text{Ans}) \leftarrow \text{SimPlus}_{g^{(1)}, f_w^{(2)}}$, $\widetilde{w} = f^{(1)}(w)$
   (b) If Ans =
   - Case $\perp$: $\text{flag}_1 = 0$, $\text{flag}_2 = 0$, $\text{mask} = 0^\ell$
   - Case same*: If $(\widetilde{w} = w)$, $\text{flag}_1 = 1$; Else $\text{flag}_1 = 0$
     $\text{flag}_2 = 0$, $\text{mask} = 0^\ell$
   - Case $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s})$: $\text{flag}_1 = 0$, Let $\text{mask} = \text{Ext}(\widetilde{w}, \widetilde{s})$
     If $\left(\text{Verify}'_{\widetilde{k_2}}(\widetilde{w}, \widetilde{t_2})\right) = 1$, $\text{flag}_2 = 1$; Else $\text{flag}_2 = 0$

   (c) $\mathcal{L}(w) := (L, \text{Ans}, \text{flag}_1, \text{flag}_2, \text{mask})$
3. $s \leftarrow U_d$, $r = \text{Ext}(w, s)$, $c = m \oplus r$, $\widetilde{c} = g_L^{(2)}(c)$
4. If Ans =
   - Case $\perp$: Output $\perp$
   - Case same*: If $\left(\widetilde{c} = c \text{ and } \text{flag}_1\right) = 1$, output same*; Else output $\perp$
   - Case $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s})$: If $\left(\text{Verify}_{\widetilde{k_1}}(\widetilde{c}, \widetilde{t_1})=0 \text{ or } \text{flag}_2 = 0\right)$, output $\perp$
     Else output $\widetilde{c} \oplus \text{mask}$

$\Big($ $\Big)$, $m$

In the next hybrid, we replace the extracted output $r$ with a uniform random $\ell$ bit string. We argue that the hybrids $H_7(f, g, m)$ and $H_8(f, g, m)$ are $\varepsilon_1$ close for appropriate length $n$ of source $w$.

Since $\mathcal{L}(w)$ outputs a $\ell + n_1^+ + \ell^+ + 3$ bits of leakage, by Lemma 4.2.1, $H_\infty(W|\mathcal{L}(W)) \geqslant n - (\ell + n_1^+ \ell^+ + 3)$. Here, $W$ denotes the random variable corresponding to $w$. We will pick $n$ such that $n - (\ell + n_1^+ + \ell^+ + 3) > \ell$ for the min-entropy extraction to give a uniform string (see Lemma 4.2.2).

$H_8(f, g, m)$:

Copy

1. $w \leftarrow U_n$
2. Leakage function $\mathcal{L}(w) : \{0,1\}^n \longrightarrow \{0,1\}^{n_1^+} \times \{0,1\}^{\beta+\beta'+\gamma+\gamma'+d+1} \times \{0,1\} \times \{0,1\} \times \{0,1\}^\ell$ be the following function:
   (a) $(L, \text{Ans}) \leftarrow \text{SimPlus}_{g^{(1)}, f_w^{(2)}}$, $\widetilde{w} = f^{(1)}(w)$
   (b) If Ans =
   - Case $\perp$: $\text{flag}_1 = 0$, $\text{flag}_2 = 0$, $\text{mask} = 0^\ell$
   - Case same*: If $(\widetilde{w} = w)$, $\text{flag}_1 = 1$; Else $\text{flag}_1 = 0$
     $\text{flag}_2 = 0$, $\text{mask} = 0^\ell$
   - Case $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s})$: $\text{flag}_1 = 0$, Let $\text{mask} = \text{Ext}(\widetilde{w}, \widetilde{s})$
     If $\left(\text{Verify}'_{\widetilde{k_2}}(\widetilde{w}, \widetilde{t_2})\right) = 1$, $\text{flag}_2 = 1$; Else $\text{flag}_2 = 0$

   (c) $\mathcal{L}(w) := (L, \text{Ans}, \text{flag}_1, \text{flag}_2, \text{mask})$
3. $r \leftarrow U_\ell$, $c = m \oplus r$, $\widetilde{c} = g_L^{(2)}(c)$
4. If Ans =
   - Case $\perp$: Output $\perp$
   - Case same*: If $\left(\widetilde{c} = c \text{ and } \text{flag}_1\right) = 1$, output same*; Else output $\perp$
   - Case $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s})$: If $\left(\text{Verify}_{\widetilde{k_1}}(\widetilde{c}, \widetilde{t_1})=0 \text{ or } \text{flag}_2 = 0\right)$, output $\perp$
     Else output $\widetilde{c} \oplus \text{mask}$

$\Big($ $\Big)$, $m\Big)$

Finally, notice that the distribution of $c$ is independent of $m$ and we can use the message $0^\ell$. This gives us our simulator. Clearly $H_8(f, g, m) = H_9(f, g, m)$. Notice that $H_9(f, g, m) = \text{Copy}\Big(\text{Sim}_{f,g}, m\Big)$.

$H_9(f, g, m)$:

Copy

1. $w \leftarrow U_n$
2. Leakage function $\mathcal{L}(w) : \{0,1\}^n \longrightarrow \{0,1\}^{n_1^+} \times \{0,1\}^{\beta+\beta'+\gamma+\gamma'+d+1} \times \{0,1\} \times \{0,1\} \times \{0,1\}^\ell$ be the following function:
   (a) $(L, \text{Ans}) \leftarrow \text{SimPlus}_{g^{(1)}, f_w^{(2)}}$, $\widetilde{w} = f^{(1)}(w)$
   (b) If Ans =
   - Case $\perp$: $\text{flag}_1 = 0$, $\text{flag}_2 = 0$, $\text{mask} = 0^\ell$
   - Case $\mathsf{same}^*$: If $(\widetilde{w} = w)$, $\text{flag}_1 = 1$; Else $\text{flag}_1 = 0$
     $\text{flag}_2 = 0$, $\text{mask} = 0^\ell$
   - Case $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s})$: $\text{flag}_1 = 0$, Let $\text{mask} = \text{Ext}(\widetilde{w}, \widetilde{s})$
     If $\left(\text{Verify}'_{\widetilde{k_2}}(\widetilde{w}, \widetilde{t_2})\right) = 1$, $\text{flag}_2 = 1$; Else $\text{flag}_2 = 0$

   $\left(\quad\right), m\Big)$

   (c) $\mathcal{L}(w) := (L, \text{Ans}, \text{flag}_1, \text{flag}_2, \text{mask})$
3. $r \leftarrow U_\ell$, $c = 0^\ell \oplus r$, $\widetilde{c} = g_L^{(2)}(c)$
4. If Ans =
   - Case $\perp$: Output $\perp$
   - Case $\mathsf{same}^*$: If $\left(\widetilde{c} = c \text{ and } \text{flag}_1\right) = 1$, output same*
     Else output $\perp$
   - Case $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s})$: If $\left(\text{Verify}_{\widetilde{k_1}}(\widetilde{c}, \widetilde{t_1})=0 \text{ or } \text{flag}_2 = 0\right)$, output $\perp$
     Else output $\widetilde{c} \oplus \text{mask}$

## 4.4 Construction for 3-Split-State Non-malleable Code

By re-organizing the information between states, we also obtain a rate-1/3 3-split-state non-malleable codes. Our coding scheme is defined in Figure 4.4. Specifically, instead of storing $w$ with $R$ and $L$ with $c$, we merge $w$ and $L$ into one state and store $c$, $(w, L)$ and $R$ independently. The proof of non-malleability is similar to the proof for 2-lookahead tampering family. We defer the proof to Appendix B.2. By similar analysis as in 2-lookahead case, it is easy to see our non-malleable codes in 3-split-state scheme also has rate-1/3.

## 4.5 Forgetful tampering in the 2-lookahead Model

In this section we restrict ourselves to the 2-lookhead model. Let us define an additional family of tampering functions. Consider a tampering function $f\colon \{0,1\}^{n_1+n_2+n_3+n_4} \rightarrow$

| $\mathsf{Enc}(m)$: | $\mathsf{Dec}(c_1, c_2, c_3)$: |
|---|---|
| 1. Sample $w \leftarrow U_n$, $s \leftarrow U_d$, $k_1 \leftarrow U_\gamma$, $k_2 \leftarrow U_{\gamma'}$ | 1. Let the tampered states be $\widetilde{c} := c_1$, $(\widetilde{w}, \widetilde{L}) := c_2$, $\widetilde{R} := c_3$ |
| 2. Compute $r = \mathrm{Ext}(w, s)$, $c = m \oplus r$ | 2. Decrypt $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s}) = \mathrm{Dec}^+(\widetilde{L}, \widetilde{R})$ |
| 3. Compute the tags $t_1 = \mathrm{Tag}_{k_1}(c)$ and $t_2 = \mathrm{Tag}'_{k_2}(w)$ | 3. If $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s}) = \perp$, output $\perp$ |
| 4. Compute the 2-state non-malleable encoding: $(L, R) \leftarrow \mathrm{Enc}^+(k_1, k_2, t_1, t_2, s)$ | 4. (Else) If $\mathrm{Verify}_{\widetilde{k_1}}(\widetilde{c}, \widetilde{t_1}) = 0$ or $\mathrm{Verify}'_{\widetilde{k_2}}(\widetilde{w}, \widetilde{t_2}) = 0$, then output $\perp$ |
| 5. Output the three states $\Big(c, (w, L), R\Big)$ | 5. (Else) Output $\widetilde{c} \oplus \mathrm{Ext}(\widetilde{w}, \widetilde{s})$ |

**Figure 4.4.** Non-malleable coding scheme against 3-split-state tampering.

$\{0,1\}^{n_1+n_2+n_3+n_4}$. The function $f$ is *1-forgetful*, if there exists a function $g\colon \{0,1\}^{n_2+n_3+n_4} \to \{0,1\}^{n_1+n_2+n_3+n_4}$ such that $f(x_1, x_2, x_3, x_4) = g(x_2, x_3, x_4)$ for all $x_1 \in \{0,1\}^{n_1}$, $x_2 \in \{0,1\}^{n_2}$, $x_3 \in \{0,1\}^{n_3}$, and $x_4 \in \{0,1\}^{n_4}$. Intuitively, the tampering function $f$ forgets its first $n_1$-bits of the codeword and do the entire tampering using only $x_2, x_3, x_4$. The set of all functions that are 1-forgetful are represented by $\mathcal{FOR}_{n_1,n_2,n_3,n_4-\{1\}}$. Analogously, we define $\mathcal{FOR}_{n_1,n_2,n_3,n_4-\{i\}}$, for each $i \in \{2,3,4\}$.

Aggarwal et al. [ADKO15] proved that we can construct constant-rate non-malleable code in the 2-split-state from a constant-rate non-malleable code that protects against the following tampering family[4]

$$\Big(\mathcal{LA}_{n_1,n_2} \times \mathcal{LA}_{n_3,n_4}\Big) \bigcup_{i=1}^{4} \mathcal{FOR}_{n_1,n_2,n_3,n_4-\{i\}}$$

We make partial progress towards the goal of constructing non-malleable codes secure against above tampering family (and hence, constant rate codes against 2-split-state family), and prove the following theorem.

**Theorem 4.5.1.** *For all constants $\alpha$, there exists a constant $\zeta$ and a computationally efficient non-malleable coding scheme against $\Big(\mathcal{LA}_{n_1,n_2} \times \mathcal{LA}_{n_3,n_4}\Big) \cup \mathcal{FOR}_{n_1,n_2,n_3,n_4-\{1\}} \cup \mathcal{FOR}_{n_1,n_2,n_3,n_4-\{3\}}$ with rate $\frac{1}{4+\alpha}$ and error $2^{-n^\zeta}$.*

[4]↑Specifically, Theorem 30 in [ADKO15] states that there exists a constant-rate non-malleable reduction from 2-split-state tampering family to the following tampering function family consisting of union of split-state lookahead and forgetful tampering functions.

We now give a construction Figure 4.5 of constant-rate non-malleable code against $\left(\mathcal{LA}_{n_1,n_2} \times \mathcal{LA}_{n_3,n_4}\right) \cup \mathcal{FOR}_{n_1,n_2,n_3,n_4-\{1\}} \cup \mathcal{FOR}_{n_1,n_2,n_3,n_4-\{3\}}$.

| $\mathsf{Enc}(m)$: | $\mathsf{Dec}\left(c_1, c_2, c_3, c_4\right)$: |
|---|---|
| 1. Sample $w_1 \leftarrow U_n$, $w_2 \leftarrow U_{n'}$, $s \leftarrow U_d$, $k_1 \leftarrow U_\gamma$, $k_2 \leftarrow U_{\gamma'}$, Let $w := (w_1, w_2)$ | 1. Let the tampered states be $\widetilde{w_1} := c_1$, $\widetilde{R} := c_2$, $(\widetilde{w_2}, \widetilde{L}) := c_3$, $\widetilde{c} := c_4$, Let $\widetilde{w} := (\widetilde{w_1}, \widetilde{w_2})$ |
| 2. Compute $r = \mathrm{Ext}(w, s)$, $c = m \oplus r$ | 2. Decrypt $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s}) = \mathsf{Dec}^+(\widetilde{L}, \widetilde{R})$ |
| 3. Compute the tags $t_1 = \mathrm{Tag}_{k_1}(c)$, $t_2 = \mathrm{Tag}'_{k_2}(w)$ | 3. If $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s}) = \perp$, output $\perp$ |
| 4. Compute the 2-state non-malleable encoding $(L, R) \leftarrow \mathsf{Enc}^+(k_1, k_2, t_1, t_2, s)$ | 4. Else If $\mathrm{Verify}_{\widetilde{k_1}}(\widetilde{c}, \widetilde{t_1}) = 0$ or $\mathrm{Verify}'_{\widetilde{k_2}}(\widetilde{w}, \widetilde{t_2}) = 0$, output $\perp$ |
| 5. Output the four states $w_1, R, (w_2, L), c$ | 5. Else Output $\widetilde{c} \oplus \mathrm{Ext}(\widetilde{w}, \widetilde{s})$ |

**Figure 4.5.** Non-malleable coding scheme against $\left(\mathcal{LA}_{n_1,n_2} \times \mathcal{LA}_{n_3,n_4}\right) \cup \mathcal{FOR}_{n_1,n_2,n_3,n_4-\{1\}} \cup \mathcal{FOR}_{n_1,n_2,n_3,n_4-\{3\}}$

The proof is similar to the proof of non-malleability against 2-lookahead tampering family. Hence, we defer it to Appendix B.3.

# 5. THE MINIMAL COMPLEXITY FOR OPTIMAL-FAIR COIN-TOSSING

Secure multi-party computation [Yao82, GMW87] allows mutually distrusting parties to compute securely over their private data. However, guaranteeing output delivery to honest parties when the adversarial parties may abort during the protocol execution has been a challenging objective. A long line of highly influential works has undertaken the task of defining security with guaranteed output delivery (i.e., *fair computation*) and fairly computing functionalities [GHKL08, BOO10, GK10, BLOO11, AP13, ALR13, HT14, Ash14, Mak14, ABMO15, AO16, BHLT17]. This chapter considers the case when honest parties are not in the majority. In particular, as is standard in this research, the sequel relies on the representative task of two-party secure coin-tossing, an elegant functionality providing uncluttered access to the primary bottlenecks of achieving security in any specific adversarial model.

In the *information-theoretic plain model*, one of the parties can fix the coin-tossing protocol's output (using attacks in two-player zero-sum games, or games against nature [Pap83]). If the parties additionally have access to the commitment functionality (a.k.a., the information-theoretic *commitment-hybrid*), an adversary is forced to follow the protocol honestly (otherwise, the adversary risks being identified), or abort the protocol execution prematurely. Against such adversaries, referred to as *fail-stop adversaries* [CI93], there are coin-tossing protocols [Blu82, BD84, ABC+85, Cle86] where a fail-stop adversary can change the honest party's output distribution by at most $\mathcal{O}\left(1/\sqrt{r}\right)$, where $r$ is the round-complexity of the protocol. That is, these protocols are $\mathcal{O}\left(1/\sqrt{r}\right)$-*insecure*. In a ground-breaking result, Moran, Naor, and Segev [MNS09] constructed the first secure coin-tossing protocol in the oblivious transfer-hybrid [Rab81, Rab05, EGL82] that is $\mathcal{O}\left(1/r\right)$-insecure. No further security improvements are possible because Cleve [Cle86] proved that $\mathcal{O}\left(1/r\right)$-insecurity is unavoidable; hence, the protocol by Moran, Naor, and Segev is *optimal*.

Incidentally, all fair computation protocols (not just coin-tossing, see, for example, [GHKL08, BOO10, GK10, BLOO11, AP13, ALR13, HT14, Ash14, Mak14, ABMO15, AO16, BHLT17]) rely on the oblivious transfer functionality to achieve $\mathcal{O}\left(1/r\right)$-insecurity. A fundamental

| | Secure Construction | Adversarial Attack |
|---|---|---|
| Pessiland | Fail-stop Adversary: | In General: constant-unfair [HO11] |
| | | Fail-stop Adversary: $1/\sqrt{r}$-unfair [CI93] |
| Minicrypt | One-way Functions: $1/\sqrt{r}$-unfair [Blu82, BD84, ABC+85, Cle86] | $1/\sqrt{r}$-unfair [This Chapter] |
| Cryptomania | Public-key Encryption: | $1/\sqrt{r}$-unfair [This Chapter] |
| | PKE + $f$-hybrid, $f \not\to$ OT: | $1/\sqrt{r}$-unfair [This Chapter] |
| | Oblivious Transfer: $1/r$-unfair [MNS09] | $1/r$-unfair [Cle86] |

**Figure 5.1.** The first column summarizes of the most secure fair coin-tossing protocols in Impagliazzo's worlds [Imp95]. Corresponding to each of these worlds, the second column has the best attacks on these fair coin-tossing protocols. The red cells are the results we shall present in this chapter.

principle in theoretical cryptography is to securely realize cryptographic primitives based on the minimal computational hardness assumptions. Consequently, the following question is natural.

*Is oblivious transfer necessary for optimal fair computation?*

**Summary of our results.** This chapter studies the insecurity of fair coin-tossing protocols in Minicrypt and (various levels of) Cryptomania [Imp95]. Our contributions are three-fold.

1. First, we prove that any coin-tossing protocol that uses one-way functions in a (fully) black-box manner must be $\Omega(1/\sqrt{r})$-insecure.

2. Second, using simillar techniques from [MMP14], we generalize the first result to prove that any coin-tossing protocol using public-key encryption in a (fully) black-box manner must also be $\Omega(1/\sqrt{r})$-insecure.

3. Finally, we prove a dichotomy for two-party secure (possibly, *randomized* output) function evaluation functionalities. For any secure function evaluation functionality $f$, either (A) optimal fair coin-tossing exists in the information-theoretic $f$-hybrid, or (B) any

coin-tossing protocol in the $f$-hybrid, even using public-key encryption algorithms in a black-box manner, is $\Omega(1/\sqrt{r})$-insecure.

Our hardness of computation results hold even for a game-theoretic definition of fairness as well (which extends to the stronger simulation-based security definition). As shown in Figure 5.1, our results further reinforce the widely-held perception that oblivious transfer is necessary for optimal fair coin-tossing.

## 5.1 Preliminaries

We use uppercase letters for random variables, (corresponding) lowercase letters for their values, and calligraphic letters for sets. For a joint distribution $(A, B)$, $A$ and $B$ represent the marginal distributions, and $A \times B$ represents the product distribution where one samples from the marginal distributions $A$ and $B$ independently.

For a sequence $(X_1, X_2, \ldots)$, we use $X_{\leqslant i}$ to denote the joint distribution $(X_1, X_2, \ldots, X_i)$. Similarly, for any $(x_1, x_2, \ldots) \in \Omega_1 \times \Omega_2 \times \cdots$, we define $x_{\leqslant i} := (x_1, x_2, \ldots, x_i) \in \Omega_1 \times \Omega_2 \times \cdots \times \Omega_i$. Let $(M_1, M_2, \ldots, M_r)$ be a joint distribution over sample space $\Omega_1 \times \Omega_2 \times \cdots \times \Omega_r$, such that for any $i \in \{1, 2, \ldots, n\}$, $M_i$ is a random variable over $\Omega_i$. A (real-valued) random variable $X_i$ is said to be $M_{\leqslant i}$ *measurable* if there exists a deterministic function $f \colon \Omega_1 \times \cdots \times \Omega_i \to \mathbb{R}$ such that $X_i = f(M_1, \ldots, M_i)$. A random variable $\tau \colon \Omega_1 \times \cdots \times \Omega_r \to \{1, 2, \ldots, r\}$ is called a *stopping time*, if the random variable $\mathbb{1}_{\tau \leqslant i}$ is $M_{\leqslant i}$ measurable, where $\mathbb{1}$ is the indicator function. For a more formal treatment of probability spaces, $\sigma$-algebras, filtrations, and martingales, refer to, for example, [Sch17].

The following inequality shall be helpful for our proof.

**Theorem 5.1.1** (Jensen's inequality)**.** *If $f$ is a multivariate convex function, then $\mathrm{E}\left[f\left(\vec{X}\right)\right] \geqslant f\left(\mathrm{E}\left[\vec{X}\right]\right)$, for all probability distributions $\vec{X}$ over the domain of $f$.*

*In particular, $f(x, y, z) = (x - y - z)^2$ is a tri-variate convex function where Jensen's inequality applys.*

## 5.2 A Key Inequality

In this section, we prove an inequality (due to [KMW20]), which turns out to be surprisingly useful later.

**Lemma 5.2.1** (Key Technical Lemma)**.** *For $n \in \mathbb{N}^*$, define*

$$\Gamma_n := \frac{1}{\sqrt{n+3}}.$$

*For all $x, a, b \in [0,1]$, we have*

$$\max\left( |x-a| + |x-b| \ , \ \Gamma_n \cdot x(1-x) \right) \geqslant \Gamma_{n+1} \cdot \left( x(1-x) + (x-a)^2 + (x-b)^2 \right).$$

This technical lemma is a direct consequence of the inequality below with $u = |x-a| + |x-b|$, $v = x(1-x)$, and $a_n = 1/\Gamma_n$.

**Claim 5.2.1.** *Let $\{a_n\}_{n\in\mathbb{N}^*}$ be a sequence of real numbers such that $a_1 \geqslant 2$ and $a_{n+1} \geqslant a_n + 1/(4a_n)$, for all $n \in \mathbb{N}$. For all $u \in [0,2]$ and $v \in [0,1/4]$, we have*

$$\max\left( u \ , \ \frac{v}{a_n} \right) \geqslant \frac{u^2+v}{a_{n+1}}.$$

*Proof.* We prove this by case analysis.

- **Suppose $u \leqslant \frac{v}{a_n}$.**

$$\begin{aligned}
& \frac{v}{a_n} = \max\left\{u, \frac{v}{a_n}\right\} \geqslant \frac{u^2+v}{a_{n+1}} \\
\Longleftarrow \quad & \frac{v}{a_n} \geqslant \frac{u^2+v}{a_n + 1/4a_n} && (\because a_{n+1} \geqslant a_n + 1/(4a_n)) \\
\Longleftrightarrow \quad & \frac{v}{4a_n^2} \geqslant u^2 \\
\Longleftarrow \quad & \frac{v}{4a_n^2} \geqslant \frac{v^2}{a_n^2} && (\because u \leqslant \tfrac{v}{a_n}) \\
\Longleftrightarrow \quad & \frac{1}{4} \geqslant v.
\end{aligned}$$

- **Suppose $u \geqslant \frac{v}{a_n}$.** Substitute $u = (v/a_n + \varepsilon)$, where $\varepsilon > 0$. We need to prove

$$
\begin{aligned}
& u = \max\left\{u, \frac{v}{a_n}\right\} \geqslant \frac{u^2+v}{a_{n+1}} \\
\Longleftarrow \quad & u \geqslant \frac{u^2+v}{a_n + 1/4a_n} && (\because a_{n+1} \geqslant a_n + 1/(4a_n)) \\
\Longleftrightarrow \quad & \frac{v}{a_n} + \varepsilon \geqslant \frac{\left(\frac{v}{a_n}+\varepsilon\right)^2 + v}{a_n + 1/4a_n} \\
\Longleftrightarrow \quad & \varepsilon a_n + \frac{v}{4a_n^2} + \frac{\varepsilon}{4a_n} \geqslant \frac{v^2}{a_n^2} + \frac{2\varepsilon v}{a_n} + \varepsilon^2 \\
\Longleftarrow \quad & \varepsilon a_n + \frac{\varepsilon}{4a_n} \geqslant \frac{2\varepsilon v}{a_n} + \varepsilon^2 = \left(\frac{v}{a_n} + u\right)\varepsilon && (\because v \leqslant 1/4) \\
\Longleftrightarrow \quad & a_n + \frac{1}{4a_n} \geqslant \frac{v}{a_n} + u \\
\Longleftarrow \quad & a_n \geqslant u. && \square
\end{aligned}
$$

### 5.3 Our Approach: Potential function and Inductive Proof

In this section, we present an overview of our technical approach. The proofs of our separation results on one-way functions, public-key encryptions, and $f$-hybrid all follow this technical approach.

Let us first introduce some notations. Consider an $r$-message coin-tossing protocol between Alice and Bob. We stress that Alice and Bob have oracle access to some function. For example, in the random oracle mode, parties have access to a random oracle; in the $f$-hybrid, parties have access to a trusted party realizing the functionality $f$. We shall let $X_i$ represent the expected output conditioned on the first $i$ messages. Let $D_i^{\mathsf{A}}$ be the expectation of Alice's $i^{th}$ defense coin conditioned on the first $i$ messages. Similarly, let $D_i^{\mathsf{B}}$ be the expectation of Bob's $i^{th}$ defense coin conditioned on the first $i$ messages.

Given any coin-tossing protocol $\pi$ and a stopping time $\tau$, we define the following score function that captures the susceptibility of this protocol with respect to this particular stopping time.

**Definition 5.3.1** (Score function)**.** *Let $\pi$ be any $r$-message coin tossing protocol. Let $\mathsf{P} \in \{\mathsf{A}, \mathsf{B}\}$ be the party who sends the last message of the protocol. For any stopping time $\tau$, define*

$$\mathsf{Score}(\pi, \tau) := \mathrm{E}\Big[\mathbb{1}_{(\tau \neq r) \vee (\mathsf{P} \neq \mathsf{A})} \cdot \Big|X_\tau - D_\tau^{\mathsf{A}}\Big| + \mathbb{1}_{(\tau \neq r) \vee (\mathsf{P} \neq \mathsf{B})} \cdot \Big|X_\tau - D_\tau^{\mathsf{B}}\Big|\Big].$$

We clarify that the binary operator $\vee$ in the expression above represents the boolean OR operation.

To provide additional perspectives to this definition, we make the following remarks.

1. Suppose Alice is about to send $(m_i^*, h_i^*)$ as the $i^{th}$ message. In the information-theoretic plain model, prior works [CI93, KMM19] consider the gap between the expected output before and after this message. Intuitively, since Alice is sending this message, she could utilize this gap to attack Bob, because Bob's defense cannot keep abreast of this new information. However, when parties have oracle access to some functionalities, both parties are potentially vulnerable to this gap. This is due to the fact that the new message could reveal information about both parties' private state (for instance, it might reveal Bob's commitments sent in previous messages using the random oracle as an idealized one-way function). Then, Alice's defense cannot keep abreast of this new information either and thus Alice is potentially vulnerable.

2. Due to the reasons above, for every message, we consider the potential deviations that *both* parties can cause by aborting appropriately. Suppose we are at a partial transcript where Alice just sent the last message $(m_i^*, h_i^*)$. Suppose this partial transcript belongs to the stopping time, i.e., $\tau = i$. Naturally, Alice can abort without sending this message to Bob when she finds out her $i^{th}$ message is $(m_i^*, h_i^*)$. This attack causes a deviation of $\Big|X_\tau - D_\tau^{\mathsf{B}}\Big|$. On the other hand, Bob can also attack by aborting when he receives Alice's message $(m_i^*, h_i^*)$. This attack ensures a deviation of $\Big|X_\tau - D_{\tau+1}^{\mathsf{A}}\Big|$. Note that for the $(i+1)^{th}$ message, Alice is not supposed to speak, her $(i+1)^{th}$ defense is exactly her $i^{th}$ defense. Hence this deviation can be also written as $\Big|X_\tau - D_\tau^{\mathsf{A}}\Big|$.

3. The above argument has a boundary case, which is the last message of the protocol. Suppose Alice sends the last message. Then, Bob, who receives this message, cannot

abort anymore because the protocol has ended. Therefore, if our stopping time $\tau = n$, the score function must exclude $\left|X_\tau - D_\tau^{\mathsf{A}}\right|$. This explains why we have the indicator function $\mathbb{1}$ in our score function.

4. Lastly, we illustrate how one can translate this score function into a fail-stop attack strategy. Suppose we find a stopping time $\tau^*$ that witnesses a large score $\mathsf{Score}(\pi, \tau^*)$. For Alice, we will partition the stopping time into two partitions depending on whether $X_\tau \geqslant D_\tau^{\mathsf{B}}$ or not. Similarly, for Bob, we partition the stopping time into two partitions depending on whether $X_\tau \geqslant D_\tau^{\mathsf{A}}$. These four attack strategies correspond to Alice or Bob deviating towards 0 or 1. And the summation of the deviations caused by these four attacks are exactly $\mathsf{Score}(\pi, \tau^*)$. Hence, there must exist a fail-stop attack strategy for one of the parties that changes the honest party's output distribution by $\geqslant \frac{1}{4} \cdot \mathsf{Score}(\pi, \tau^*)$.

Given the definition of our score function, we are interested in finding the stopping time that witnesses the largest score. This motivates the following definition.

**Definition 5.3.2.** *For any coin-tossing protocol $\pi$, define*

$$\mathsf{Opt}(\pi) := \max_{\tau} \mathsf{Score}(\pi, \tau).$$

Intuitively, $\mathsf{Opt}(\pi)$ represents the susceptibility of the protocol $\pi$. And by our discussion above, protocol $\pi$ is at least $\frac{1}{4} \cdot \mathsf{Opt}(\pi)$-unfair.

### 5.3.1 Inductive Proof Strategy

Let $\pi$ be any $r$-message protocol such that the expect output is $X_0$. We shall inductively prove that

$$\mathsf{Opt}(\pi) \geqslant \Gamma_r \cdot X_0(1 - X_0)$$

based on the message complexity $r$. Our proof makes uses of the following potential function.

**Definition 5.3.3** (Our Potential Function)**.** *For any $x, a, b \in [0, 1]$, define*

$$\Phi(x, a, b) := x(1 - x) + (x - a)^2 + (x - b)^2.$$

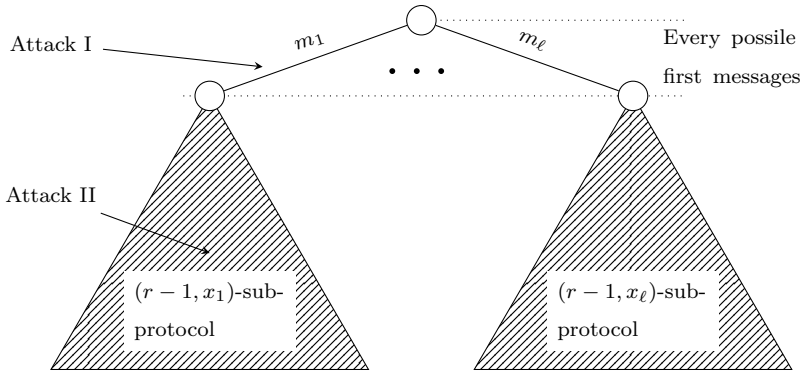



**Figure 5.2.** Intuition for the inductive proof.

*In particular, observe that*

$$\Phi(x, a, b) = x + (x - a - b)^2 - 2ab.$$

The following remarks provide perspective to our potential function.

- The term $x(1-x)$ is the susceptibility due to the expected output of the protocol. Intuitively, if the expected output is 0 or 1 then the protocol should be perfectly secure.
- The term $(x-a)^2$ is the penalty term that penalizes Alice if her defense is too far away from the expected output of the protocol.
- Analogously, the term $(x-b)^2$ is the penalty term for Bob.

Now, let us see how one inductively prove that

$$\mathsf{Opt}(\pi) \geqslant \Gamma_r \cdot X_0(1 - X_0).$$

For every possible first message of this protocol, we consider two types of attacks (refer to Figure 5.2). First, parties can attack by immediately abort upon this first message. Second, parties can defer their attack to the remaining sub-protocol, which has only $r-1$ messages. Suppose when the first message is $m_i$, the remaining sub-protocol has expected output $x_i$.

Additionally, the expectation of Alice and Bob defense is $a_i$ and $b_i$. The effectiveness of the first attack is precisely (according to our score function)

$$|x_i - a_i| + |x_i - b_i|,$$

where $|x_i - a_i|$ is the change of Alice's output if Bob aborts, and analogously, $|x_i - b_i|$ is the change of Bob's output if Alice aborts. On the other hand, by the inductive hypothesis, we know the effectiveness of the second attack is at least

$$\Gamma_{r-1} \cdot x_i(1 - x_i).$$

Now, Lemma 5.2.1 shows that the maximum of these two quantities is lower bounded by

$$\Gamma_r \cdot \Phi(x_i, a_i, b_i).$$

So, over all possible first messages,

$$\mathsf{Opt}(\pi) \geqslant \mathop{\mathrm{E}}_i \left[\Gamma_r \cdot \Phi(x_i, a_i, b_i)\right].$$

Now, observe that if Jensen's inequality holds, i.e.,

$$\mathop{\mathrm{E}}_i \left[\Phi(x_i, a_i, b_i)\right] \geqslant \Phi\left(\mathop{\mathrm{E}}_i \left[x_i\right], \mathop{\mathrm{E}}_i \left[a_i\right], \mathop{\mathrm{E}}_i \left[b_i\right]\right), \tag{5.1}$$

then the proof is complete since we have

$$\begin{aligned}
&\mathop{\mathrm{E}}_i \left[\Gamma_r \cdot \Phi(x_i, a_i, b_i)\right] \\
&\geqslant \Gamma_r \cdot \Phi\left(\mathop{\mathrm{E}}_i \left[x_i\right], \mathop{\mathrm{E}}_i \left[a_i\right], \mathop{\mathrm{E}}_i \left[b_i\right]\right) \\
&= \Gamma_r \cdot \Phi(X_0, D_0^{\mathsf{A}}, D_0^{\mathsf{B}}) \\
&\geqslant \Gamma_r \cdot X_0(1 - X_0)
\end{aligned}$$

Unfortunately, $\Phi(x, a, b)$ is *not* a convex function. However, as we observe in Definition 5.3.3, $\Phi(x, a, b)$ could be rewritten as

$$\Phi(x, a, b) = x + (x - a - b)^2 - 2ab.$$

Note that $x$ and $(x - a - b)^2$ are convex functions, and, hence, Jensen's inequality holds. The only problematic term is $ab$. To resolve this, all we need is the following guarantee.

| Conditioned on the partial transcript,<br>Alice private view and Bob private view are (close to) independent. |
|---|

Then we shall have $\mathop{\mathrm{E}}_i [a_i b_i] \approx \mathop{\mathrm{E}}_i [a_i] \mathop{\mathrm{E}}_i [b_i]$,[1] and, hence, Equation 5.1 shall (approximately) hold and the proof is done.

Therefore, all that is left to do is to ensure that Alice and Bob private views are (close to) independent in the respective model.

- For black-box separation from one-way functions, we consider the information-theoretic random oracle model. The existing techniques [BM09, MMP14] is sufficient to show that by asking polynomially many additional queries, one could ensure this guarantee. This result and the additional subtleties are presented in Section 5.4.

- For $f$-hybrid model, the characterization of Kilian [Kil00] shows that either $f$ is sufficient to imply oblivious transfer protocol, or Alice and Bob private views are always (perfectly) independent conditioned on an arbitrary number of realizations of $f$. This result and the additional subtleties are presented in Section 5.5.

- For black-box separation from public-key encryptions, prior work [MMP14] also shows that one could ask polynomially many additional queries to ensure this guarantee. This result and the additional subtleties are presented in Section 5.6.

[1] ↑In particular, if Alice private view and Bob private view are *perfectly* independent, we shall have $\mathop{\mathrm{E}}_i [a_i b_i] = \mathop{\mathrm{E}}_i [a_i] \mathop{\mathrm{E}}_i [b_i]$.

## 5.4 Separation from One-way function

### 5.4.1 Two-party interactive protocols in the random oracle model

Alice and Bob speak in alternate rounds. We denote the $i^{th}$ message by $M_i$. For every message $M_i$, we denote Alice's private view immediately after sending/receiving message $M_i$ as $V_i^{\mathsf{A}}$, which consists of Alice's random tape $R^{\mathsf{A}}$, her private queries, and the first $i$ messages exchanged. We use $V_0^{\mathsf{A}}$ to represent Alice's private view before the protocol begins. Similarly, we define Bob's private view $V_i^{\mathsf{B}}$ and use $R^{\mathsf{B}}$ to denote his private random tape.

**Query Operator $\mathcal{Q}$.** For any view $V$, we use $\mathcal{Q}(V)$ to denote the set of all queries contained in the view $V$.

### 5.4.2 Heavy Querier and the Augmented Protocol

For two-party protocols in the random oracle model, [IR89, BM09] introduced a standard algorithm, namely, the *heavy querier*. We shall use the following imported theorem.

**Imported Theorem 5.4.1** (Guarantees of Heavy Querier [BM09, MMP14])**.** *Let $\pi$ be any two-party protocol between Alice and Bob in the random oracle model, in which both parties ask at most $n$ queries. For all threshold $\varepsilon \in (0,1)$, there exists a public algorithm, called the* heavy querier*, who has access to the transcript between Alice and Bob. After receiving each message $M_i$, the heavy querier performs a sequence of queries and obtains its corresponding answers from the random oracle. Let $H_i$ denote the sequence of query-answer pairs asked by the heavy querier after receiving message $M_i$. Let $T_i$ be the union of the $i^{th}$ message $M_i$ and the $i^{th}$ heavy querier message $H_i$. The heavy querier guarantees that the following conditions are simultaneously satisfied.*

- ***$\varepsilon$-Lightness.*** *For any $i$, any $t_{\leqslant i} \in \mathrm{Supp}(T_{\leqslant i})$, and query $q \notin \mathcal{Q}\left(h_{\leqslant i}\right)$,*

$$\Pr\left[q \in \mathcal{Q}\left(V_i^{\mathsf{A}} \middle| T_{\leqslant i} = t_{\leqslant i}\right)\right] \leqslant \varepsilon, \quad \text{and} \quad \Pr\left[q \in \mathcal{Q}\left(V_i^{\mathsf{B}} \middle| T_{\leqslant i} = t_{\leqslant i}\right)\right] \leqslant \varepsilon.$$

- $n\varepsilon$***-Dependence.*** *Fix any $i$,*

$$\mathop{\mathrm{E}}_{t_{\leqslant i} \leftarrow T_{\leqslant i}} \left[ \mathsf{SD}\left( \left( V_i^{\mathsf{A}}, V_i^{\mathsf{B}} \middle| T_{\leqslant i} = t_{\leqslant i} \right), \left( V_i^{\mathsf{A}} \middle| T_{\leqslant i} = t_{\leqslant i} \right) \times \left( V_i^{\mathsf{B}} \middle| T_{\leqslant i} = t_{\leqslant i} \right) \right) \right] \leqslant n\varepsilon.$$

  *Intuitively, it states that* on average*, the statistical distance between (1) the joint distribution of Alice's and Bob's private view, and (2) the product of the marginal distributions of Alice's private views and Bob's private views is small.*

- $\mathcal{O}(n/\varepsilon)$***-Efficiency.*** *The expected number of queries asked by the heavy querier is bounded by $\mathcal{O}(n/\varepsilon)$. Consequently, it has $\mathcal{O}(n/\varepsilon^2)$ query complexity with probability (at least) $(1-\varepsilon)$ by an averaging argument.*

We refer to the protocol with the heavy querier's messages attached as the *augmented protocol*. We call $T_i$ the augmented message.

### 5.4.3 Coin-Tossing Protocol in the Random Oracle Model

We will prove our main result by induction on the message complexity of the protocol. Therefore, after any partial transcript $t_{\leqslant i}$, we will treat the remainder of the orginal protocol starting from the $(i+1)^{th}$ message, as a protocol of its own. Hence, it is helpful to define the coin-tossing protocol where, before the beginning of the protocol, Alice's and Bob's private views are already correlated with the random oracle. However, note that, in the augmented protocol, after each augmented message $t_i$, the heavy querier has just ended. Thus, these correlations will satisfy Imported Theorem 5.4.1. Therefore, we need to define a general class of coin-tossing protocols in the random oracle model over which we shall perform our induction.

**Definition 5.4.1** ($(\varepsilon, \vec{\alpha}, r, n, X_0)$-Coin-Tossing)**.** *An interactive protocol $\pi$ between Alice and Bob with random oracle $O : \{0,1\}^\lambda \to \{0,1\}^\lambda$ is called an* $(\varepsilon, \vec{\alpha}, r, n, X_0)$-coin-tossing protocol *if it satisfies the following.*

- ***Setup.*** *There is an arbitrary set $\mathcal{S} \subseteq \{0,1\}^\lambda$, which is publicly known, such that for all queries $s \in \mathcal{S}$, the query answers $O(s)$ are also publicly known. Let $\Omega^{\mathsf{A}}$, $\Omega^{\mathsf{B}}$, and*

$\Omega^{\mathsf{O}}$ *be the universes of Alice's random tape, Bob's random tape, and the random oracle, respectively. There are also publicly known sets* $\mathcal{A} \subseteq \Omega^{\mathsf{A}} \times \Omega^{\mathsf{O}}$ *and* $\mathcal{B} \subseteq \Omega^{\mathsf{B}} \times \Omega^{\mathsf{O}}$*. The random variables* $R^{\mathsf{A}}$*,* $R^{\mathsf{B}}$*, and* $O$ *are sampled uniformly conditioned on that (1)* $(R^{\mathsf{A}}, O) \in \mathcal{A}$*, (2)* $(R^{\mathsf{B}}, O) \in \mathcal{B}$*, and (3)* $O$ *is consistent with the publicly known answers at* $\mathcal{S}$*. Alice's private view before the beginning of the protocol is a deterministic function of* $R^{\mathsf{A}}$ *and* $O$*, which might contain private queries. Likewise, Bob's private view is a deterministic function of* $R^{\mathsf{B}}$ *and* $O$*.*[2]

- ***Agreement.*** *At the end of the protocol, both parties always agree on the output* $\in \{0, 1\}$*. Without loss of generality, we assume the output is concatenated to the last message in the protocol.*[3]

- ***Defense preparation.*** *At message* $M_i$*, if Alice is supposed to speak, in addition to preparing the next-message* $M_i$*, she will also prepare a defense coin for herself as well. If Bob decides to abort the next message, she shall* not *make any additional queries to the random oracle, and simply output the defense she has just prepared. [DLMM11, DMM14] introduced this constraint as the "instant construction." They showed that, without loss of generality, one can assume this property for all the defense preparations except for the first defense (see Remark 5.4.2). We shall refer to this defense both as Alice's* $i^{th}$ *defense and also as her* $(i+1)^{th}$ *defense. Consequently, Alice's defense for every* $i$ *is well-defined. Bob's defense is defined similarly. We assume the party who receives the first message has already prepared her defense for the first message before the protocol begins.*

- $\varepsilon$***-Lightness at Start.*** *For any query* $q \notin \mathcal{S}$*, the probability that Alice has asked query* $q$ *before the protocol begins is upper bounded by* $\varepsilon \in [0, 1]$*. Similarly, the probability that Bob has asked query* $q$ *is at most* $\varepsilon$*.*

[2]↑ Basically, $\mathcal{S}$ is the set of all the queries that the heavy querier has published. $\mathcal{A}$ is the set of all possible pairs of Alice's private randomness $r^{\mathsf{A}}$ and random oracle $o$ that are consistent with Alice's messages before this protocol begins. Similarly, $\mathcal{B}$ is the set of all consistent pairs of Bob's private randomness $r^{\mathsf{B}}$ and random oracle $o$.

[3]↑This generalization shall not make the protocol any more vulnerable. Any attack in this protocol shall also exist in the original protocol with the same amount of deviation. This only helps simplify the presentation of our proof.

- $\vec{\alpha}$***-Dependence.*** *For all $i \in \{0, 1, \ldots, r\}$, Alice's and Bob's private views are $\alpha_i$-dependent on average immediate after the message $T_i$. That is, the following condition is satisfied for every $i$.*

$$\alpha_i := \mathop{\mathrm{E}}_{t_{\leqslant i} \leftarrow T_{\leqslant i}} \left[ \mathsf{SD}\left( \left( V_i^{\mathsf{A}}, V_i^{\mathsf{B}} \middle| T_{\leqslant i} = t_{\leqslant i} \right), \left( V_i^{\mathsf{A}} \middle| T_{\leqslant i} = t_{\leqslant i} \right) \times \left( V_i^{\mathsf{B}} \middle| T_{\leqslant i} = t_{\leqslant i} \right) \right) \right]$$

- $r$***-Message complexity.*** *The number of messages of this protocol is $r = \mathsf{poly}(\lambda)$. We emphasize that the length of the message could be arbitrarily long.*

- $n$***-Query complexity.*** *For all possible complete executions of the protocol, the number of queries that Alice asks (including the queries asked before the protocol begins) is at most $n = \mathsf{poly}(\lambda)$. This also includes the queries that are asked for the preparation of the defense coins. Likewise, Bob asks at most $n$ queries as well.*

- $X_0$***-Expected Output.*** *The expectation of the output is $X_0 \in (0, 1)$.*

**Remark 5.4.1.** *Let us justify the necessity of $\vec{\alpha}$-dependence in the definition. We note that when heavy querier stops, Alice's and Bob's view are not necessarily close to the product of their respective marginal distributions.*[4] *However, to prove any meaningful bound on the susceptibility of this protocol, we have to treat $\vec{\alpha}$ as an additional error term. Therefore, we introduce this parameter in our definition. However, the introduction of this error shall not be a concern globally, because the heavy querier guarantees that over all possible executions this dependence is at most $n\varepsilon$ (on average), which we shall ensure to be sufficiently small.*

**Remark 5.4.2.** *We note that, after every heavy querier message, the remaining sub-protocol always satisfies the definition above. However, the original coin-tossing protocol might not meet these constraints. For example, consider a one-message protocol where Alice queries $O(0^\lambda)$, and sends the parity of this string to Bob as the output. On the other hand, Bob also queries $O(0^\lambda)$ and uses the parity of this string as his defense. This protocol is perfectly secure in the sense that no party can deviate the output of the protocol at all. However,*

[4]↑For instance, suppose Alice samples a uniform string $u_1 \leftarrow \{0,1\}^\lambda$ and sends $O(u_1)$ to Bob. Next, Bob samples a uniform string $u_2 \leftarrow \{0,1\}^\lambda$ and sends $O(u_2)$ to Alice. Assume the first message and the second message are the same, i.e., $O(u_1) = O(u_2)$. Then, there are no heavy queries, but Alice's and Bob's private views are largely correlated.

*the query $0^\lambda$ is 1-heavy in Bob's private view even before the protocol begins. Prior works [DLMM11, DMM14] rule out such protocols by banning Bob from making any queries when he prepares his first defense. In this work, we consider protocols such that no queries are more than $\varepsilon$-heavy when Bob prepares his first defense. We call this the* $\varepsilon$-lightness at start assumption. *The set of protocols that prior works consider is identical to the set of protocols that satisfies 0-lightness at start assumption.*

*To justify our $\varepsilon$-lightness at start assumption, we observe that one can always run a heavy querier with a threshold $\varepsilon$ before the beginning of the protocol as a pre-processing step. Note that this step fixes only a small part (of size $\mathcal{O}(n/\varepsilon)$) of the random oracle, and, hence, the random oracle continues to be an "idealized" one-way function. If this protocol is a black-box construction of a coin-tossing protocol with any one-way function, the choice of the one-way function should not change its expected output. Therefore, by running a heavy querier before the beginning of the protocol, it should not alter the expected output of the protocol. After this compilation step, all queries are $\varepsilon$-light in Bob's view* before *the protocol begins. Consequently, our inductive proof technique is applicable.*

**Remark 5.4.3.** *Let us use the an example to further illustrate how we number Alice's and Bob's defense coins. Suppose Alice sends the first message in the protocol. Bob shall prepare his first defense coin even before the protocol begins. Alice, during her preparation of the first message, shall also prepare a defense coin as her first defense.*

*The second message in the protocol is sent by Bob. Since Alice is not speaking during this message preparation, her second defense coin remains identical to her first defense coin. Bob, on the other hand, shall update a new defense coin as his second defense during his preparation of the second message.*

*For the third message, Alice shall prepare a new third defense coin and Bob's third defense coin is identical to his second defense coin. This process continues for $r$ messages during the protocol execution.*

**Notation.** Let $X_i$ represent the expected output conditioned on the first $i$ augmented messages, i.e., the random variable $T_{\leqslant i}$. Let $D_i^{\mathsf{A}}$ be the expectation of Alice's $i^{th}$ defense coin conditioned on the first $i$ augmented messages. Similarly, let $D_i^{\mathsf{B}}$ be the expectation of Bob's

$i^{th}$ defense coin conditioned on the first $i$ augmented messages. (Refer to Definition 5.4.1 for the definition of $i^{th}$ defense. Recall that, for both Alice and Bob, the $i^{th}$ defense is defined for all $i \in \{1, 2, \ldots, r\}$.) Note that random variables $X_i$, $D_i^{\mathsf{A}}$, and $D_i^{\mathsf{B}}$ are all $T_{\leqslant i}$-measurable.

### 5.4.4 Main Technical Result in the Random Oracle Model

**Theorem 5.4.1.** *For any $(\varepsilon, \vec{\alpha}, r, n, X_0)$-coin-tossing protocol $\pi$, the following holds.*

$$\mathsf{Opt}(\pi) \geqslant \Gamma_r \cdot X_0 (1 - X_0) - \left( nr \cdot \varepsilon + \alpha_0 + 2 \sum_{i=1}^{r} \alpha_i \right).$$

*Furthermore, one needs to make an additional $\mathcal{O}(n/\varepsilon)$ queries to the random oracle (in expectation) to identify a stopping time $\tau$ witnessing this lower bound.*

We defer the proof to Section 5.4.5. In light of the remarks above, this theorem implies the following corollary.

**Corollary 5.4.2.** *Let $\pi$ be a coin-tossing protocol in the random oracle model that satisfies the $\varepsilon$-lightness at start assumption (see Remark 5.4.2). Suppose $\pi$ is an $r$-message protocol, and Alice and Bob ask at most $n$ queries. The expected output of $\pi$ is $X_0$. Then, either Alice or Bob has a fail-stop attack strategy that deviates the honest party's output distribution by*

$$\Omega\left( \frac{X_0 (1 - X_0)}{\sqrt{r}} \right).$$

*This attack strategy performs $\mathcal{O}\left(\frac{n^2 r^2}{X_0(1-X_0)}\right)$ additional queries to the random oracle in expectation.*

This corollary is obtained by substituting $\varepsilon = \frac{X_0(1-X_0)}{nr^2}$ in Theorem 5.4.1. Imported Theorem 5.4.1 guarantees that, for all $i$, the average dependencies after the $i^{th}$ message are bounded by $n\varepsilon$. Hence, the error term is $\mathrm{o}\left(\frac{X_0(1-X_0)}{\sqrt{r}}\right)$.

The efficiency of the heavy querier is guaranteed by Imported Theorem 5.4.1. One can transform the average-case efficiency to worst-case efficiency by forcing the heavy querier to stop when it asks more than $\frac{n^2 r^3}{(X_0(1-X_0))^2}$ queries. By Markov's inequality, this happens

with probability at most $\mathcal{O}\left(\frac{X_0(1-X_0)}{r}\right) = \mathrm{o}\left(\frac{X_0(1-X_0)}{\sqrt{r}}\right)$, and thus the quality of this attack is essentially identical to the averge-case attack.

### 5.4.5 The Proof

In this section, we prove Theorem 5.4.1 using induction on the message complexity $r$.

Our proof relies on the following useful lemma. It is implicit in [BM09] that if (1) Alice's and Bob's private view before the protocol begins are $\alpha_0$-dependent, (2) all the queries are $\varepsilon$-light for Bob, and (3) Alice asks at most $n$ queries to prepare her first message, then after the first message, Alice's and Bob's private view are $(\alpha_0 + n\varepsilon)$-dependent.

**Lemma 5.4.1** (Technical Lemma [BM09])**.** *We have*

$$\mathsf{SD}\left(\left(V_1^{\mathsf{A}}, V_0^{\mathsf{B}}\right), \left(V_1^{\mathsf{A}} \times V_0^{\mathsf{B}}\right)\right) \leqslant \alpha_0 + n\varepsilon.$$

**Base case of the Induction: Message Complexity $r = 1$.**

Let $\pi$ be an $(\varepsilon, \vec{\alpha}, r, n, X_0)$-coin-tossing protocol with $r = 1$. In this protocol, Alice sends the only message $M_1$. We shall pick the stopping time $\tau$ to be 1. Note that this is the last message of the protocol and hence Bob who receives it cannot abort any more. Therefore, our score function is the following

$$\mathsf{Score}(\pi, \tau) = \mathrm{E}\Big[\Big|X_1 - D_1^{\mathsf{B}}\Big|\Big].$$

Let $D_0^{\mathsf{B}} = \mathrm{E}\Big[D_1^{\mathsf{B}}\Big]$, which is the expectation of Bob's first defense before the protocol begins. Recall that in the augmented protocol $T_1 = (M_1, H_1)$, and $X_1$ and $D_1^{\mathsf{B}}$ are $T_1$ measurable. We have

$$\begin{aligned}
\mathrm{E}\Big[\Big|X_1 - D_1^{\mathsf{B}}\Big|\Big] &= \mathop{\mathrm{E}}_{m_1 \leftarrow M_1}\left[\mathop{\mathrm{E}}_{h_1 \leftarrow (H_1 | M_1 = m_1)}\left[\Big|X_1 - D_1^{\mathsf{B}}\Big|\right]\right] \\
&\overset{\text{(i)}}{\geqslant} \mathop{\mathrm{E}}_{m_1 \leftarrow M_1}\left[\Big|\,\mathrm{E}[X_1 | M_1 = m_1] - \mathrm{E}\Big[D_1^{\mathsf{B}} | M_1 = m_1\Big]\Big|\right] \\
&\overset{\text{(ii)}}{\geqslant} \mathop{\mathrm{E}}_{m_1 \leftarrow M_1}\left[\Big|\,\mathrm{E}[X_1 | M_1 = m_1] - D_0^{\mathsf{B}}\Big| - \Big|D_0^{\mathsf{B}} - \mathrm{E}\Big[D_1^{\mathsf{B}} | M_1 = m_1\Big]\Big|\right]
\end{aligned}$$

$$
\begin{aligned}
&\overset{\text{(iii)}}{\geqslant} \mathop{\mathrm{E}}_{m_1 \leftarrow M_1}\left[\left|\,\mathrm{E}[X_1|M_1=m_1] - D_0^{\mathsf{B}}\right|\right] - \alpha_0 - n\varepsilon \\
&\overset{\text{(iv)}}{\geqslant} X_0 \cdot \left(1 - D_0^{\mathsf{B}}\right) + (1-X_0)\cdot D_0^{\mathsf{B}} - \alpha_0 - n\varepsilon \\
&\geqslant X_0\,(1-X_0) + \left(X_0 - D_0^{\mathsf{B}}\right)^2 - \alpha_0 - n\varepsilon \\
&\geqslant X_0\,(1-X_0) - \alpha_0 - n\varepsilon.
\end{aligned}
$$

In the above inequality, (i) and (ii) are because of triangle inequality. Since we assume the output is concatenated to the last message of the protocol, $\mathrm{E}[X_1|M_1=m_1] \in \{0,1\}$. And by the definition of $X_0$, the probability of the output being 1 is $X_0$. Hence we have (iv).

To see (iii), note that

$$
\begin{aligned}
\mathrm{E}\Big[D_1^{\mathsf{B}}\Big|M_1=m_1\Big] &= \sum_{v_1^{\mathsf{A}},v_0^{\mathsf{B}}} \Pr\Big[V_1^{\mathsf{A}}=v_1^{\mathsf{A}}, V_0^{\mathsf{B}}=v_0^{\mathsf{B}}\Big|M_1=m_1\Big]\,\mathrm{E}\Big[D_1^{\mathsf{B}}\Big|V_0^{\mathsf{B}}=v_0^{\mathsf{B}}\Big] \\
&\leqslant \sum_{\mathcal{Q}(v_1^{\mathsf{A}})\cap\mathcal{Q}(v_0^{\mathsf{B}})=\emptyset} \Pr\Big[V_1^{\mathsf{A}}=v_1^{\mathsf{A}}\Big|M_1=m_1\Big]\cdot\Pr\Big[V_0^{\mathsf{B}}=v_0^{\mathsf{B}}\Big]\,\mathrm{E}\Big[D_1^{\mathsf{B}}\Big|V_0^{\mathsf{B}}=v_0^{\mathsf{B}}\Big] \\
&\quad + \sum_{\mathcal{Q}(v_1^{\mathsf{A}})\cap\mathcal{Q}(v_0^{\mathsf{B}})\neq\emptyset} \Pr\Big[V_1^{\mathsf{A}}=v_1^{\mathsf{A}}, V_0^{\mathsf{B}}=v_0^{\mathsf{B}}\Big|M_1=m_1\Big]\,\mathrm{E}\Big[D_1^{\mathsf{B}}\Big|V_0^{\mathsf{B}}=v_0^{\mathsf{B}}\Big]
\end{aligned}
$$

Hence,

$$
\left|\mathrm{E}\Big[D_1^{\mathsf{B}}\Big|M_1=m_1\Big] - D_0^{\mathsf{B}}\right| \leqslant \Pr_{(v_1^{\mathsf{A}},v_0^{\mathsf{B}})\leftarrow(V_1^{\mathsf{A}},V_0^{\mathsf{B}})|M_1=m_1}\left[\mathcal{Q}\left(v_1^{\mathsf{A}}\right)\cap\mathcal{Q}\left(v_0^{\mathsf{B}}\right)\neq\emptyset\right].
$$

Therefore,

$$
\begin{aligned}
&\mathop{\mathrm{E}}_{m_1\leftarrow M_1}\left[\left|\,\mathrm{E}\Big[D_1^{\mathsf{B}}\Big|M_1=m_1\Big] - D_0^{\mathsf{B}}\right|\right] \\
&\leqslant \mathop{\mathrm{E}}_{m_1\leftarrow M_1}\left[\Pr_{(v_1^{\mathsf{A}},v_0^{\mathsf{B}})\leftarrow(V_1^{\mathsf{A}},V_0^{\mathsf{B}})|M_1=m_1}\left[\mathcal{Q}\left(v_1^{\mathsf{A}}\right)\cap\mathcal{Q}\left(v_0^{\mathsf{B}}\right)\neq\emptyset\right]\right] \\
&\leqslant \Pr_{(v_1^{\mathsf{A}},v_0^{\mathsf{B}})\leftarrow(V_1^{\mathsf{A}},V_0^{\mathsf{B}})}\left[\mathcal{Q}\left(v_1^{\mathsf{A}}\right)\cap\mathcal{Q}\left(v_0^{\mathsf{B}}\right)\neq\emptyset\right] \leqslant \alpha_0 + n\varepsilon.
\end{aligned}
$$

This completes the proof for the base case.

**Inductive Step.**

Suppose the theorem is true for $r = r_0 - 1$, we are going to prove it for $r = r_0$. Let $\pi$ be an arbitrary $(\varepsilon, \vec{\alpha}, r_0, n, X_0)$-coin-tossing protocol. Assume the first augmented message is $(M_1, H_1) = (m_1^*, h_1^*)$, and conditioned on that, $X_1 = x_1^*$, $D_1^{\mathsf{A}} = d_1^{\mathsf{A},*}$, and $D_1^{\mathsf{B}} = d_1^{\mathsf{B},*}$. Moreover, the remaining sub-protocol $\pi^*$ is an $(\varepsilon, \vec{\alpha}^*, r_0 - 1, n, x_1^*)$-coin-tossing protocol. By our induction hypothesis,

$$\mathsf{Opt}\left(\pi^*\right) \geqslant \Gamma_{r_0-1} \cdot x_1^*\left(1 - x_1^*\right) - \left(n(r_0-1)\varepsilon + \alpha_0^* + \sum_{i=1}^{r_0-1} \alpha_i^*\right).$$

(For simplicity, we shall use $\mathsf{Err}\left(\vec{\alpha}, n, r\right)$ to represent $\alpha_0 + \sum_{i=1}^{r} \alpha_i + nr\varepsilon$ in the rest of the proof.) That is, there exists a stopping time $\tau^*$ for sub-protocol $\pi^*$, whose score is lower bounded by the quantity above. On the other hand, we may choose not to continue by picking this message $(M_1, H_1) = (m_1^*, h_1^*)$ as our stopping time. This would yield a score of

$$\left|x_1^* - d_1^{\mathsf{A},*}\right| + \left|x_1^* - d_1^{\mathsf{B},*}\right|.$$

Hence, the optimal stopping time would decide on whether to abort now or defer the attack to sub-protocol $\pi^*$ by comparing which one of those two quantities is larger. This would yield a score of

$$\begin{aligned}
&\max\left(\mathsf{Opt}\left(\pi^*\right)\ ,\ \left|x_1^* - d_1^{\mathsf{A},*}\right| + \left|x_1^* - d_1^{\mathsf{B},*}\right|\right) \\
\geqslant\ &\max\left(\Gamma_{r_0-1} \cdot x_1^*\left(1 - x_1^*\right)\ ,\ \left|x_1^* - d_1^{\mathsf{A},*}\right| + \left|x_1^* - d_1^{\mathsf{B},*}\right|\right) - \mathsf{Err}\left(\vec{\alpha}^*, n, r_0 - 1\right) \\
\overset{\text{(i)}}{\geqslant}\ &\Gamma_{r_0} \cdot \Phi\left(x_1^*, d_1^{\mathsf{A},*}, d_1^{\mathsf{B},*}\right) - \mathsf{Err}\left(\vec{\alpha}^*, n, r_0 - 1\right),
\end{aligned}$$

where inequality (i) is because of Lemma 5.2.1. Now that we have a lower bound on how much score we can yield at every first augmented message, we are interested in how much they sum up to.

Without loss of generality, assume there are totally $\ell$ possible first augmented messages, namely $t_1^{(1)}, t_1^{(2)}, \ldots, t_1^{(\ell)}$. The probability of the first message being $t_1^{(i)}$ is $p^{(i)}$ and conditioned

that, $X_1 = x_1^{(i)}$, $D_1^{\mathsf{A}} = d_1^{\mathsf{A},(i)}$, and $D_1^{\mathsf{B}} = d_1^{\mathsf{B},(i)}$. Moreover, the remaining $r_0 - 1$ protocol has dependence vector $\vec{\alpha}^{(i)}$. Therefore, we are interested in,

$$\sum_{i=1}^{\ell} p^{(i)} \left( \Gamma_{r_0} \cdot \Phi \left( x_1^{(i)}, d_1^{\mathsf{A},(i)}, d_1^{\mathsf{B},(i)} \right) - \mathsf{Err} \left( \vec{\alpha}^{(i)}, n, r_0 - 1 \right) \right)$$

Recall that our potential function $\Phi$ satisfies

$$\Phi(x, a, b) = x + (x - a - b)^2 - 2ab.$$

Therefore, we can rewrite the above quantity as

$$\sum_{i=1}^{\ell} p^{(i)} \left( \Gamma_{r_0} \left( x_1^{(i)} + \left( x_1^{(i)} - d_1^{\mathsf{A},(i)} - d_1^{\mathsf{B},(i)} \right)^2 - 2 \cdot d_1^{\mathsf{A},(i)} \cdot d_1^{\mathsf{B},(i)} \right) - \mathsf{Err} \left( \vec{\alpha}^{(i)}, n, r_0 - 1 \right) \right)$$

We observe the following case analysis for the three expressions in the potential function above.

1. For the $x$ term, we observe that the expectation of $x_1^{(i)}$ is $X_0$, i.e., we have $\sum_{i=1}^{\ell} p^{(i)} \cdot x_1^{(i)} = X_0$.
2. For the $(x-y-z)^2$ term, we note that it is a convex tri-variate function. Hence, Jensen's inequality is applicable.
3. For the $y \cdot z$ term, we have the following claim.

**Claim 5.4.1** (Global Invariant)**.**

$$\left| \sum_{i=1}^{\ell} p^{(i)} \cdot d_1^{\mathsf{A},(i)} \cdot d_1^{\mathsf{B},(i)} - \mathrm{E}\left[ D_1^{\mathsf{A}} \right] \mathrm{E}\left[ D_1^{\mathsf{B}} \right] \right| \leqslant (\alpha_0 + n\varepsilon) + \alpha_1.$$

*Proof.* To see this, consider the expectation of the product of Alice and Bob defense when we sample from $\left( V_1^{\mathsf{A}}, V_0^{\mathsf{B}} \right)$. This expectation is $\alpha_0 + n\varepsilon$ close to $\mathrm{E}\left[ D_1^{\mathsf{A}} \right] \mathrm{E}\left[ D_1^{\mathsf{B}} \right]$ because joint distribution $\left( V_1^{\mathsf{A}}, V_0^{\mathsf{B}} \right)$ is $\alpha_0 + n\varepsilon$ close to the product of its marginal distribution by Lemma 5.4.1.

On the other hand, this expectation is identical to the average (over all possible messages) of the expectation of the product of Alice and Bob defense when we sample from

$\left(V_1^{\mathsf{A}}, V_0^{\mathsf{B}} \middle| T_1 = t_1^{(i)}\right)$. Conditioned on first message being $t_1^{(i)}$, this expectation is $\alpha_0^{(i)}$-close to $d_1^{\mathsf{A},(i)} \cdot d_1^{\mathsf{B},(i)}$ because $\left(V_1^{\mathsf{A}}, V_0^{\mathsf{B}} \middle| T_1 = t_1^{(i)}\right)$ has $\alpha_0^{(i)}$-dependence by definition.

Finally, we note that, by definition, $\sum_{i=1}^{\ell} p^{(i)} \alpha_0^{(i)} = \alpha_1$. Note that the indices between $\alpha$ and $\alpha^{(i)}$ are shifted by 1. This is because of that the dependence after the first message of the original protocol is the average of the dependence before each sub-protocol begins.

This proves that $\sum_{i=1}^{\ell} p^{(i)} \cdot d_1^{\mathsf{A},(i)} d_1^{\mathsf{B},(i)}$ and $\mathrm{E}\left[D_1^{\mathsf{A}}\right] \mathrm{E}\left[D_1^{\mathsf{B}}\right]$ are $(\alpha_0 + n\varepsilon) + \alpha_1$ close. □

Given these observations, we can push the expectation inside each term, and they imply that our score is lower bounded by

$$\Gamma_{r_0}\Big(X_0 + \left(X_0 - \mathrm{E}\left[D_1^{\mathsf{A}}\right] - \mathrm{E}\left[D_1^{\mathsf{B}}\right]\right)^2 - 2 \cdot \mathrm{E}\left[D_1^{\mathsf{A}}\right] \cdot \mathrm{E}\left[D_1^{\mathsf{B}}\right] - (\alpha_0 + \alpha_1 + n\varepsilon)\Big) - \sum_{i=1}^{\ell} p^{(i)} \cdot \mathsf{Err}\left(\vec{\alpha}^{(i)}, n, r_0 - 1\right)$$

We note that by definition (again note that the indices of $\alpha$ and $\alpha^{(i)}$ are shifted by 1),

$$(\alpha_0 + \alpha_1 + n\varepsilon) + \sum_{i=1}^{\ell} p^{(i)} \cdot \mathsf{Err}\left(\vec{\alpha}^{(i)}, n, r_0 - 1\right) = \mathsf{Err}\left(\vec{\alpha}, n, r_0\right).$$

Therefore, our score is at least

$$\Gamma_{r_0}\left(X_0 + \left(X_0 - \mathrm{E}\left[D_1^{\mathsf{A}}\right] - \mathrm{E}\left[D_1^{\mathsf{B}}\right]\right)^2 - 2 \cdot \mathrm{E}\left[D_1^{\mathsf{A}}\right] \cdot \mathrm{E}\left[D_1^{\mathsf{B}}\right]\right) - \mathsf{Err}\left(\vec{\alpha}, n, r_0\right).$$

This is lower-bounded by

$$\begin{aligned} & \Gamma_{r_0} \cdot \Phi\left(X_0, \mathrm{E}\left[D_1^{\mathsf{A}}\right], \mathrm{E}\left[D_1^{\mathsf{B}}\right]\right) - \mathsf{Err}\left(\vec{\alpha}, n, r_0\right) \\ \geqslant\ & \Gamma_{r_0} \cdot X_0\left(1 - X_0\right) - \mathsf{Err}\left(\vec{\alpha}, n, r_0\right) \\ =\ & \Gamma_{r_0} \cdot X_0\left(1 - X_0\right) - \left(n r_0 \varepsilon + \alpha_0 + 2\sum_{i=1}^{r_0} \alpha_i\right). \end{aligned}$$

This completes the proof of the inductive step and, hence, the proof of Theorem 5.4.1.

## 5.5 Separation from Idealized $f$-hybrid

Let $f\colon \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$ be an arbitrary (possibly randomized) function. As standard in the literature, we shall restrict to $f$ such that the input domain $\mathcal{X}$ and $\mathcal{Y}$ and the range $\mathcal{Z}$ are of constant size. A two-party protocol in the $f$-hybrid model is defined as follows.

**Definition 5.5.1** ($f$-hybrid Model [Can00, Lin17])**.** *A protocol between Alice and Bob in the $f$-hybrid model is identical to a protocol in the plain model except that both parties have access to a trusted party realizing $f$. At any point during the execution, the protocol specifies which party is supposed to speak.*

- ***Alice/Bob message.*** *If Alice is supposed to speak, she shall prepare her next message as a deterministic function of her private randomness and the partial transcript. If Bob is supposed to speak, his message is prepared in a similar manner.*
- ***Trusted party message.*** *At some point during the execution, the protocol might specify that the trusted party shall speak next. In this case, the protocol shall also specify a natural number $\ell$, which indicates how many instances of $f$ should the trusted party compute. Alice (resp., Bob) will prepare her inputs $\vec{x} = (x_1, \ldots, x_\ell)$ (resp., $\vec{y} = (y_1, \ldots, y_\ell)$) and send it privately to the trusted party. The trusted party shall compute $(f(x_1, y_1), \ldots, f(x_\ell, y_\ell))$ and send it as the next message.*

Next, we define fair coin-tossing protocols in the $f$-hybrid model.

**Definition 5.5.2** (Fair Coin-tossing in the $f$-hybrid Model)**.** *An $(X_0, r)$-fair coin-tossing in the $f$-hybrid model is a two-party protocol between Alice and Bob in the $f$-hybrid model such that it satisfies the following.*

- ***$X_0$-Expected Output.*** *At the end of the protocol, parties always agree on the output $\in \{0, 1\}$ of the protocol. The expectation of the output of an honest execution is $X_0 \in (0, 1)$.*
- ***$r$-Message Complexity.*** *The total number of messages of the protocol is (at most) $r$. This includes both the Alice/Bob message and the trusted party message.*

- ***Defense Preparation.*** *Anytime a party speaks, she shall also prepare a defense coin based on her private randomness and the partial transcript. Her latest defense coin shall be her output when the other party decides to abort. To ensure that parties always have a defense to output, they shall prepare a defense before the protocol begins.*

- ***Insecurity.*** *The insecurity is defined as the maximum change a fail-stop adversary can cause to the expectation of the other party's output.*

For any (randomized) functionality $f$, Kilian [Kil00] proved that if $f$ *does not* satisfy the following cross product rule, $f$ is complete for information-theoretic semi-honest adversaries. That is, for any functionality $g$, there is a protocol in the $f$-hybrid model that realizes $g$, which is secure against information-theoretic semi-honest adversaries. In particular, this implies that there is a protocol in the $f$-hybrid model that realizes oblivious transfer.

**Definition 5.5.3** (Cross Product Rule)**.** *A (randomized) functionality $f\colon \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$ is said to satisfy the cross product rule if for all $x_0, x_1 \in \mathcal{X}$, $y_0, y_1 \in \mathcal{Y}$, and $z \in \mathcal{Z}$ such that*

$$\Pr[f(x_0, y_0) = z] > 0 \quad \textit{and} \quad \Pr[f(x_1, y_0) = z] > 0,$$

*we have*

$$\Pr[f(x_0, y_0) = z] \cdot \Pr[f(x_1, y_1) = z] = \Pr[f(x_1, y_0) = z] \cdot \Pr[f(x_0, y_1) = z].$$

We recall the MNS protocol by Moran, Naor, and Segev [MNS09]. The MNS protocol makes black-box uses of the oblivious transfer as a subroutine to construct optimal-fair coin-tossing protocols. In particular, their protocol enjoys the property that any fail-stop attack during the oblivious transfer subroutine is an entirely ineffective attack. Therefore, the MNS protocol, combined with the results of Kilian [Kil00], gives us the following theorem.

**Theorem 5.5.1** ([Kil00, MNS09])**.** *Let $f$ be a (randomized) functionality that is complete. For any $X_0 \in (0, 1)$ and $r \in \mathbb{N}^*$, there is an $(X_0, r)$-fair coin-tossing protocol in the $f$-hybrid model that is (at most) $\mathcal{O}(1/r)$-insecure against fail-stop attackers.*

**Remark 5.5.1** (On the necessity of the unfairness of $f$)**.** *We emphasize that it is necessary that in the $f$-hybrid model, $f$ is realized* unfairly*. That is, the adversary receives the output of $f$ before the honest party does. If $f$ is realized fairly, i.e., both parties receive the output simultaneously, it is possible to construct perfectly-secure fair coin-tossing. For instance, let $f$ be the* XOR *function. Consider the protocol where Alice samples $x \leftarrow \{0,1\}$, Bob samples $y \leftarrow \{0,1\}$, and the trusted party broadcast $f(x,y)$, which is the final output of the protocol. Trivially, one can verify that this protocol is perfectly-secure.*

Intuitively, the results of Kilian [Kil00] and Moran, Naor, and Segev [MNS09] showed that when $f$ is a functionality that does not satisfy the cross product rule, a secure protocol realizing $f$ can be used to construct optimal-fair coin-tossing.

In this section, we complement the above results by showing that when $f$ is a functionality that does satisfy the cross product rule, a fair coin-tossing protocol in the $f$-hybrid model is (qualitatively) as insecure as a fair coin-tossing protocol in the information-theoretic model. In other words, $f$ is completely useless for fair coin-tossing. Our results are summarized as the following theorem.

**Theorem 5.5.2** (Main Theorem for $f$-hybrid)**.** *Let $f$ be a randomized functionality that is* not *complete. Any $(X_0, r)$-fair coin-tossing protocol in the $f$-hybrid model is (at least) $\Omega\left(\frac{X_0(1-X_0)}{\sqrt{r}}\right)$-insecure.*

### 5.5.1 Properties of Functionalities

To prove Theorem 5.5.2, we start with some observations on functionality that satisfies the cross product rule.

Let $f$ be a functionality that satisfies the cross product rule. We start by observing some properties of $f$. Firstly, let us recall the following definition.

**Definition 5.5.4** (Function Isomorphism [MPR09])**.** *Let $f\colon \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$ and $g\colon \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}'$ be any two (randomized) functionalities. We say $f \leqslant g$ if there exist deterministic mappings $M_{\mathsf{A}}\colon \mathcal{X} \times \mathcal{Z}' \to \mathcal{Z}$ and $M_{\mathsf{B}}\colon \mathcal{Y} \times \mathcal{Z}' \to \mathcal{Z}$ such that, for all $x \in \mathcal{X}$, $y \in \mathcal{Y}$, and randomness $s$,*

$$M_{\mathsf{A}}\left(x, g(x,y;s)\right) = M_{\mathsf{B}}\left(y, g(x,y;s)\right)$$

*and*

$$\mathsf{SD}\left(f(x,y), M_{\mathsf{A}}\left(x, g(x,y)\right)\right) = 0.$$

*We say $f$ and $g$ are isomorphic (i.e., $f \simeq g$) if $f \leqslant g$ and $g \leqslant f$.*

Intuitively, $f$ and $g$ are isomorphic if securely computing $f$ can be realized by one ideal call to $g$ without any further communication and vise versa. As an example, the (deterministic) $\mathsf{XOR}$ functionality $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ is isomorphic to $\begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix}$.

Given two isomorphic functionalities $f$ and $g$, it is easy to see that there is a natural bijection between protocols in the $f$-hybrid model and $g$-hybrid model.

**Lemma 5.5.1.** *Let $f$ and $g$ be two functionalities such that $f \simeq g$. For every fair coin-tossing protocol $\pi$ in the $f$-hybrid model, there is a fair coin-tossing protocol $\pi'$ in the $g$-hybrid model such that*

- *$\pi$ and $\pi'$ have the same message complexity $r$ and expected output $X_0$.*
- *For every fail-stop attack strategy for $\pi$, there exists a fail-stop attack strategy for $\pi'$ such that the insecurities they cause are identical and vice versa.*

*Sketch.* Given any protocol $\pi$ in the $f$-hybrid model between $\mathsf{A}$ and $\mathsf{B}$, consider the protocol $\pi'$ in the $g$-hybrid model between $\mathsf{A}'$ and $\mathsf{B}'$. In $\pi'$, $\mathsf{A}'$ simply simulates $\mathsf{A}$ and does what $\mathsf{A}$ does. Except when the trusted party sends the output of $g$, $\mathsf{A}'$ uses the mapping $M_{\mathsf{A}}$ to recover the output of $f$ and feeds it to $\mathsf{A}$. $\mathsf{B}'$ behaves similarly. Easily, one can verify that these two protocols have the same message complexity and expected output. Additionally, for every fail-stop adversary $\mathsf{A}^*$ for $\pi$, there is a fail-stop adversary $(\mathsf{A}^*)'$ for $\pi'$ that simulates $\mathsf{A}^*$ in the same manner, which deviates the output of Bob by the same amount. □

We are now ready to state our next lemma.

**Lemma 5.5.2** (Maximally Renaming the Outputs of $f$)**.** *Let $f\colon \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$ be a (randomized) functionality that is not complete. There exists a functionality $f'\colon \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}'$ such that $f \simeq f'$ and $f'$ satisfies the following* strict cross product rule*. That is, for all $x_0, x_1 \in \mathcal{X}$, $y_0, y_1 \in \mathcal{Y}$, and $z' \in \mathcal{Z}'$, we have*

$$\Pr[f'(x_0,y_0) = z'] \cdot \Pr[f'(x_1,y_1) = z'] = \Pr[f'(x_1,y_0) = z'] \cdot \Pr[f'(x_0,y_1) = z'].$$

Following the example above, the $\mathsf{XOR}$ functionality $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ satisfies the cross product rule, i.e., $\mathsf{XOR}$ is not complete, but it does not satisfy the strict cross product rule since

$$\Pr[\mathsf{XOR}(0,0)=1]\cdot\Pr[\mathsf{XOR}(1,1)=1] \neq \Pr[\mathsf{XOR}(1,0)=1]\cdot\Pr[\mathsf{XOR}(0,1)=1].$$

On the other hand, functionality $\begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix}$ is isomorphic to $\mathsf{XOR}$ and does satisfy the strict cross product rule.

*Proof of Lemma 5.5.2.* We shall rename the output of $f$ as follows. For all $z \in \mathcal{Z}$, define

$$\mathcal{S}_z := \{(x,y)\colon x\in\mathcal{X},\ y\in\mathcal{Y},\ \Pr[f(x,y)=z]>0\}.$$

By the cross product rule, we know that there does not exist $x_0, x_1 \in \mathcal{X}$ and $y_0, y_1 \in \mathcal{Y}$ such that

$$(x_0,y_0),(x_0,y_1),(x_1,y_0)\in\mathcal{S}_z \quad \text{but} \quad (x_1,y_1)\notin\mathcal{S}_z.$$

Therefore, we can always partition $\mathcal{S}_z$ as a collection of combinatorical rectangles. That is, there exists subsets $\mathcal{X}_1,\ldots,\mathcal{X}_\ell \subseteq \mathcal{X}$ and $\mathcal{Y}_1,\ldots,\mathcal{Y}_\ell \subseteq \mathcal{Y}$ such that

$$\mathcal{S}_z = \bigcup_{i=1}^{\ell} \mathcal{X}_i \times \mathcal{Y}_i,$$

and

$$\forall 1 \leqslant i < j \leqslant \ell \quad \mathcal{X}_i \cap \mathcal{X}_j = \emptyset \quad \text{and} \quad \mathcal{Y}_i \cap \mathcal{Y}_j = \emptyset.$$

Now define randomized functionality $f'\colon \mathcal{X}\times\mathcal{Y}\to\mathcal{Z}'$ as follows. Given input $x$ and $y$ with randomness $s$, let $z = f(x,y;s)$. Let $i$ be the index such that $(x,y)\in\mathcal{X}_i\times\mathcal{Y}_i$. Define

$$f'(x,y;s) := z^{(i)}.$$

Here, $z^{(i)}$ is an (arbitrarily picked) distinct output.

It is trivial to verify that, given $f'(x,y)$, Alice and Bob can recover the same sample, which is identically distributed as $f(x,y)$. On the other hand, given private input $x$ (resp., $y$) and a sample of $f(x,y)$, Alice (resp., Bob) can recover a sample of $f'(x,y)$. Additionally,

they shall always recover the same sample, which is identically distributed as $f'(x, y)$. This proves that $f$ and $f'$ are isomorphic.

Next, we verify that $f'$ satisfies the strict cross product rule. Given any $x_0, x_1 \in \mathcal{X}$, $y_0, y_1 \in \mathcal{Y}$, and $z^{(i)} \in \mathcal{Z}'$, if either $x_0 \notin \mathcal{X}_i$ or $x_1 \notin \mathcal{X}_i$, it is trivially true. Similarly, if either $y_0 \notin \mathcal{Y}_i$ or $y_1 \notin \mathcal{Y}_i$, it is also trivial. Otherwise, when both $x_0, x_1 \in \mathcal{X}_i$ and $y_0, y_1 \in \mathcal{Y}_i$, strict cross product rule follows from cross product rule.

This completes the proof. □

By Lemma 5.5.1, the insecurity of a fair coin-tossing protocol in the $f$-hybrid model is identical to a fair coin-tossing protocol in the $f'$-hybrid model when $f \simeq f'$. Therefore, in the rest of this section, without loss of generality, we shall always assume $f$ is maximally renamed according to Lemma 5.5.2 such that it satisfies the strict cross product rule.

### 5.5.2 Notations and the Technical Theorem

Let $\pi$ be an $(X_0, r)$-fair coin-tossing protocol in the $f$-hybrid model. We shall use $R^{\mathsf{A}}$ and $R^{\mathsf{B}}$ to denote the private randomness of Alice and Bob. We use random variable $M_i$ to denote the $i^{th}$ message of the protocol, which could be either an Alice/Bob message or a trusted party message. Let $X_i$ be the expected output of the protocol conditioned on the first $i$ messages of the protocol. In particular, this definition is consistent with the definition of $X_0$.

For an arbitrary $i$, we consider both Alice aborts and Bob aborts the $i^{th}$ message. Suppose the $i^{th}$ message is Alice's message. Alice abort means that she aborts without sending this message to Bob. Conversely, Bob abort means he aborts in his next message immediately after receiving this message. On the other hand, if this is a trusted party message, then both a fail-stop Alice and a fail-stop Bob can abort this message. This prevents the other party from receiving the message. We refer to the defense output of Alice when Bob aborts the $i^{th}$ message as Alice's $i^{th}$ defense. Similarly, we define the $i^{th}$ defense of Bob. Let $D_i^{\mathsf{A}}$ (resp., $D_i^{\mathsf{B}}$) be the expectation of Alice's (resp., Bob's) $i^{th}$ defense conditioned on the first $i$ messages.

Now, we are ready to state our main theorem, which shows that the most devastating fail-stop attack is guaranteed to achieve a high score. In light of the remarks above, Theorem 5.5.3 directly implies Theorem 5.5.2.

**Theorem 5.5.3.** *For any $(X_0, r)$-fair coin-tossing protocol $\pi$ in the $f$-hybrid model, we have*

$$\mathsf{Opt}(\pi) \geqslant \Gamma_r \cdot X_0 \left(1 - X_0\right).$$

### 5.5.3 The Proof

In this section, we shall prove Theorem 5.5.3 by using mathematical induction on the message complexity $r$. Let us first state some useful lemmas.

Firstly, we note that in the $f$-hybrid model, where $f$ is a (randomized) functionality that satisfies the strict cross product rule, Alice view and Bob view are always independent conditioned on the partial transcript.

**Lemma 5.5.3** (Independence of Alice and Bob view)**.** *For any $i$ and partial transcript $m_{\leqslant i}$, conditioned on this partial transcript, the joint distribution of Alice and Bob private randomness is identical to the product of the marginal distribution. That is,*

$$\mathsf{SD}\left( \left(R^{\mathsf{A}}, R^{\mathsf{B}}\right) \middle| M_{\leqslant i} = m_{\leqslant i} \ , \ \left(R^{\mathsf{A}} \middle| M_{\leqslant i} = m_{\leqslant i}\right) \times \left(R^{\mathsf{B}} \middle| M_{\leqslant i} = m_{\leqslant i}\right) \right) = 0.$$

In particular, this lemma implies the following claim.

**Claim 5.5.1** (Global Invariant)**.** *Let $\pi$ be an arbitrary fair coin-tossing protocol in the $f$-hybrid model. Suppose there are $\ell$ possible first messages, namely, $m_1^{(1)}, m_1^{(2)}, \ldots, m_1^{(\ell)}$, each happens with probability $p^{(1)}, p^{(2)}, \ldots, p^{(\ell)}$. Suppose conditioned on the first message being $M_1 = m_1^{(i)}$, the expected defense of Alice and Bob are $d_1^{\mathsf{A},(i)}$ and $d_1^{\mathsf{B},(i)}$ respectively. Then we have*

$$\sum_{i=1}^{\ell} p^{(i)} \cdot d_1^{\mathsf{A},(i)} d_1^{\mathsf{B},(i)} = D_0^{\mathsf{A}} \cdot D_0^{\mathsf{B}}.$$

*Proof.* Consider the probability that both Alice's first defense and Bob's first defense are 1. On the one hand, since Alice view and Bob view are independent, this equals to the product of the probability that Alice's first defense is 1 and the probability that Bob's first defense is

1, i.e., $D_0^{\mathsf{A}} \cdot D_0^{\mathsf{B}}$. On the other hand, conditioned on the first message being $M_1 = m_1^{(i)}$, Alice view and Bob view are still independent. Hence, by the same reasoning, the probability that both Alice's first defense and Bob's first defense are 1 is $d_1^{\mathsf{A},(i)} d_1^{\mathsf{B},(i)}$. Therefore,

$$\sum_{i=1}^{\ell} p^{(i)} \cdot d_1^{\mathsf{A},(i)} d_1^{\mathsf{B},(i)} = D_0^{\mathsf{A}} \cdot D_0^{\mathsf{B}}. \qquad \square$$

**Base case:** $r = 1$**.**

We are now ready to prove Theorem 5.5.3. Let us start with the base case. In the base case, the protocol consists of only one message. Recall that the last message of the protocol is a boundary case of our score function. It might not be the case that both parties can attack this message. Hence, we prove it in different cases.

*Case 1: Alice message.* Suppose this message is an Alice message. In this case, we shall only consider the attack by Alice. By definition, with probability $X_0$, Alice will send a message, conditioned on which the output shall be 1. And with probability $1 - X_0$, Alice will send a message, conditioned on which the output shall be 0. On the other hand, the expectation of Bob's defense will remain the same as $D_0^{\mathsf{B}}$. Therefore, the maximum of the score shall be

$$X_0 \cdot \left|1 - D_0^{\mathsf{B}}\right| + (1 - X_0) \cdot \left|0 - D_0^{\mathsf{B}}\right|,$$

which is

$$\geqslant X_0 (1 - X_0).$$

In particular, this is

$$\geqslant \Gamma_1 \cdot X_0 (1 - X_0).$$

*Case 2: Bob message.* This case is entirely analogous to case 1.

*Case 3: Trusted party message.* In this case, we shall consider the effectiveness of the attacks by both parties. Suppose there are $\ell$ possible first message by the trusted party, namely, $m_1^{(1)}, m_1^{(2)}, \ldots, m_1^{(\ell)}$, each happens with probability $p^{(1)}, p^{(2)}, \ldots, p^{(\ell)}$. Conditioned on first message being $M_1 = m_1^{(i)}$, the output of the protocol is $x_1^{(i)}$. We must have $x_1^{(i)} \in \{0, 1\}$ since

the protocol has ended and parties shall agree on the output. Furthermore, let the expected defense of Alice and Bob be $d_1^{\mathsf{A},(i)}$ and $d_1^{\mathsf{B},(i)}$. Therefore, the maximum of the score will be

$$\sum_{i=1}^{\ell} p^{(i)} \cdot \left( \left| x_1^{(i)} - d_1^{\mathsf{A},(i)} \right| + \left| x_1^{(i)} - d_1^{\mathsf{B},(i)} \right| \right).$$

We have

$$\begin{aligned}
&\sum_{i=1}^{\ell} p^{(i)} \cdot \left( \left| x_1^{(i)} - d_1^{\mathsf{A},(i)} \right| + \left| x_1^{(i)} - d_1^{\mathsf{B},(i)} \right| \right) \\
\geqslant &\sum_{i=1}^{\ell} p^{(i)} \cdot \left( x_1^{(i)} \left(1 - x_1^{(i)}\right) + \left( x_1^{(i)} - d_1^{\mathsf{A},(i)} \right)^2 + \left( x_1^{(i)} - d_1^{\mathsf{B},(i)} \right)^2 \right) && \text{(Since } x_1^{(i)} \in \{0,1\}\text{)} \\
= &\sum_{i=1}^{\ell} p^{(i)} \cdot \left( x_1^{(i)} + \left( x_1^{(i)} - d_1^{\mathsf{A},(i)} - d_1^{\mathsf{B},(i)} \right)^2 - 2 d_1^{\mathsf{A},(i)} d_1^{\mathsf{B},(i)} \right) && \text{(Identity Transformation)} \\
\geqslant &X_0 + \left( X_0 - D^{\mathsf{A}} - D^{\mathsf{B}} \right)^2 - \sum_{i=1}^{\ell} p^{(i)} \cdot 2 d_1^{\mathsf{A},(i)} d_1^{\mathsf{B},(i)} \\
& && \text{(Jensen's inequality on convex function } F(x,y,z) := (x-y-z)^2\text{)} \\
= &X_0 + \left( X_0 - D^{\mathsf{A}} - D^{\mathsf{B}} \right)^2 - 2 D_0^{\mathsf{A}} \cdot D_0^{\mathsf{B}} && \text{(Claim 5.5.1)} \\
= &X_0 \left(1 - X_0\right) + \left( X_0 - D_0^{\mathsf{A}} \right)^2 + \left( X_0 - D_0^{\mathsf{B}} \right)^2 && \text{(Identity Transformation)} \\
\geqslant &X_0 \left(1 - X_0\right) \\
\geqslant &\Gamma_1 \cdot X_0 \left(1 - X_0\right)
\end{aligned}$$

This completes the proof of the base case.

**Inductive Step.** Suppose the statement is true for message complexity $r$. Let $\pi$ be an arbitrary protocol with message complexity $r+1$. Suppose there are $\ell$ possible first messages, namely, $m_1^{(1)}, m_1^{(2)}, \ldots, m_1^{(\ell)}$, each happens with probability $p^{(1)}, p^{(2)}, \ldots, p^{(\ell)}$. Conditioned on first message being $M_1 = m_1^{(i)}$, the output of the protocol is $x_1^{(i)}$ and the expected defense of Alice and Bob are $d_1^{\mathsf{A},(i)}$ and $d_1^{\mathsf{B},(i)}$ respectively. Note that conditioned on the first message being $M_1 = m_1^{(i)}$, the remaining protocol $\pi^{(i)}$ becomes a protocol with expected output $x_1^{(i)}$ and message complexity $r$. By our inductive hypothesis, we have

$$\mathsf{Opt}\left(\pi^{(i)}\right) \geqslant \Gamma_r \cdot x_1^{(i)} \left(1 - x_1^{(i)}\right).$$

On the other hand, we could also pick the first message $m_1^{(i)}$ as our stopping time, which yields a score of

$$\left|x_1^{(i)} - d_1^{\mathsf{A},(i)}\right| + \left|x_1^{(i)} - d_1^{\mathsf{B},(i)}\right|.$$

Therefore, the stopping time that witnesses the largest score yields (at least) a score of

$$\begin{aligned}
&\max\left(\Gamma_r \cdot x_1^{(i)}\left(1 - x_1^{(i)}\right)\ ,\ \left|x_1^{(i)} - d_1^{\mathsf{A},(i)}\right| + \left|x_1^{(i)} - d_1^{\mathsf{B},(i)}\right|\right) \\
\geqslant &\Gamma_{r+1} \cdot \left(x_1^{(i)}\left(1 - x_1^{(i)}\right) + \left(x_1^{(i)} - d_1^{\mathsf{A},(i)}\right)^2 + \left(x_1^{(i)} - d_1^{\mathsf{B},(i)}\right)^2\right) && \text{(Lemma 5.2.1)}
\end{aligned}$$

Therefore, $\mathsf{Opt}(\pi)$ is lower bounded by

$$\begin{aligned}
&\sum_{i=1}^{\ell} p^{(i)} \cdot \Gamma_{r+1} \cdot \left(x_1^{(i)}\left(1 - x_1^{(i)}\right) + \left(x_1^{(i)} - d_1^{\mathsf{A},(i)}\right)^2 + \left(x_1^{(i)} - d_1^{\mathsf{B},(i)}\right)^2\right) \\
=&\Gamma_{r+1} \cdot \sum_{i=1}^{\ell} p^{(i)} \cdot \left(x_1^{(i)} + \left(x_1^{(i)} - d_1^{\mathsf{A},(i)} - d_1^{\mathsf{B},(i)}\right)^2 - 2d_1^{\mathsf{A},(i)} d_1^{\mathsf{B},(i)}\right) && \text{(Identity Transformation)} \\
\geqslant &\Gamma_{r+1} \cdot \left(X_0 + \left(X_0 - D^{\mathsf{A}} - D^{\mathsf{B}}\right)^2 - \sum_{i=1}^{\ell} p^{(i)} \cdot 2d_1^{\mathsf{A},(i)} d_1^{\mathsf{B},(i)}\right) \\
&&& \text{(Jensen's inequality on convex function } F(x,y,z) := (x-y-z)^2\text{)} \\
=&\Gamma_{r+1} \cdot \left(X_0 + \left(X_0 - D^{\mathsf{A}} - D^{\mathsf{B}}\right)^2 - 2D_0^{\mathsf{A}} \cdot D_0^{\mathsf{B}}\right) && \text{(Claim 5.5.1)} \\
=&\Gamma_{r+1} \cdot \left(X_0\left(1 - X_0\right) + \left(X_0 - D_0^{\mathsf{A}}\right)^2 + \left(X_0 - D_0^{\mathsf{B}}\right)^2\right) && \text{(Identity Transformation)} \\
\geqslant &\Gamma_{r+1} \cdot X_0\left(1 - X_0\right)
\end{aligned}$$

This completes the proof of the inductive step.

## 5.6 Separation from Public-key encryption

In this section, we prove that public-key encryption used in a black-boxed manner shall not enable optimal fair coin-tossing. Our objective is to prove the existence of an oracle, with respect to which public-key encryption exists, but optimal fair coin-tossing does not.

### 5.6.1 Public-key Encrytion Oracles

Let $n$ be the security parameter. We follow the work of [MMP14] and define the following set of functions.

- $\mathsf{Gen}\colon \{0,1\}^n \to \{0,1\}^{3n}$. This function is a random injective function.
- $\mathsf{Enc}\colon \{0,1\}^{3n} \times \{0,1\}^n \to \{0,1\}^{3n}$. This function is uniformly randomly sampled among all functions that are injective with respect to the second input. That is, when the first input is fixed, this function is injective.
- $\mathsf{Dec}\colon \{0,1\}^n \times \{0,1\}^{3n} \to \{0,1\}^n \cup \{\perp\}$. This function is the uniquely determined by functions $\mathsf{Gen}$ and $\mathsf{Enc}$ as follows. $\mathsf{Dec}$ takes as inputs a $\mathsf{sk}sk \in \{0,1\}^n$ and a ciphertext $c \in \{0,1\}^{3n}$. If there exists a message $m \in \{0,1\}^n$ such that $\mathsf{Enc}(\mathsf{Gen}(sk), m) = c$, define $\mathsf{Dec}(sk, c) := m$. Otherwise, define $\mathsf{Dec}(sk, c) := \perp$. Note that such message $m$, if exists, must be unique, because $\mathsf{Enc}$ is injective with respect to the second input.
- $\mathsf{Test}_1\colon \{0,1\}^{3n} \to \{0,1\}$. This function is uniquely determined by function $\mathsf{Gen}$. It takes as an input a $\mathsf{pk}pk \in \{0,1\}^{3n}$. If there exists a $\mathsf{sk}sk \in \{0,1\}^n$ such that $\mathsf{Gen}(sk) = pk$, define $\mathsf{Test}_1(pk) := 1$. Otherwise, define $\mathsf{Test}_1(pk) := 0$.
- $\mathsf{Test}_2\colon \{0,1\}^{3n} \times \{0,1\}^{3n} \to \{0,1\}$. This function is uniquely determined by function $\mathsf{Enc}$. It takes as inputs a $\mathsf{pk}pk \in \{0,1\}^{3n}$ and a ciphertext $c \in \{0,1\}^{3n}$. If there exists a message $m$ such that $\mathsf{Enc}(pk, m) = c$, define $\mathsf{Test}_2(pk, c) := 1$. Otherwise, define $\mathsf{Test}_2(pk, c) := 0$.

We shall refer to this collection of oracles the PKE oracle. Trivially, the PKE oracle enables public-key encryption. We shall prove that it does not enable optimally-fair coin-tossing. We remark that it is necessary to include the test functions $\mathsf{Test}_1$ and $\mathsf{Test}_2$. Otherwise, it can be used to construct oblivious transfer protocols against semi-honest adversaries [GKM+00, LOZ12], which can be further used to construct optimally-fair coin-tossing protocols [MNS09].

### 5.6.2 Our Results

We shall prove the following theorem.

**Theorem 5.6.1** (Main theorem for PKE Oracle)**.** *There exists a universal polynomial $p(\cdot,\cdot,\cdot,\cdot)$ such that the following holds. Let $\pi$ be any fair coin-tossing protocol in the PKE oracle model, where Alice and Bob make at most $m$ queries. Let $X_0$ be the expected output, and $r$ be the message complexity of $\pi$. There exists an (information-theoretic) fail-stop attacker that deviates the expected output of the other party by (at least)*

$$\Omega\left(\frac{X_0\left(1-X_0\right)}{\sqrt{r}}\right).$$

*This attacker shall ask at most $p\left(n, m, r, \frac{1}{X_0(1-X_0)}\right)$ additional queries.*

It is instructive to understand why Theorem 5.5.2 does not imply Theorem 5.6.1. One may be tempted to model the public-key encryption primitive as an idealized secure function evaluation functionality to prove this implication. The idealized functionality for public-key encryption delivers sender's message to the receiver, while hiding it from the eavesdropper. So, the "idealized public-key encryption" functionality is a three-party functionality where the sender's input is delivered to the receiver; the eavesdropper has no input or output. This idealized effect is easily achieved given secure point-to-point communication channels, which we assume in our work. The non-triviality here is that our result is with respect to an *oracle* that implements the public-key encryption functionality. An oracle for public-key encryption is not necessarily used just for secure message passing.

**Remark 5.6.1.** *As usual in the literature [DLMM11, DMM14, MW20], we shall only consider* instant protocols*. That is, once a party aborts, the other party shall not make any additional queries to defend, but directly output her current defense coin. We refer the reader to [DLMM11] for justification and more details on this assumption.*

In fact, our proof technique is sufficient to prove the following stronger theorem.

**Theorem 5.6.2.** *There exists a universal polynomial $p(\cdot,\cdot,\cdot,\cdot)$ such that the following holds. Let $f$ be any (randomized) functionality that is not complete. Let $\pi$ be any fair coin-tossing protocol in the $f$-hybrid model where parties have access to the PKE oracle model. Assume Alice and Bob make at most $m$ queries. Let $X_0$ be the expected output, and $r$ be the message*

*complexity of $\pi$. There exists an (information-theoretic) fail-stop attacker that deviates the expected output of the other party by (at least)*

$$\Omega\left(\frac{X_0\left(1-X_0\right)}{\sqrt{r}}\right).$$

*This attacker shall ask at most $p\left(n, m, r, \frac{1}{X_0(1-X_0)}\right)$ additional queries.*

Our proof strategy is similar to that of [MMP14]. It consists of the following two steps.

1. Given a protocol in the PKE oracle model, we shall first convert it into a protocol where parties do not invoke the decryption queries. By Imported Theorem 5.6.1 proven in [MMP14], we can convert it in a way such that the insecurity of these two protocols in the presence of a semi-honest adversary is (almost) identical. In particular, this ensures that the insecurity of fair coin-tossing protocol in the presence of a fail-stop adversary is (almost) identical.

2. Next, we shall extend the results of [MW20], where they proved a fair coin-tossing protocol in the random oracle model is highly insecure, to the setting of PKE oracles without decryption oracle. Intuitively, The proof of [MW20] only relied on the fact that in the random oracle model, there exists a public algorithm [BM09] that asks polynomially many queries and decorrelate the private view of Alice and Bob. Mahmoody, Maji, and Prabhakaran [MMP14] proved that (summarized as Imported Theorem 5.6.2) the PKE oracles without the decryption oracle satisfies the similar property. Hence, the proof of [MW20] extends naturally to this setting.

Together, these two steps prove Theorem 5.6.1. The first step is summarized in Section 5.6.3. The second step is summarized in Section 5.6.4.

### 5.6.3 Reduction from PKE Oracle to Image Testable Random Oracle

A (keyed version of) *image-testable random oracles* is a collection of pairs of oracles $(R^{\mathsf{key}}, T^{\mathsf{key}})$ parameterized by a key, such that for every key, the following holds.

- $R^{\mathsf{key}}\colon \{0,1\}^n \to \{0,1\}^{3n}$ is a randomly sampled injective function.

- $T^{\mathsf{key}}\colon \{0,1\}^{3n} \to \{0,1\}$ is uniquely determined by function $R^{\mathsf{key}}$ as follows. Define $T^{\mathsf{key}}(\beta) := 1$ if there exists an $\alpha \in \{0,1\}^n$ such that $R^{\mathsf{key}}(\alpha) = \beta$. Otherwise, define $T^{\mathsf{key}}(\beta) = 0$.

Observe that the PKE oracle without the decryption oracle $\mathsf{Dec}$ is exactly a (keyed version of) image-testable random oracles with the keys drawn from $\{\perp\} \cup \{0,1\}^{3n}$. If the key is $\perp$, it refers to the pair of oracles $(\mathsf{Gen}, \mathsf{Test}_1)$. If the $\mathsf{key} \in \{0,1\}^{3n}$, it refers to the pair of oracles $(\mathsf{Enc}(\mathsf{key},\cdot), \mathsf{Test}_2(\mathsf{key},\cdot))$. We shall refer to the PKE oracle without the decryption oracle $\mathsf{Dec}$ as ITRO.

We shall use the following imported theorem, which is implicitly proven in [MMP14].

**Imported Theorem 5.6.1** ([MMP14])**.** *There exists a universal polynomial $p(\cdot,\cdot)$ such that the following holds. Let $\pi$ be a fair coin-tossing protocol in the PKE oracle model. Let $X_0$ and $r$ be the expected output and message complexity. Suppose Alice and Bob ask (at most) $m$ queries. For any $\varepsilon > 0$, there exists a fair coin-tossing protocol $\pi'$ in the ITRO model such that the following holds.*

- *Let $X_0'$ and $r'$ be the expected output and message complexity of $\pi'$. Then, $r' = r$ and $|X_0' - X_0| < \varepsilon$.*
- *Parties asks at most $p(m, 1/\varepsilon)$ queries in protocol $\pi'$.*
- *For any semi-honest adversary $\mathcal{A}'$ for protocol $\pi'$, there exists a semi-honest adversary $\mathcal{A}$ for protocol $\pi$, such that the view of $\mathcal{A}$ is $\varepsilon$-close to the view of $\mathcal{A}'$. And vice versa. In particular, this implies that if $\pi'$ is $\alpha$-insecure. $\pi$ is (at least) $(\alpha - \varepsilon)$-insecure.*

The intuition behind this theorem is the following. To avoid the uses of decryption oracle, parties are going to help each other decrypt. In more detail, suppose Alice generates a ciphertext using Bob's public key. Whenever the probability that Bob invokes the decryption oracle on this ciphertext is non-negligibly high, Alice will directly reveal the message to Bob. Hence, Bob does not need to use the decryption oracle. This shall not harm the security as a semi-honest Bob can recover the message by asking polynomially many additional queries. We refer the readers to [MMP14] for more details.

Looking forward, we shall prove that any fair coin-tossing protocol in the ITRO model is $\Omega\left(\frac{X_0'(1-X_0')}{\sqrt{r}}\right)$-insecure. By setting $\varepsilon$ to be $1/\mathsf{poly}$ for some sufficiently large polynomial, we shall guarantee that

$$\varepsilon = \mathrm{o}\left(\frac{X_0\left(1-X_0\right)}{\sqrt{r}}\right).$$

This guarantees that the insecurity of the protocol in the PKE oracle model is (qualitatively) identical to the insecure of the protocol in the ITRO model.

### 5.6.4 Extending the proof of [MW20] to Image Testable Random Oracle

We first recall the following theorem from [MMP14].

**Imported Theorem 5.6.2** (Common Information Learner [MMP14])**.** *There exists a universal polynomial $p(\cdot,\cdot)$ such that the following holds. Let $\pi$ be any two-party protocol in the ITRO model, in which both parties make at most $m$ queries. For all threshold $\varepsilon \in (0,1)$, there exists a public algorithm, called the* common information learner*, who has access to the transcript between Alice and Bob. After receiving each message, the common information learner performs a sequence of queries and obtain its corresponding answers from the ITRO. Let $M_i$ denote the $i^{th}$ message of the protocol. Let $H_i$ denote the sequence of query-answer pairs asked by the common information learner after receiving the message $M_i$. Let $T_i$ be the union of the $i^{th}$ message $M_i$ and the $i^{th}$ common information learner message $H_i$. Let $V_i^{\mathsf{A}}$ (resp., $V_i^{\mathsf{B}}$) denote Alice's (resp., Bob's) private view immediately after message $T_i$, which includes her private randomness, private queries, and the public partial transcript. , The common information learner guarantees that the following conditions are simultaneously satisfied.*

- ***Cross-product Property.*** *Fix any round $i$,*

$$\mathop{\mathrm{E}}_{t_{\leqslant i}\leftarrow T_{\leqslant i}}\left[\mathsf{SD}\left(\left(V_i^{\mathsf{A}},V_i^{\mathsf{B}}\middle|T_{\leqslant i}=t_{\leqslant i}\right),\left(V_i^{\mathsf{A}}\middle|T_{\leqslant i}=t_{\leqslant i}\right)\times\left(V_i^{\mathsf{B}}\middle|T_{\leqslant i}=t_{\leqslant i}\right)\right)\right]\leqslant\varepsilon.$$

  *Intuitively, it states that* on average*, the statistical distance between (1) the joint distribution of Alice's and Bob's private view, and (2) the product of the marginal distributions of Alice's private views and Bob's private views is small.*

- ***Efficient Property.*** *The expected number of queries asked by the common information learner is bounded by $p(m, 1/\varepsilon)$.*

This theorem, combined with proof of [MW20] gives the following theorem.

**Theorem 5.6.3.** *There exists a universal polynomial $p(\cdot,\cdot,\cdot,\cdot)$ such that the following holds. Let $\pi$ be a protocol in the ITRO model, where Alice and Bob make at most $m$ queries. Let $X_0$ and $r$ be the expected output and message complexity. Then, there exists an (information-theoretic) fail-stop adversary that deviates the expected output of the other party by*

$$\Omega\left(\frac{X_0\left(1-X_0\right)}{\sqrt{r}}\right).$$

*This attacker asks at most $p\left(n, m, r, \frac{1}{X_0(1-X_0)}\right)$ additional queries.*

Below, we briefly discuss why Imported Theorem 5.6.2 is sufficient to prove this theorem. The full proof is analogous to [MW20] and the proof of the results in the $f$-hybrid model. Hence we omit it here.

On a high level, the proof goes as follows. We prove Theorem 5.6.3 by induction. Conditioned on the first message, the remaining protocol becomes an $(r-1)$-message protocol, and one can apply the inductive hypothesis. For every possible first message $i$, we consider whether to abort immediately or defer the attack to the remaining sub-protocol. By invoking Lemma 5.2.1, we obtain a potential function, which characterizes the insecurity of the protocol with first message being $i$. This potential function will be of the form

$$\Phi(x_i, a_i, b_i) = x_i(1-x_i) + (x_i - a_i)^2 + (x_i - b_i)^2,$$

where $x_i$, $a_i$, and $b_i$ stands for the expected output, expected Alice defense, and expected Bob defense, respectively. To complete the proof, [MW20] showed that it suffices to prove the following Jensen's inequality.

$$\mathop{\mathrm{E}}_i\left[\Phi(x_i, a_i, b_i)\right] \geqslant \Phi\left(\mathop{\mathrm{E}}_i\left[x_i\right], \mathop{\mathrm{E}}_i\left[a_i\right], \mathop{\mathrm{E}}_i\left[b_i\right]\right).$$

To prove this, one can rewrite $\Phi(x, a, b)$ as

$$\Phi(x, a, b) = x + (x - a - b)^2 - 2ab.$$

We note that $x$ and $(x - a - b)^2$ are convex functions, and hence Jensen's inequality holds. As for the term $ab$, we shall have

$$\mathop{\mathrm{E}}_{i}[a_i b_i] \approx \mathop{\mathrm{E}}_{i}[a_i] \cdot \mathop{\mathrm{E}}_{i}[b_i]$$

as long as, conditioned on every possible first message $i$, Alice's private view is (almost) independent to Bob's private view. This is exactly what Imported Theorem 5.6.2 guarantees except for a small error depending on $\varepsilon$, which we shall set to be sufficiently small. Therefore, the proof shall follow.

# 6. OPTIMALLY-SECURE COIN-TOSSING AGAINST A BYZANTINE ADVERSARY

In a seminal work, Ben-Or and Linial [BL85, BL89] introduced the full information model to study collective coin-tossing protocols. One relies on collective coin-tossing protocols to upgrade local private randomness of each of the $n$ processors into shared randomness that all processors agree. In this model, all the processors have unbounded computational power and communicate with each other over one broadcast channel. This model for the design and analysis of coin-tossing protocols turns out to be highly influential with close connections with diverse topics in mathematics and computer science, for example, extremal graph theory [Kru63, Kat68, Har66], extracting randomness from imperfect sources [SV84, CGH$^+$85, Vaz85, Fri92], cryptography [CI93, DLMM11, DMM14, HOZ16, KMM19, KMW20, MW20], game theory [BI64, Col71], circuit representation [Win71, OS08, OS11], distributed protocols [Asp97, Asp98, BJB98], and poisoning and evasion attacks on learning algorithms [DMM18, MDM19, MM19, EMM20].

A *bias-$X$ $n$-processor coin-tossing protocol* is an interactive protocol where every complete transcript is publicly associated with output 0 or 1, and the expected output for an honest execution of the protocol is $X \in [0,1]$. Given a bias-$X$ $n$-processor coin-tossing protocol $\pi$ and model for adversarial corruption and attack, let $\varepsilon^+(\pi) \in [0,1]$ represent the maximum increase in the expected output that an adversarial strategy can cause. Similarly, let $\varepsilon^-(\pi) \in [0,1]$ represent the maximum decrease in the expected output caused by an adversarial strategy. One defines the insecurity of a protocol $\pi$ as $\varepsilon(\pi) := \max\{\varepsilon^+(\pi), \varepsilon^-(\pi)\}$. For a fixed $X \in [0,1]$, the *optimal bias-$X$ $n$-processor protocol* minimizes $\varepsilon(\pi)$ among all bias-$X$ $n$-processor coin-tossing protocols.

For practical applications, given the tolerance for insecurity, one needs precise guarantees on the insecurity of coin-tossing protocols to estimate the necessary number of processors to keep the insecurity acceptably low. If the insecurity estimates for the potential coin-tossing protocols involve large latent constants or poly-logarithmic factors, then such a decision needs to be overly pessimistic in calculating the necessary number of processors. Consequently, it is essential to characterize coin-tossing protocols that are optimal or within a small constant

factor of the optimal protocol for every pair $(n, X)$. We emphasize that this outrightly rules out asymptotic bounds involving $n$. This work contributes to this endeavor.

We study $n$-processor coin-tossing protocols where every processor broadcasts a message exactly once (i.e., *single-turn*), and there are $n$ *rounds*, i.e., every round a unique processor broadcasts her message. The distribution of the messages sent by processors prescribed to speak in one round may depend on the messages sent in the previous rounds. For example, in one-round protocols, the distribution over the message space of the coin-tossing protocol is a product space. On the other hand, in single-turn $n$-round protocols, only one processor speaks in a round, and her message distribution possibly depends on all previously broadcast messages. Furthermore, which processor speaks in which round may depend on the messages sent in the previous rounds. We consider *adaptive Byzantine* adversaries who can corrupt $k = 1$ processor, i.e., based on the evolution of the protocol, our adversary can corrupt one processor and fix her message arbitrarily. As is standard in cryptography, our adversary is always *rushing*, i.e., it can arbitrarily schedule all those processors who are supposed to speak in a round.

Variants this model have been studied, and we highlight, in the sequel, some of the most prominent works and their technical highlights.

**Lichtenstein, Linial, and Saks [LLS89].** Lichtenstein et al. [LLS89] consider the restriction where the $i$-th processor broadcasts an independent and uniformly random bit $x_i$, where $1 \leqslant i \leqslant n$, and the adversary can corrupt up to $k$ processors, where $1 \leqslant k \leqslant n$. The coin-tossing protocol is a function $f\colon \{0,1\}^n \to \{0,1\}$. In this case, the underlying message space is $\{0,1\}^n$, which is a product space involving a small-size alphabet, and the probability distribution induced by the transcript is the uniform distribution over the message space. Note that, for $n$-processor coin-tossing protocols, the bias of such a protocol can only be an integral multiple of $2^{-n}$. Therefore, this is a discrete optimization problem.

Given $n$, $k$, and $X$, they begin with the objective of minimizing *only* the quantity $\varepsilon^+(\pi)$ over bias-$X$ $n$-processor coin-tossing protocols $\pi$. Their recursive characterization of the protocol that minimizes $\varepsilon^+$, *incidentally*, turns out to be identical to the optimal solution for the vertex isoperimetric inequality over the Boolean hypercube [Kru63, Kat68, Har66].

Therefore, a threshold protocol[1] $\pi$ is the optimal protocol and minimizes $\varepsilon^+$. The complementary protocol, which swaps the outputs 0 and 1 of $\pi$, is also a threshold protocol and, consequently, minimizes $\varepsilon^-$. So, threshold protocols simultaneously minimize $\varepsilon^+$ and $\varepsilon^-$ and achieve optimal security.

**Significantly altering the output distribution.** For symmetric functions (i.e., permuting the inputs of the function $f$ does not change its output), Goldwasser, Kalai, and Park [GKP15] prove that $k = \mathcal{O}\left(\sqrt{n} \cdot \mathsf{polylog}\left(n\right)\right)$ corruptions suffice to completely fix the output of any coin-tossing protocol even if the protocol relies on *arbitrary-length messages*. After that, Kalai, Komargodski, and Raz [KKR18] remove the restriction of symmetric functions. Recently, in independent work, Haitner and Karidi-Heller [HK20] extend this result to *multi-turn* coin-tossing protocols. These papers use global analysis techniques for martingales that are inherently non-constructive; consequently, they prove the optimality of threshold protocols up to $\mathcal{O}\left(\mathsf{polylog}\left(n\right)\right)$ factors when the adversary corrupts at most $k = \mathcal{O}\left(\sqrt{n} \cdot \mathsf{polylog}\left(n\right)\right)$ processors.

**Challenge for arbitrary-length messages.** Our objective is to provide tight insecurity estimates for the optimal coin-tossing protocols that use *arbitrary-length messages*. Let us understand why the technical approach of [LLS89] fails; and an entirely new approach is needed. In the full information model, without the loss of generality, one can assume that all interactive protocols are stateless, and processors use a fresh block of private randomness to generate the next message at any point during the evolution of the coin-tossing protocol [Jer85, JVV86, BGP00]. Furthermore, the security of the internal state of processors is not a concern, so, without loss of generality, every processor broadcasts its appropriate block of randomness whenever it speaks.[2] A Byzantine adversary can corrupt a processor and arbitrarily set its randomness. So, for an appropriately large alphabet $\Sigma$, which depends on the randomness complexity of generating each message, our message space is $\Sigma^n$, a product space involving a large alphabet set. Over such product spaces, the isolated objective

[1]↑More generally, protocols that output 1 for all strings smaller in the simplicial order than a threshold string are the optimal protocols.

[2]↑Let $\pi$ be the original coin-tossing protocol. In the compiled $\pi'$, suppose parties reveal the block of randomness that they use to prepare their next-message in the protocol $\pi'$. The new protocol $\pi'$, first, emulates the next-message function of $\pi$ to generate the entire transcript, and, then, uses $\pi$ to determine the output.

of minimizing $\varepsilon^+$ does not entail the simultaneous minimization $\varepsilon^-$. Given any $n \in \mathbb{N}$ and $X \in [0,1]$, there exist protocols with $(\varepsilon^+, \varepsilon^-) = (\frac{1-X}{n}, X)$ and $(\varepsilon^+, \varepsilon^-) = (1-X, \frac{X}{n})$, when the adversary can corrupt $k = 1$ processor (refer to Figure 6.1 and Figure 6.2 for the protocols). More generally, for product spaces over large alphabets, one does not expect such a vertex isoperimetric inequality [FHH+19, Har99].

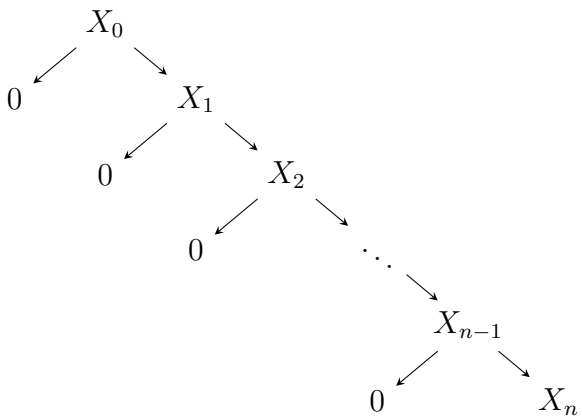



**Figure 6.1.** An example $n$-processor coin-tossing protocol that is easy to deviate toward 0, but hard to deviate toward 1. In this protocol, $X_k = X_0 + k \cdot \frac{1-X_0}{n}$. Adversary can corrupt the first processor and achieve $\varepsilon^+ = X_1 - X_0 = \frac{1-X_0}{n}$ by setting its message to be 1 or achieve $\varepsilon^- = X_0 - 0 = X_0$ by setting the its message to be 0.

Finally, global analysis techniques of [GKP15, KKR18, HK20] analyze the case of a large number of corruptions $k$. The optimally secure protocol for $k = 1$ is not apriori related to the optimal protocols robust to a large number of corruptions. Furthermore, the inductive proof technique of Aspnes [Asp97, Asp98] is agnostic of the expected output of the coin-tossing protocol. Consequently, reconstructing the optimal protocol from the lower-bound on insecurity is not apparent.

We follow the geometric technique of Khorasgani, Maji, and Mukherjee [KMM19], which is inherently constructive, to obtain tight estimates of the optimally secure protocols.

### Connection to Isoperimetric Inequalities

The connection to isoperimetric inequalities [Kru63, Kat68, Har66, Har99] (via the expansion of fixed density subset of product spaces) establishes the relevance to topics in

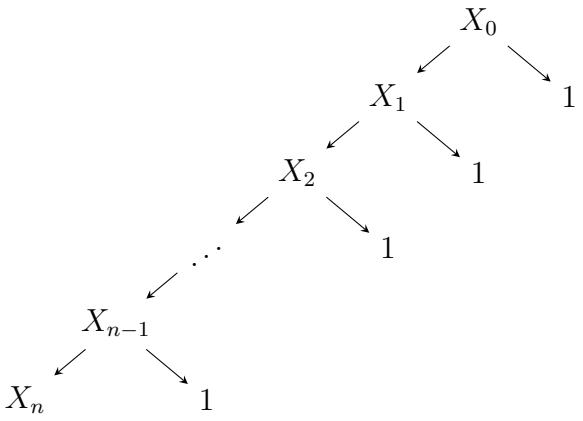



**Figure 6.2.** An example $n$-processor coin-tossing protocol that is easy to deviate toward 1, but hard to deviate toward 0. In this protocol, $X_k = k \cdot \frac{X_0}{n}$. Adversary can corrupt the first processor and achieve $\varepsilon^+ = 1 - X_0$ by setting the its message to be 1 or achieve $\varepsilon^- = X_1 - X_0 = \frac{X_0}{n}$ by setting its message to be 0.

theoretical computer science like expander graphs, complexity theory, and error-correcting codes.

**Encoding Security of Coin-tossing Protocols.** Every coin-tossing protocol is equivalent to a unique subset $S$ of an $n$-dimension product space $\Sigma^n$, where the size of the alphabet set $\sigma := |\Sigma|$ depends on the randomness complexity of the coin-tossing protocol. The elements of this product space represent the complete transcript of the coin-tossing protocol. The $i$-th coordinate of an element corresponds to the message sent by processor $i$, and the subset $S$ contains all elements of the product space on which the coin-tossing protocol outputs 1. One considers the uniform distribution over $\Sigma^n$ to sample the elements. This subsection considers a *stronger* Byzantine adversary who can edit one processor's message after seeing the message of all processors.

The discussion in this subsection extends to arbitrary corruption threshold $k$. However, for the simplicity of the presentation, we consider the specific case of $k = 1$. Let $\partial S_k^+$ be the set of elements in $\overline{S}$ (the complement of $S$) that are at a Hamming distance $k = 1$ from the set $S$. Consequently, the strong Byzantine adversary can change an element from the set $\partial S_k^+ \subseteq \overline{S}$ into some element of $S$ by editing (at most) $k$ coordinates. Note that if the

stronger Byzantine adversary can see *all* the messages and then performs the edits, then it can increase the expected output by exactly $\varepsilon^+ = \left|\partial S_k^+\right| / \sigma^n$.

Analogously, one defines the set $\partial S_k^- \subseteq S$ that contains all elements at a Hamming distance $k = 1$ from the set $\overline{S}$. So, a stronger Byzantine adversary can reduce the expected output by $\varepsilon^- = \left|\partial S_k^-\right| / \sigma^n$.

**Extremal Graph Theory Perspective.** The (width-$k$) *vertex perimeter* of the set $S$, represented by $\partial_{N,k} S$, is the set of all elements in $\overline{S}$ that are at a Hamming distance of at most $k$ from some element in $S$. Observe that the perimeter $\partial_{V,k} S$ is identical to the set $\partial S_k^+$. Similarly, the vertex perimeter of the set $\overline{S}$ (which is $\partial_{V,k}\overline{S}$) is identical to the set $\partial S_k^-$.

The objective of extremal graph theory is to characterize the optimal set $S$ of a density-$X$ that minimizes its vertex perimeter. This optimal set $S$, in turn, characterizes the bias-$X$ coin-tossing protocol with the minimum $\varepsilon^+$. In Figure 6.1 and Figure 6.2, we saw that minimizing $\varepsilon^+$ does not automatically entail the simultaneous minimization of $\varepsilon^-$ for general $\Sigma$.[3] In fact, that example highlighted that the protocol minimizing $\varepsilon^+$ resulted in a protocol where the stronger Byzantine adversary can force the outcome 0 with certainty. Therefore, there is a disconnect between the cryptographic objective of simultaneously minimizing $\varepsilon = \max\{\varepsilon^+, \varepsilon^-\}$ with the standard objective in extremal graph theory for large alphabet set $\Sigma$.

**Cryptography-inspired Extremal Graph Theory.** Instead of minimizing the vertex perimeter of a density-$S$ set $S$, one should consider the alternative objective of minimizing the *symmetric perimeter* of $S$ defined under various norms.

$$\partial^{\mathsf{sym}}_{V,k,\ell}(S) := \left(\left|\partial_{V,k} S\right|^\ell + \left|\partial_{V,k}\overline{S}\right|^\ell\right)^{1/\ell}.$$

The $\ell = \infty$ case corresponds to our cryptographic objective; however, this norm is difficult to analyze. Consequently, we study the norm $\ell = 1$ as a proxy, which is a 2-approximation of the norm $\ell = \infty$. Our results provide evidence that such symmetric perimeters may be more well-behaved in general.

Recall that, in our setting, the element in $\Sigma^n$ is exposed one coordinate at a time and our Byzantine adversaries cannot go back to edit previously exposed coordinates. So, our

[3]↑For $\Sigma = \{0, 1\}$, this entailment holds; otherwise, it is not known to hold in general.

Byzantine adversaries have lesser power than the stronger Byzantine adversaries considered in this section. Consequently, the *minimum achievable insecurity* for bias-$X$ $n$-processor coin-tossing protocols in our setting *lower-bounds* the proxy norm above. For instance, when $\ell = 1$, our results imply that the density of the symmetric perimeter is $1/\sqrt{n}$ for any dense set $S$, irrespective of the size of the alphabet set.

*Remark.* We identify a density-$X$ set with its corresponding bias-$X$ coin-tossing protocol. Using the independent bounded differences inequality for the Hamming distance function (using Azuma's inequality [Azu67]) on a constant-density subset $S$ implies that $k = \mathcal{O}(\sqrt{n})$ edits suffice to achieve any constant $\varepsilon^+$ and $\varepsilon^-$, for any $\sigma$.[4] However, for small $k$ (for example, $k = 1$), obtaining meaningful guarantees on both $\varepsilon^+$ and $\varepsilon^-$ is not possible for large $\sigma$. On the other hand, interestingly, we shall show that $\max\{\varepsilon^+, \varepsilon^-\} \geqslant 1/\sqrt{n}$ for any $\sigma$. This result lends support to the hypothesis that the symmetric perimeter is more well-behaved.

### 6.1 Our Contributions

Any $n$-processor coin-tossing protocol $\pi$ is equivalent to a depth-$n$ tree, where each node $v$ corresponds to a partial transcript. For every leaf of this tree, one associates the output of the coin-tossing protocol $\in \{0, 1\}$. For a partial transcript $v$, the *color of* $v$, represented by $x_v$, represents the expected output of the coin-tossing protocol conditioned on the partial transcript being $v$. For example, the leaves have color $\in \{0, 1\}$, and the color of the root of a bias-$X$ coin-tossing protocol is $X$. The probability $p_v$ represents the probability that the partial transcript $v$ is generated during the protocol evolution of $\pi$.

A Byzantine adversary, in this interpretation of a coin-tossing protocol, that corrupts at most $k = 1$ processor is equivalent to a prefix-free set of edges. That is, for any two edges $(u, v)$ and $(u', v')$ such that $u$ is the parent of $v$ and $u'$ is the parent of $v'$, the root to leaf path through $u$ does not pass through $u'$. Any such collection of edges corresponds to a unique Byzantine adversarial strategy. For example, if an edge $(u, v)$ lies in this set and $u$ is the parent of $v$, then this edges indicates that the Byzantine adversary decides to interfere when the protocol generates the partial transcript $u$, and this adversary sends the next message

[4]↑Even computationally efficient attacks are known to achieve this bound [MM19, EMM20].

that generates the partial transcript $v$. Note that the partial transcript $u$ uniquely identifies the processor that the adversary needs to corrupt.

Let $\tau$ be one such attack strategy. Suppose $\tau$ is a collection of $\ell$ edges, namely, $\{(u_i, v_i)\}_{i=1}^{\ell}$. Assume $u_i$ is the parent of $v_i$, for $i = 1, \ldots, \ell$. Then, we define the score of the attack strategy $\tau$ on protocol $\pi$ as

$$\mathsf{Score}\,(\pi, \tau) \;:=\; \sum_{i=1}^{\ell} p_{u_i} \cdot |x_{u_i} - x_{v_i}|\,.$$

The term $\mathsf{Score}\,(\pi, \tau)$ represents the vulnerability of protocol $\pi$ under attack strategy $\tau$. Furthermore, we define

$$\mathsf{Score}\,(\pi) \;:=\; \sup_{\tau}\; \mathsf{Score}\,(\pi, \tau)\,.$$

Intuitively, $\mathsf{Score}\,(\pi)$ represents the insecurity of the protocol under the most devastating attack, a.k.a., our potential function.

We emphasize that our score is not identical to the deviation in output distribution that a Byzantine adversary causes. It is a 2-approximation of that quantity. Define the insecurity as the maximum change that a Byzantine adversary can cause to the output distribution. Then, it is evident that the insecurity of $\pi$ is at least $\mathsf{Score}(\pi)/2$.

For an arbitrary $n \in \mathbb{N}^*$ and $t \in \{0, 1, \ldots, n+1\}$, let $\pi^{n,t}$ denote the $n$-processor $t$-threshold *threshold protocol*. In this threshold protocol, every processor broadcasts an independent and uniformly random bit. The output of this threshold protocol is 1 if and only if the total number of ones in the complete transcript is $\geqslant t$. An $n$-processor $t$-threshold protocol has color $2^{-n} \cdot \left(\sum_{i=t}^{n} \binom{n}{i}\right)$.

We prove the following theorem about the threshold protocol.

**Theorem 6.1.1.** *For any bias $X$ $n$-processor protocol $\pi$, where $X = 2^{-n} \cdot \left(\sum_{i=t}^{n} \binom{n}{i}\right)$, where $0 \leqslant t \leqslant n+1$, then*

$$\mathsf{Score}(\pi^{n,t}) \leqslant \mathsf{Score}(\pi).$$

That is, the threshold protocol is the protocol that minimizes the score. Equivalently, the insecurity of the threshold protocol is a 2-approximation of the optimal insecurity in our corruption model (refer to Corollary 6.5.2).

Furthermore, we also prove the following result. Suppose $X$ is not a root-color that admits a threshold protocol, and $X_0$ is inverse-polynomially far from both 0 and 1. Suppose $X$ is intermediate to the bias of the threshold protocols $\pi^{n-1,t}$ and $\pi^{n-1,t-1}$. Let $\pi$ be a protocol where the first processor decides to run the threshold protocol $\pi^{n-1,t}$ or $\pi^{n-1,t-1}$ with suitable probability so that the resulting protocol is a bias-$X$ protocol. Then, the insecurity of this protocol $\pi$ is a 4-approximation of the protocols with minimum insecurity against Byzantine adversaries (refer to Corollary 6.5.3).

## 6.2 Technical Overview

The techniques closest to our approach are those introduced by Aspnes [Asp97, Asp98] and Khorasgani et al. [KMM19, KMW20, MW20].

Aspnes' technique [Asp97, Asp98] tracks the locus of all possible $(\varepsilon^+, \varepsilon^-)$ corresponding to any $n$-processor $k$-corruption threshold protocol. However, the information regarding the root-color is lost and, consequently, the technique does not yield the optimal protocol construction. Next, one lower-bounds this space using easy-to-interpret (hyperbolic) curves and obtains bounds on the insecurity of any $n$-processor protocol with $k$ corruption threshold (against adversaries who erase the messages of processors).

The technique of Khorsgani et al. [KMM19, KMW20, MW20] use a potential function as a proxy to study the actual problem at hand. They maintain the locus of all $n$-processor bias-$X$ protocols that minimize the potential function. Next, they inductively build the next curve of $(n+1)$-processors bias-$X$ protocols that minimize the potential function. Their approach outrightly yields optimal constructions that minimize the potential function, and easily handle the case of processors sending *arbitrary-length* messages.

**High-level summary of our approach.** We use the potential function as introduced in Section 6.1, which is a 2-approximation of the optimal insecurity against Byzantine adversaries, for any $n$-processor bias-$X$ protocol. Let $C_n(X)$ represent the minimum realizable potential for bias-$X$ $n$-processor coin-tossing protocols.

Next, we prove that if an $n$-processor threshold protocol has potential $\delta$ and bias-$X$, then the point $(\delta, X)$ lies on the optimal curve $C_n(X)$. Therefore, the potential of these threshold protocols are 2-approximation of the optimal bias-$X$ protocol against Byzantine adversaries.

After that, inductively, we prove that the linear interpolation of the set of points $(\delta, X)$ realized by $n$-processor threshold protocols with potential $\delta$ and root-color $X$, where $0 \leqslant t \leqslant n+1$, is a lower-bound to the actual curve $C_n(X)$. Finally, we argue that a linear interpolation of appropriate threshold functions yields a protocol with potential that is 4-approximation of the optimal protocol against Byzantine adversaries.

**The curves and the inductive transformation.** Consider the case of $n = 1$ and arbitrary bias-$X$. If $X = 0$ or $X = 1$, then we have $C_1(X) = 0$. If $X \in (0, 1/2]$, then we include that edge that sets the output to 1. This observation creates a potential of $C_1(X) = 1 - X$. Similarly, we have $C_1(X) = X$, for all $X \in [1/2, 1)$. Our characterization of the curve $C_1(X)$ is complete (refer to Figure 6.4).

Next, consider the case of $n = 2$ and bias-$X$. This case is sufficient to understand how to inductively build the locus of the curve $C_{n+1}(X)$ inductive from $C_n(X)$. Consider any arbitrary 2-processor bias-$X$ coin-tossing protocol. Suppose the first processor sends message $1, 2, \ldots, \ell$. Let $x_i$, for $1 \leqslant i \leqslant \ell$, be the expected output conditioned on the first message being $i$. At the root of this protocol, we have two options. Corrupt processor one and send the message that achieves the highest potential. Or, defer the intervention to a later point in time.

Corrupting the root of this protocol causes the potential to become

$$\max_{i=1}^{\ell} |X - x_i| .$$

Deferring the intervention to a later point in time results in the potential becoming at least

$$\sum_{i=1}^{\ell} p_i \cdot C_1(x_i),$$

where $p_i$ is the probability that processor 1 outputs $i$. The actual potential of $\pi$ is the maximum of these two quantities. Our objective is to characterize the choice of $x_1, \ldots, x_\ell$ such that the potential is minimized (refer to Figure 6.3).

## 6.3 Preliminaries

We use $\mathbb{N}^*$ for the set of positive integers. For any two curves $C_1, C_2$ defined on $[0,1]$, we write $C_1 \preceq C_2$ ($C_1$ is below $C_2$) to denote that $C_1(x) \leqslant C_2(x)$ for each $x \in [0,1]$. A curve $C$ defined on $[0,1]$, is called concave if for all $0 \leqslant x < y \leqslant 1$, and any $\alpha \in [0,1]$, we have $C(\alpha x + (1-\alpha)y) \geqslant \alpha C(x) + (1-\alpha)C(y)$. Statistical distance between two distributions $A$ and $B$ defined over discrete sample space $\Omega$ is defined as $\mathsf{SD}(A,B) := \frac{1}{2}\sum_{x\in\Omega} |A(x) - B(x)|$. A function $f\colon \mathbb{N} \to \mathbb{R}$ is called negligible if for any polynomial $p(n)$, $f(n) = \mathrm{o}(1/p(n))$.

### 6.3.1 Coin-tossing Protocols

In this chapter, we consider coin-tossing protocols among $n$ processors in the *full information* model. That is, all processors communicate through one single broadcast channel. In particular, we consider an $n$-round protocol. At round $i$, the $i^{th}$ processor will broadcast a (random) message based on the first $i-1$ broadcast messages. After every processor broadcasts her messages, the final output $\in \{0,1\}$ is a deterministic function of all the broadcast messages. We do not limit to protocols with unbiased output (i.e., the probability of the output being 1 is 1/2).

**Definition 6.3.1** ($(n, X_0)$-Coin-tossing protocols)**.** *For any $n \in \mathbb{N}^*$ and $X_0 \in [0,1]$, an $(n, X_0)$-coin-tossing protocol is an $n$-round coin-tossing protocol among $n$ processors, where the expectation of the output is $X_0$.*

We often refer to the expected output $X_0$ as the *color* of the protocol. The *insecurity* of a coin-tossing protocol is the maximum change (in terms of statistical distance) that the adversary can cause to the distribution of the output of the protocol.

In this work, threshold protocols will be very useful examples, which are defined as follows.

**Definition 6.3.2** ($(n,t)$-Threshold protocol)**.** *In an $(n,t)$-threshold protocol, denoted by $\pi^{n,t}$, each processor broadcasts an (independently) uniform bit. The output is 1 if the total number of 1-message $\geqslant t$.*[5] *In particular, when $n$ is odd and $t = \frac{n+1}{2}$, this is the majority protocol.*

[5]↑Here, $t \in \{0, 1, \ldots, n+1\}$.

### 6.3.2 Adversarial Setting

In this work, we consider *Byzantine adaptive* adversaries. Such an adversary will eavesdrop on the execution of the protocol. After every round, it will decide whether to corrupt the processor, who is going to speak next. Once a processor is corrupted, the adversary takes full control and fixes the message that she is going to send. We will focus on such adversaries that corrupt (at most) *one* processor.

## 6.4 A Geometric Perspective

In this section, we shall study the insecurity of coin-tossing protocols through a geometric perspective.

**Protocol tree.** For every coin-tossing processor protocol, we will think of it as a tree. Every edge represents a message, and the root denotes the beginning of the protocol. Therefore, every node $u$ on this tree represents a partial transcript of the protocol. And we can associate it with a color $x_u$ and a probability $p_u$, where $x_u$ is the expected output conditioned on partial transcript $u$, and $p_u$ is the probability that partial transcript $u$ happens. For an $(n, X_0)$-coin-tossing protocol, by our definition, its protocol tree shall have depth $n$, and the color at the root shall be $X_0$.

**Attack.** A Byzantine adaptive adversary that corrupts at most one processor can be viewed equivalently as a collection of edges $\{(u_i, v_i)\}$, where $u_i$ is the parent of $v_i$. This implies that when partial transcript $u_i$ happens, the attacker intervenes and fixes the next message to be $v_i$. Since this attacker corrupts at most one processor during the entire collection of the protocol, this collection of edges must be prefix-free. That is, no parent node of an edge is on the path from the root to other edges.

Given a protocol tree $\pi$, let an attack strategy $\tau$ be the collection of edges $\{(u_i, v_i)\}$, where $u_i$ is the parent of $v_i$. We define the following score function.

**Definition 6.4.1.** $\mathsf{Score}(\pi, \tau) := \sum_{(u_i,v_i)\in\tau} p_{u_i} \cdot |x_{u_i} - x_{v_i}|$.

That is, $\mathsf{Score}(\pi, \tau)$ is the average of the absolute change in color the attacker $\tau$ causes. Intuitively, it represents the vulnerability of protocol $\pi$ in the presence of the attack $\tau$. Furthermore, for any protocol $\pi$, let us define

$$\mathsf{Score}(\pi) := \sup_{\tau} \mathsf{Score}(\pi, \tau).$$

Intuitively, $\mathsf{Score}(\pi)$ represents the score of the most devastating attacks on protocol $\pi$. Finally, we define

$$C_n(X_0) := \inf_{\pi} \mathsf{Score}(\pi),$$

where the infimum is taken over all $(n, X_0)$-coin-tossing protocols $\pi$. Intuitively, $C_n(X_0)$ represents the score of the optimal protocol against the most devastating attack among all protocols with $n$ processors and color $X_0$.

**Remark 6.4.1.** *We remark that for a protocol $\pi$, the deviation (to the distribution of the output) an attack $\tau$ causes is not exactly $\mathsf{Score}(\pi, \tau)$. However, one can always bi-partition the set $\tau$ as $\tau_0$ and $\tau_1$. $\tau_0$ will consist of all edges $(u_i, v_i)$ that decrease the expected output, i.e., $x_{u_i} \geqslant x_{v_i}$, while $\tau_1$ will consist of all edges $(u_i, v_i)$ that increase the expected output, i.e., $x_{u_i} < x_{v_i}$. Consequently, the summation of the deviations caused by attack $\tau_0$ and $\tau_1$ shall be $\mathsf{Score}(\pi, \tau)$. Therefore, there must exist an attack that deviates the protocol by $\mathsf{Score}(\pi, \tau)/2$. In light of this, for any $(n, X_0)$-coin-tossing protocol, there must exist an attack that deviates the protocol by $C_n(X_0)/2$. Hence, any $(n, X_0)$-coin-tossing protocol is (at least) $C_n(X_0)/2$ insecure.*

### 6.4.1 Geometric Transformation of $C_n$

In this section, we shall see how we can (inductively) construct $C_n$ from a geometric perspective.

Let us start with the simplest case $n = 1$. If $X_0 = 0$ or 1, the output is independent of the message and is always fixed. Hence, the score is always 0. If $X_0 \in (0, 1/2]$, the attack with the highest score is to fix the message such that the output is fixed to be 1. Hence,

the score is $1 - X_0$. Similarly, when $X_0 \in (1/2, 1)$, the score is $X_0$. Consequently, $C_1$ is the following curve.

$$C_1(x) = \begin{cases} 0 & x \in \{0, 1\} \\ 1 - x & x \in (0, 1/2] \\ x & x \in (1/2, 1) \end{cases}$$

Next, suppose we have curve $C_n$, we shall construct the next curve $C_{n+1}$. Let us use Figure 6.3 as an intuitive example to understand how to construct $C_{n+1}(x)$ from $C_n$.

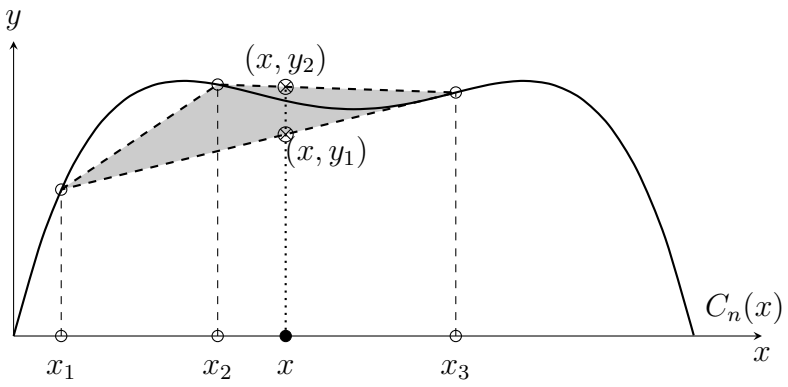



**Figure 6.3.** An intuitive example of the geometric transformation

Let $\pi$ be an $(n + 1, x)$-coin-tossing protocol. Suppose there are three possible messages that the first processor might send, namely $m_1$, $m_2$, and $m_3$. Conditioned on the first message being $m_1$, $m_2$, and $m_3$, the expected output is $x_1$, $x_2$, and $x_3$, respectively. The probability of the first message being $m_1$, $m_2$, and $m_3$, are $p_1$, $p_2$, and $p_3$, respectively. Note that after the first processor sends message $m_i$, the remaining protocol $\pi_i$ becomes a $(n, x_i)$-coin-tossing protocol.

An adaptive adversary that corrupts at most one processor has four choices for the first processor. Either it can carry out the attack now by fixing the first processor's message to be $m_i$, for $i \in \{1, 2, 3\}$, or it can defer the attack to subprotocols $\pi_1$, $\pi_2$, and $\pi_3$. If it fixes the first processor's message to be $m_i$, this will increase the score by $|x_i - x|$. On the other

hand, if it defers the attack to each subprotocol, by the definition of curve $C_n$, it can ensure a score of (at least) $C_n(x_i)$ in subprotocol $\pi_i$. Overall, it ensures a score of (at least)

$$p_1 \cdot C_n(x_1) + p_2 \cdot C_n(x_2) + p_3 \cdot C_n(x_3).$$

Note that it must hold that $x = p_1x_1 + p_2x_2 + p_3x_3$. Therefore, $p_1 \cdot C_n(x_1) + p_2 \cdot C_n(x_2) + p_3 \cdot C_n(x_3)$ must lie between $y_1$ and $y_2$ in Figure 6.3.

The most devastating attack will do the attack based on which strategy results in the highest score, which is

$$\max\left(|x - x_1|, |x - x_2|, |x - x_3|, p_1 \cdot C_n(x_1) + p_2 \cdot C_n(x_2) + p_3 \cdot C_n(x_3)\right).$$

The optimal protocol shall, however, pick $x_1, \ldots, x_\ell$ and $p_1, \ldots, p_\ell$ accordingly to minimize the above quantity. Therefore, by our definition,

$$C_{n+1}(x) := \inf_{\substack{x_1,\ldots,x_\ell \in [0,1] \\ p_1,\ldots,p_\ell \in [0,1] \\ p_1+\cdots+p_\ell=1 \\ p_1x_1+\cdots+p_\ell x_\ell = x}} \max\left(|x - x_1|, \ldots, |x - x_\ell|, \sum_{i=1}^{\ell} p_i \cdot C_n(x_i)\right).$$

For convenience, let us define geometric transformation $T$, which takes any curve $C$ on $[0, 1]$ as input, and outputs a curve $T(C)$ defined as

$$T(C)(x) := \inf_{\substack{x_1,\ldots,x_\ell \in [0,1] \\ p_1,\ldots,p_\ell \in [0,1] \\ p_1+\cdots+p_\ell=1 \\ p_1x_1+\cdots+p_\ell x_\ell = x}} \max\left(|x - x_1|, \ldots, |x - x_\ell|, \sum_{i=1}^{\ell} p_i \cdot C(x_i)\right).$$

Hence, by our definition, $C_{n+1}$ is exactly $T(C_n)$.

## 6.5 Tight Bounds on $C_n$ and the Implications

In this section, we shall first prove a tight lower bound on the curve $C_n$.

We define our lower bound curve $L_n$ through threshold protocols. Recall that an $(n, t)$-threshold protocol $\pi^{n,t}$ is a protocol where each processor broadcast an (independent) uniform

bit. The final output is 1 if the number of 1-message is $\geqslant t$. Trivially, the color of $(n,t)$-threshold protocol $\pi^{n,t}$ is

$$\mathsf{Color}\left(\pi^{n,t}\right) = 2^{-n} \cdot \left(\sum_{i=t}^{n} \binom{n}{i}\right).$$

We argue that the score of $\pi^{n,t}$ is

$$\mathsf{Score}\left(\pi^{n,t}\right) = 2^{-n} \cdot \binom{n-1}{t-1}.$$

To see this, note that, without of loss of generality, we can assume that anytime the adversary fixes a message, it fixes that message to be 1.[6] Moreover, which message that the adversary fixes does not matter; effectively, the output will be 1 if and only if the rest $n-1$ messages contain $\geqslant t-1$ 1-message. Therefore, by fixing one message to be 1, it changes the expected output of the protocol to be $2^{-(n-1)} \cdot \left(\sum_{i=t-1}^{n-1} \binom{n-1}{i}\right)$. Easily, one can verify that $2^{-(n-1)} \cdot \left(\sum_{i=t-1}^{n-1} \binom{n-1}{i}\right) - 2^{-n} \cdot \left(\sum_{i=t}^{n} \binom{n}{i}\right) = 2^{-n} \cdot \binom{n-1}{t-1}$.

For a $n$-processor threshold protocol, threshold $t \in \{n+1, n, \ldots, 0\}$.[7] We define the lower bound curve $L_n$ as follows.

**Definition 6.5.1.** *For every $n \in \mathbb{N}^*$, let $L_n$ be the curve that linearly connects points*

$$P_{n,t} := \left(\mathsf{Color}\left(\pi^{n,t}\right),\ \mathsf{Score}\left(\pi^{n,t}\right)\right) = \left(2^{-n} \cdot \left(\sum_{i=t}^{n} \binom{n}{i}\right),\quad 2^{-n} \cdot \binom{n-1}{t-1}\right)$$

*for $t = n+1, n, \ldots, 0$. That is, $L_n$ linearly interpolates all the points defined by the color and score of $(n,t)$-threshold protocols.*

As an example, $L_1$ is shown in Figure 6.4.

In particular, we have the following theorem regarding the curve $L_n$ and the curve $C_n$.

**Theorem 6.5.1.** *For all $n \in \mathbb{N}^*$, $L_n \preceq C_n$.*

**Remark 6.5.1.** *Note that, by the definition of $C_n$, we have*

$$C_n\left(\mathsf{Color}\left(\pi^{n,t}\right)\right) := \inf_{\pi}\ \mathsf{Score}(\pi) \leqslant \mathsf{Score}\left(\pi^{n,t}\right).$$

[6]↑For any node $u$, let its two children node be $v_0$ and $v_1$. Since every message is a uniform bit for threshold protocol, it must hold that $|x_u - x_{v_0}| = |x_u - x_{v_1}|$. Therefore, whether the attack picks edge $(u, v_0)$ or $(u, v_1)$ does not change the score.

[7]↑When $t = n+1$, the color is 0, and when $t = 0$, the color is 1.

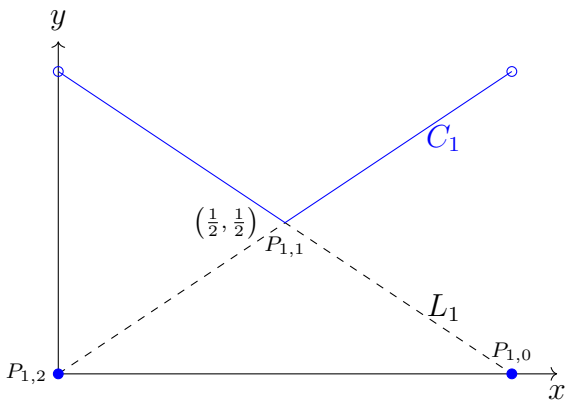



**Figure 6.4.** The (black) dashed curve is $L_1$ and the (blue) solid curve is $C_1$. Note that $P_{1,t}$ corresponds to the point defined by $(1,t)$-threshold protocol.

*On the other hand, by Theorem 6.5.1,*

$$C_n\left(\mathsf{Color}\left(\pi^{n,t}\right)\right) \geqslant L_n\left(\mathsf{Color}\left(\pi^{n,t}\right)\right) = \mathsf{Score}\left(\pi^{n,t}\right).$$

*Therefore,* $C_n\left(\mathsf{Color}\left(\pi^{n,t}\right)\right) = \mathsf{Score}\left(\pi^{n,t}\right)$*. That is, points* $P_{n,t}$ *is on the curve* $C_n$ *as well. This also implies that threshold protocol is the protocol that minimizes the score function.*

We defer the proof of Theorem 6.5.1 to Section 6.5.1. Let us first discuss the implications of this theorem. We have the following corollaries.

**Corollary 6.5.2** (Threshold protocols)**.** *For any* $n \in \mathbb{N}^*$ *and* $X_0 \in [0,1]$ *such that* $X_0 = 2^{-n} \cdot \left(\sum_{i=t}^{n} \binom{n}{i}\right)$ *for some* $t \in \{0,1,\ldots,n+1\}$*. The insecurity of* $(n,t)$*-threshold protocol is at most two times the insecurity of the least insecure* $(n,X_0)$*-coin-tossing protocols.*

This corollary is immediate from Theorem 6.5.1. This is because the insecurity of threshold protocol $\pi^{n,t}$ is exactly $\mathsf{Score}\left(\pi^{n,t}\right)$; for any other $\left(n,\mathsf{Color}\left(\pi^{n,t}\right)\right)$-coin-tossing protocol, in light of Remark 6.4.1, we know its insecurity is at least

$$C_n\left(\mathsf{Color}\left(\pi^{n,t}\right)\right)/2 \geqslant L_n\left(\mathsf{Color}\left(\pi^{n,t}\right)\right)/2 = \mathsf{Score}\left(\pi^{n,t}\right)/2.$$

Therefore, the insecurity of the threshold protocol is at most two times the insecurity of the optimal protocol.

**Corollary 6.5.3** (Non-threshold protocols)**.** *For an arbitrary color $X_0 \in (0,1)$ that does not correspond to any threshold protocol, we can consider a linear combination of threshold protocols. Specifically, suppose $\mathsf{Color}\left(\pi^{n,t}\right) < X_0 < \mathsf{Color}\left(\pi^{n,t-1}\right)$, consider an $(n+1, X_0)$-coin-tossing protocol as follows. The first processor sends a bit. If this bit is 0, the rest $n$ processors execute the $(n,t)$-threshold protocol; if this bit is 1, the rest $n$ processors execute the $(n,t-1)$-threshold protocol. The probability of this bit being 0 is defined to be*

$$\frac{\mathsf{Color}\left(\pi^{n,t-1}\right) - X_0}{\mathsf{Color}\left(\pi^{n,t-1}\right) - \mathsf{Color}\left(\pi^{n,t}\right)}.$$

*For $X_0$ that is not negligibly close to 0 or 1, the insecurity of this protocol is at most $4 + o(1)$ times the insecurity of the least insecure $(n+1, X_0)$-protocol.*

Without loss of generality, assume $X_0 < 1/2$. Therefore, $t > n/2$. One can easily see that the insecurity of this protocol is bounded by

$$\max\left(\mathsf{Color}\left(\pi^{n,t-1}\right) - \mathsf{Color}\left(\pi^{n,t}\right)\ ,\ \frac{\mathsf{Score}\left(\pi^{n,t-1}\right) + \mathsf{Score}\left(\pi^{n,t}\right)}{2}\right),$$

which is bounded by $2^{-n} \cdot \binom{n}{t-1}$. On the other hand, Theorem 6.5.1 says that every $(n+1, X_0)$-coin-tossing protocol is at least $L_{n+1}(X_0)/2$-insecure, which is at least $2^{-(n+2)}\binom{n+1}{t}$. When $X_0$ is non-negligibly bounded away from 0 and 1, by Chernoff's bound, we must have $|t - n/2| \leqslant \sqrt{n}\log n$. Consequently, $\binom{n}{t-1}$ and $\binom{n+1}{t}$ are $(1+\mathrm{o}(1))$ approximation to each other. Hence, the insecurity of this protocol is (at most) $(4+\mathrm{o}(1))$-approximate of the optimal $(n+1, X_0)$-protocol.

### 6.5.1 Proof of Theorem 6.5.1

To prove this theorem, it suffices to prove the following claims.

**Claim 6.5.1.** *If $A \preccurlyeq B$, then $T(A) \preccurlyeq T(B)$.*

**Claim 6.5.2.** *$L_{n+1} = T\left(L_n\right)$.*

*Proof of Theorem 6.5.1 using Claim 6.5.1 and Claim 6.5.2.* We prove this theorem inductively. The base case $n = 1$ is trivial (See Figure 6.4).

Suppose the statement is correct for $n$, i.e., $L_n \preceq C_n$. Then we have

$$L_n \preceq C_n \xRightarrow{\text{Claim 6.5.1}} T(L_n) \preceq T(C_n) \xRightarrow{\text{Claim 6.5.2}} L_{n+1} \preceq C_{n+1}$$

This completes the proof. □

Next we prove Claim 6.5.1 and Claim 6.5.2.

*Proof of Claim 6.5.1.* Since $A \preccurlyeq B$, for all $x$, $x_1, \ldots, x_\ell$, and $p_1, \ldots, p_\ell$, we have

$$\max\left(|x - x_1|, \ldots, |x - x_\ell|, \sum_{i=1}^{\ell} p_i \cdot A(x_i)\right) \leqslant \max\left(|x - x_1|, \ldots, |x - x_\ell|, \sum_{i=1}^{\ell} p_i \cdot B(x_i)\right).$$

Therefore, by definition, for all $x$, $T(A)(x) \leqslant T(B)(x)$, or equivalently $T(A) \preccurlyeq T(B)$. □

Before we prove Claim 6.5.2, the following claim will be useful.

**Claim 6.5.3.** *Let $U$ be an arbitray concave curve. Suppose $0 \leqslant x_0 < x < x_2 \leqslant 1$ satisfies that*

$$x - x_0 = x_1 - x = \frac{U(x_0) + U(x_1)}{2},$$

*Then $T(U)(x) = \frac{U(x_0)+U(x_1)}{2}$. That is, $x_0$ and $x_1$ witness the transformation $T$ of $U$ at $x$.*

*Proof of Claim 6.5.3.* To see this, let us use Figure 6.5 for intuition. In Figure 6.5, $U(x)$ is a concave curve and the choice of $x_0$ and $x_1$ satisfies that $x - x_0 = x_1 - x = \frac{U(x_0)+U(x_1)}{2}$. Recall that

$$T(U)(x) := \inf_{\substack{x'_1,\ldots,x'_\ell \in [0,1] \\ p_1,\ldots,p_\ell \in [0,1] \\ p_1+\cdots+p_\ell = 1 \\ p_1 x'_1 + \cdots + p_\ell x'_\ell = x}} \max\left(|x - x'_1|, \ldots, |x - x'_\ell|, \sum_{i=1}^{\ell} p_i \cdot D(x'_i)\right).$$

By definition, clearly, $T(U)(x) \leqslant \frac{U(x_0)+U(x_1)}{2}$. To prove the other direction, we need to show that, for any choices of $x'_1, x'_2, \ldots, x'_\ell$ and $p_1, p_2, \ldots, p_\ell$, we have

$$\frac{U(x_0) + U(x_1)}{2} \leqslant \max\left(|x - x'_1|, \ldots, |x - x'_\ell|, \sum_{i=1}^{\ell} p_i \cdot U(x'_i)\right)$$

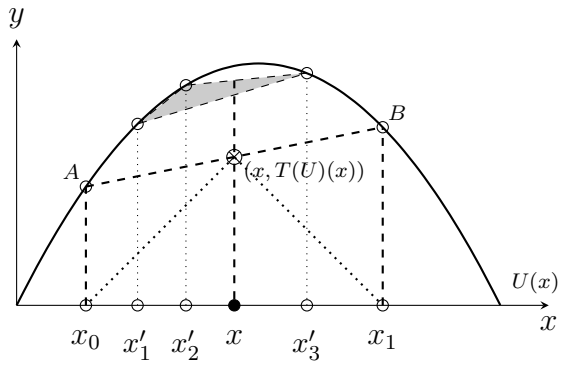



**Figure 6.5.** The geometric transformation of curve $U(x)$. Intuitively, if $x'_1$, $x'_2$, and $x'_3$ are $\in (x_0, x_1)$, the shaded region is always above line segment $AB$ by the concaveness of $U$.

Firstly, if there exists an $x'_i$ such that $|x - x'_i| \geqslant |x_1 - x|$, then the statement trivially holds. Next, if for all $i$, $|x - x'_i| \leqslant |x_1 - x|$, then by the concaveness of curve $U$,

$$\frac{1}{2} \cdot (U(x_1) + U(x_2)) \leqslant \sum_{i=1}^{\ell} p_i \cdot U(x'_i).$$

This completes the proof. □

Now, we prove Claim 6.5.2.

*Proof of Claim 6.5.2.* Recall that $L_n$ is the curve that linearly connects points $P_{n,n+1}, P_{n,n}, \ldots, P_{n,1}, P_{n,0}$, where

$$P_{n,t} := \left( 2^{-n} \cdot \left( \sum_{i=t}^{n} \binom{n}{i} \right) \quad , \quad 2^{-n} \cdot \binom{n-1}{t-1} \right).$$

Let us first observe some properties of $L_n$.

**Claim 6.5.4.** *$L_n$ is a concave curve and the slope of any line segment of $L_n$ is $\in [-1, 1]$.*

*Proof of Claim 6.5.4.* Easily, we can verify that the slope of line segment $P_{n,t}P_{n,t-1}$ is

$$\frac{2^{-n} \cdot \binom{n-1}{t-1} - 2^{-n} \cdot \binom{n-1}{t-2}}{2^{-n} \cdot \left( \sum_{i=t}^{n} \binom{n}{i} \right) - 2^{-n} \cdot \left( \sum_{i=t-1}^{n} \binom{n}{i} \right)} = \frac{2t - 2 - n}{n}.$$

Since the slope of $P_{n,t}P_{n,t-1}$ decreases as $t$ decreases, this proves that $L_n$ is concave. Moreover, for any $t \in \{n+1, \ldots, 1\}$, the slope of $P_{n,t}P_{n,t-1}$ is $\in [-1, 1]$. □

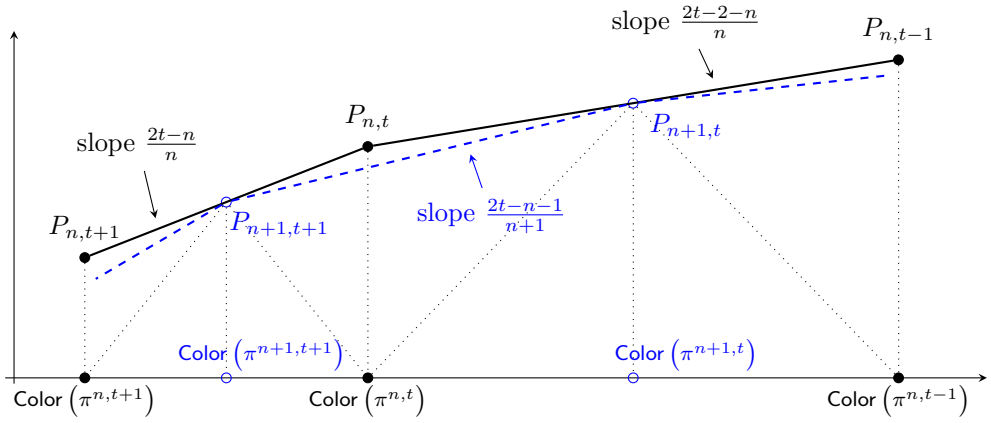



**Figure 6.6.** The relation between (black solid) $L_n$ and (blue dashed) $L_{n+1}$. The geometric transformation of $L_n$ is exactly $L_{n+1}$.

**Claim 6.5.5.** *$P_{n+1,t}$ is the middle point of $P_{n,t}$ and $P_{n,t-1}$.*

*Proof of Claim 6.5.5.* One just need to verify that

$$2^{-(n+1)}\left(\sum_{i=t}^{n+1}\binom{n+1}{i}\right)=\frac{1}{2}\cdot\left[2^{-n}\left(\sum_{i=t}^{n}\binom{n}{i}\right)+2^{-n}\left(\sum_{i=t-1}^{n}\binom{n}{i}\right)\right],$$

and

$$2^{-(n+1)}\cdot\binom{n}{t-1}=\frac{1}{2}\cdot\left[2^{-n}\cdot\binom{n-1}{t-1}+2^{-n}\cdot\binom{n-1}{t-2}\right]. \qquad \square$$

Now, let us prove $L_{n+1}=T(L_n)$ with all the claims that we have proven. It suffices to verify $L_{n+1}(x)=T(L_n)(x)$ for all $x\in(0,1)$. In light of Claim 6.5.4 and Claim 6.5.5, we know the relation between $L_n$ and $L_{n+1}$ looks like Figure 6.6.

We first verify it at $x=\mathsf{Color}\left(\pi^{n+1,t}\right)$. In this case, we can set $x_0=\mathsf{Color}\left(\pi^{n,t}\right)$ and $x_1=\mathsf{Color}\left(\pi^{n,t-1}\right)$. One can verify that

$$\mathsf{Color}\left(\pi^{n+1,t}\right)-x_0=x_1-\mathsf{Color}\left(\pi^{n+1,t}\right)=\mathsf{Score}\left(\pi^{n+1,t}\right),$$

and

$$\frac{L_n(x_0)+L_n(x_1)}{2}=\frac{\mathsf{Score}\left(\pi^{n,t}\right)+\mathsf{Score}\left(\pi^{n,t-1}\right)}{2}=\mathsf{Score}\left(\pi^{n+1,t}\right).$$

Hence, by Claim 6.5.3,

$$T(L_n)\left(\mathsf{Color}\left(\pi^{n+1,t}\right)\right) = \mathsf{Score}\left(\pi^{n+1,t}\right) = L_{n+1}\left(\mathsf{Color}\left(\pi^{n+1,t}\right)\right).$$

Next, we verify $L_{n+1} = T(L_n)$ for some $x$ such that $\mathsf{Color}\left(\pi^{n+1,t+1}\right) < x < \mathsf{Color}\left(\pi^{n+1,t}\right)$. By Claim 6.5.3, it suffices to set $x_0 = x - L_{n+1}(x)$ and $x_1 = x + L_{n+1}(x)$ and verify that

$$\frac{L_n(x_0) + L_n(x_1)}{2} = L_{n+1}(x).$$

Note that

$$x_0 \in \left[\mathsf{Color}\left(\pi^{n,t+1}\right), \mathsf{Color}\left(\pi^{n,t}\right)\right] \quad \text{and} \quad x_1 \in \left[\mathsf{Color}\left(\pi^{n,t}\right), \mathsf{Color}\left(\pi^{n,t-1}\right)\right].$$

One can verify that this is indeed correct. □

# 7. CONCLUSIONS

In this work, we have furthered our understanding of the optimal constructions in non-malleable codes, optimal-fair coin-tossing, and collective coin-tossing. Many questions, however, still remain open. In this section, we list a few open problems.

## 7.1 Non-malleable Codes

There are still many simple tampering families where high-rate constructions are unknown. One example is the affine tampering family over $\mathbb{F}_2$. That is, every output bit is the parity of some input bits. Interestingly, only rate-0 construction [CL17] are currently known for this seemingly simple tampering family.

**Open Problem 1.** *Can we find an explicit rate-1 non-malleable code for affine tampering over $\mathbb{F}_2$?*

Although we have found constant-rate two-split-state non-malleable code [AO20] recently, the rate of their construction is still quite low. It remains an intriguing open problem whether we can amplify the rate of their construction through the technique developed in this work.

**Open Problem 2.** *Can we find an explicit two-split-state non-malleable code with an (explicit) high rate?*

## 7.2 Optimal-fair Coin-tossing

Although we showed that ideal invocations of (incomplete) functionalities are useless for fair coin-tossing, we have not proven a black-box separation result for it. That is, we did not present a collection of oracles such that (1) $f$ can be securely realized using these oracles and (2) optimal-fair coin-tossing does not exist relative to them. Such a problem is extremely challenging and still remains one of the most crucial open problems.

**Open Problem 3.** *Could we prove that optimal-fair coin-tossing and ideal (incomplete) functionalities are black-box separated?*

### 7.3 Collective Coin-tossing

The recent work by Haitner and Karidi-Heller [HKH20] finally settled the conjecture by Ben-Or and Linial [BL85] in the positive. That is, for any collective coin-tossing protocol among $n$ parties, an adaptive adversary can corrupt $\mathcal{O}\left(\sqrt{n} \cdot \mathsf{polylog}\left(n\right)\right)$ parties to nearly fix the output of the protocol. However, suppose we want to ensure that the expected output protocol to be either $\leqslant \varepsilon$ or $\geqslant (1-\varepsilon)$ for a constant $\varepsilon$. How many processors does one need to corrupt? Haitner and Karidi-Heller's attack needs to corrupt $\mathcal{O}\left(\sqrt{n} \cdot \mathsf{polylog}\left(n\right)\right)$ parties, while it is conjectured that $\mathcal{O}\left(\sqrt{n}\right)$ suffices. Therefore, it remains open whether one could find a more fine-grained attack (compared to [HKH20]).

**Open Problem 4.** *For any constant $\varepsilon$, could we find an attack that only corrupts $\mathcal{O}\left(\sqrt{n}\right)$ parties to ensure that the expected output is either $\leqslant \varepsilon$ or $\geqslant (1-\varepsilon)$?*

# Bibliography


[AAG+16] Divesh Aggarwal, Shashank Agrawal, Divya Gupta, Hemanta K. Maji, Omkant Pandey, and Manoj Prabhakaran. Optimal computational split-state non-malleable codes. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A: 13th Theory of Cryptography Conference, Part II*, volume 9563 of *Lecture Notes in Computer Science*, pages 393–417, Tel Aviv, Israel, January 10–13, 2016. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-49099-0_15. 28, 79, 80, 81

[ABC+85] Baruch Awerbuch, Manuel Blum, Benny Chor, Shafi Goldwasser, and Silvio Micali. How to implement bracha's o (log n) byzantine agreement algorithm. *Unpublished manuscript*, 1985. 16, 34, 98, 99

[ABMO15] Gilad Asharov, Amos Beimel, Nikolaos Makriyannis, and Eran Omri. Complete characterization of fairness in secure two-party computation of Boolean functions. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015: 12th Theory of Cryptography Conference, Part I*, volume 9014 of *Lecture Notes in Computer Science*, pages 199–228, Warsaw, Poland, March 23–25, 2015. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-46494-6_10. 33, 98

[ACJ17] Prabhanjan Ananth, Arka Rai Choudhuri, and Abhishek Jain. A new approach to round-optimal secure multiparty computation. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part I*, volume 10401 of *Lecture Notes in Computer Science*, pages 468–499, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Heidelberg, Germany. doi:10.1007/978-3-319-63688-7_16. 12

[ADKO15] Divesh Aggarwal, Yevgeniy Dodis, Tomasz Kazana, and Maciej Obremski. Non-malleable reductions and applications. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th Annual ACM Symposium on Theory of Computing*, pages 459–468, Portland, OR, USA, June 14–17, 2015. ACM Press. doi:10.1145/2746539.2746544. 24, 25, 31, 76, 78, 79, 96

[ADL14] Divesh Aggarwal, Yevgeniy Dodis, and Shachar Lovett. Non-malleable codes from additive combinatorics. In David B. Shmoys, editor, *46th Annual ACM Symposium on Theory of Computing*, pages 774–783, New York, NY, USA, May 31 – June 3, 2014. ACM Press. doi:10.1145/2591796.2591804. 25, 26, 76, 79, 80, 81

[ADN+19] Divesh Aggarwal, Nico Döttling, Jesper Buus Nielsen, Maciej Obremski, and Erick Purwanto. Continuous non-malleable codes in the 8-split-state model. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019, Part I*, volume 11476 of *Lecture Notes in Computer Science*, pages 531–561, Darmstadt, Germany, May 19–23, 2019. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-17653-2_18. 30

[AGM+15a] Shashank Agrawal, Divya Gupta, Hemanta K. Maji, Omkant Pandey, and Manoj Prabhakaran. Explicit non-malleable codes against bit-wise tampering and permutations. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *Advances in Cryptology – CRYPTO 2015, Part I*, volume 9215 of *Lecture Notes in Computer Science*, pages 538–557, Santa Barbara, CA, USA, August 16–20, 2015. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-47989-6_26. 28

[AGM+15b] Shashank Agrawal, Divya Gupta, Hemanta K. Maji, Omkant Pandey, and Manoj Prabhakaran. A rate-optimizing compiler for non-malleable codes against bit-wise tampering and permutations. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015: 12th Theory of Cryptography Conference, Part I*, volume 9014 of *Lecture Notes in Computer Science*, pages 375–397, Warsaw, Poland, March 23–25, 2015. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-46494-6_16. 28, 43, 46

[AKO17] Divesh Aggarwal, Tomasz Kazana, and Maciej Obremski. Inception makes non-malleable codes stronger. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017: 15th Theory of Cryptography Conference, Part II*, volume 10678 of *Lecture Notes in Computer Science*, pages 319–343, Baltimore, MD, USA, November 12–15, 2017. Springer, Heidelberg, Germany. doi:10.1007/978-3-319-70503-3_10. 30

[AL93] Miklós Ajtai and Nathan Linial. The influence of large coalitions. *Combinatorica*, 13(2):129–145, 1993. 36, 37

[ALR13] Gilad Asharov, Yehuda Lindell, and Tal Rabin. A full characterization of functions that imply fair coin tossing and ramifications to fairness. In Amit Sahai, editor, *TCC 2013: 10th Theory of Cryptography Conference*, volume 7785 of *Lecture Notes in Computer Science*, pages 243–262, Tokyo, Japan, March 3–6, 2013. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-36594-2_14. 33, 98

[AN90] Noga Alon and Moni Naor. Coin-flipping games immune against linear-sized coalitions (extended abstract). In *31st Annual Symposium on Foundations of Computer Science*, pages 46–54, St. Louis, MO, USA, October 22–24, 1990. IEEE Computer Society Press. doi:10.1109/FSCS.1990.89523. 37

[AO16] Bar Alon and Eran Omri. Almost-optimally fair multiparty coin-tossing with nearly three-quarters malicious. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B: 14th Theory of Cryptography Conference, Part I*, volume 9985 of *Lecture Notes in Computer Science*, pages 307–335, Beijing, China, October 31 – November 3, 2016. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-53641-4_13. 33, 98

[AO20] Divesh Aggarwal and Maciej Obremski. A constant rate non-malleable code in the split-state model. In *61st Annual Symposium on Foundations of Computer Science*, pages 1285–1294, Durham, NC, USA, November 16–19, 2020. IEEE Computer Society Press. doi:10.1109/FOCS46700.2020.00122. 28, 158

[AP13] Shashank Agrawal and Manoj Prabhakaran. On fair exchange, fair coins and fair sampling. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 259–276, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-40041-4_15. 98

[Ash14] Gilad Asharov. Towards characterizing complete fairness in secure two-party computation. In Yehuda Lindell, editor, *TCC 2014: 11th Theory of Cryptography Conference*, volume 8349 of *Lecture Notes in Computer Science*, pages 291–316, San Diego, CA, USA, February 24–26, 2014. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-54242-8_13. 33, 98

[Asp97] James Aspnes. Lower bounds for distributed coin-flipping and randomized consensus. In *29th Annual ACM Symposium on Theory of Computing*, pages 559–568, El Paso, TX, USA, May 4–6, 1997. ACM Press. doi:10.1145/258533.258649. 39, 136, 139, 144

[Asp98] James Aspnes. Lower bounds for distributed coin-flipping and randomized consensus. *J. ACM*, 45(3):415–450, 1998. doi:10.1145/278298.278304. 39, 136, 139, 144

[Azu67] Kazuoki Azuma. Weighted sums of certain dependent random variables. *Tohoku Mathematical Journal, Second Series*, 19(3):357–367, 1967. 38, 142

[BCL+20] Marshall Ball, Eshan Chattopadhyay, Jyun-Jie Liao, Tal Malkin, and Li-Yang Tan. Non-malleability against polynomial tampering. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology – CRYPTO 2020, Part III*, volume 12172 of *Lecture Notes in Computer Science*, pages 97–126, Santa Barbara, CA, USA, August 17–21, 2020. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-56877-1_4. 29

[BD84] Andrei Z. Broder and Danny Dolev. Flipping coins in many pockets (byzantine agreement on uniformly random values). In *25th Annual Symposium on Foundations of Computer Science*, pages 157–170, Singer Island, Florida, October 24–26, 1984. IEEE Computer Society Press. doi:10.1109/SFCS.1984.715912. 16, 34, 98, 99

[BDG+18] Marshall Ball, Dana Dachman-Soled, Siyao Guo, Tal Malkin, and Li-Yang Tan. Non-malleable codes for small-depth circuits. In Mikkel Thorup, editor, *59th Annual Symposium on Foundations of Computer Science*, pages 826–837, Paris, France, October 7–9, 2018. IEEE Computer Society Press. doi:10.1109/FOCS.2018.00083. 25, 29, 47

[BDK+19] Marshall Ball, Dana Dachman-Soled, Mukul Kulkarni, Huijia Lin, and Tal Malkin. Non-malleable codes against bounded polynomial time tampering. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019, Part I*, volume 11476 of *Lecture Notes in Computer Science*, pages 501–530, Darmstadt, Germany, May 19–23, 2019. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-17653-2_17. 29

[BDKM16] Marshall Ball, Dana Dachman-Soled, Mukul Kulkarni, and Tal Malkin. Non-malleable codes for bounded depth, bounded fan-in circuits. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 881–908, Vienna, Austria, May 8–12, 2016. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-49896-5_31. 25, 28, 29, 42, 47, 52

[BDKM18] Marshall Ball, Dana Dachman-Soled, Mukul Kulkarni, and Tal Malkin. Non-malleable codes from average-case hardness: $\mathsf{AC}^0$, decision trees, and streaming space-bounded tampering. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part III*, volume 10822 of *Lecture Notes in Computer Science*, pages 618–650, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Heidelberg, Germany. doi:10.1007/978-3-319-78372-7_20. 29, 47

[BDKM20] Marshall Ball, Dana Dachman-Soled, Mukul Kulkarni, and Tal Malkin. Limits to non-malleability. In Thomas Vidick, editor, *ITCS 2020: 11th Innovations in Theoretical Computer Science Conference*, volume 151, pages 80:1–80:32, Seattle, WA, USA, January 12–14, 2020. LIPIcs. doi:10.4230/LIPIcs.ITCS.2020.80. 29

[Bea96] Donald Beaver. Correlated pseudorandomness and the complexity of private computations. In *28th Annual ACM Symposium on Theory of Computing*, pages 479–488, Philadephia, PA, USA, May 22–24, 1996. ACM Press. doi:10.1145/237814.237996. 11

[BEGH16] Mark Braverman, Klim Efremenko, Ran Gelles, and Bernhard Haeupler. Constant-rate coding for multiparty interactive communication is impossible. In Daniel Wichs and Yishay Mansour, editors, *48th Annual ACM Symposium on Theory of Computing*, pages 999–1010, Cambridge, MA, USA, June 18–21, 2016. ACM Press. doi:10.1145/2897518.2897563. 11

[BGP00] Mihir Bellare, Oded Goldreich, and Erez Petrank. Uniform generation of np-witnesses using an np-oracle. *Inf. Comput.*, 163(2):510–526, 2000. 138

[BGW19] Marshall Ball, Siyao Guo, and Daniel Wichs. Non-malleable codes for decision trees. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019, Part I*, volume 11692 of *Lecture Notes in Computer Science*, pages 413–434, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-26948-7_15. 29

[BHLT17] Niv Buchbinder, Iftach Haitner, Nissan Levi, and Eliad Tsfadia. Fair coin flipping: Tighter analysis and the many-party case. In Philip N. Klein, editor, *28th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2580–2600, Barcelona, Spain, January 16–19, 2017. ACM-SIAM. doi:10.1137/1.9781611974782.170. 33, 98

[BHMO18] Amos Beimel, Iftach Haitner, Nikolaos Makriyannis, and Eran Omri. Tighter bounds on multi-party coin flipping via augmented weak martingales and differentially private sampling. In Mikkel Thorup, editor, *59th Annual Symposium on Foundations of Computer Science*, pages 838–849, Paris, France, October 7–9, 2018. IEEE Computer Society Press. doi:10.1109/FOCS.2018.00084. 33

[BHP17] Zvika Brakerski, Shai Halevi, and Antigoni Polychroniadou. Four round secure computation without setup. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017: 15th Theory of Cryptography Conference, Part I*, volume 10677 of *Lecture Notes in Computer Science*, pages 645–677, Baltimore, MD, USA, November 12–15, 2017. Springer, Heidelberg, Germany. doi:10.1007/978-3-319-70500-2_22. 12

[BI64] John F Banzhaf III. Weighted voting doesn't work: A mathematical analysis. *Rutgers L. Rev.*, 19:317, 1964. 136

[BJB98] Ziv Bar-Joseph and Michael Ben-Or. A tight lower bound for randomized synchronous consensus. In Brian A. Coan and Yehuda Afek, editors, *17th ACM Symposium Annual on Principles of Distributed Computing*, pages 193–199, Puerto Vallarta, Mexico, June 28 – July 2, 1998. Association for Computing Machinery. doi:10.1145/277697.277733. 136

[BKK+92] Jean Bourgain, Jeff Kahn, Gil Kalai, Yitzhak Katznelson, and Nathan Linial. The influence of variables in product spaces. *Israel Journal of Mathematics*, 77(1):55–64, 1992. 36

[BL85] Michael Ben-Or and Nathan Linial. Collective coin flipping, robust voting schemes and minima of banzhaf values. In *26th Annual Symposium on Foundations of Computer Science*, pages 408–416, Portland, Oregon, October 21–23, 1985. IEEE Computer Society Press. doi:10.1109/SFCS.1985.15. 12, 17, 38, 136, 159

[BL89] Michael Ben-Or and Nathan Linial. Collective coin flipping. *Advances in Computing Research*, 5:91–115, 1989. 34, 136

[BL18] Fabrice Benhamouda and Huijia Lin. k-round multiparty computation from k-round oblivious transfer via garbled interactive circuits. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part II*, volume 10821 of *Lecture Notes in Computer Science*, pages 500–532, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Heidelberg, Germany. doi:10.1007/978-3-319-78375-8_17. 12

[BLOO11] Amos Beimel, Yehuda Lindell, Eran Omri, and Ilan Orlov. $1/p$-Secure multiparty computation without honest majority and the best of both worlds. In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 277–296, Santa Barbara, CA, USA, August 14–18, 2011. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-22792-9_16. 98

[Blu82] Manuel Blum. Coin flipping by telephone - A protocol for solving impossible problems. pages 133–137, 1982. 16, 34, 98, 99

[BM84] G. R. Blakley and Catherine Meadows. Security of ramp schemes. In G. R. Blakley and David Chaum, editors, *Advances in Cryptology – CRYPTO'84*, volume 196 of *Lecture Notes in Computer Science*, pages 242–268, Santa Barbara, CA, USA, August 19–23, 1984. Springer, Heidelberg, Germany. 42, 192

[BM09] Boaz Barak and Mohammad Mahmoody-Ghidary. Merkle puzzles are optimal - an $O(n^2)$-query attack on any key exchange from a random oracle. In Shai Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 374–390, Santa Barbara, CA, USA, August 16–20, 2009. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-03356-8_22. 107, 108, 114, 131

[BMR90] Donald Beaver, Silvio Micali, and Phillip Rogaway. The round complexity of secure protocols (extended abstract). In *22nd Annual ACM Symposium on Theory of Computing*, pages 503–513, Baltimore, MD, USA, May 14–16, 1990. ACM Press. doi:10.1145/100216.100287. 12

[BN93] Ravi B. Boppana and Babu O. Narayanan. The biased coin problem. In *25th Annual ACM Symposium on Theory of Computing*, pages 252–257, San Diego, CA, USA, May 16–18, 1993. ACM Press. doi:10.1145/167088.167164. 37

[BOO10] Amos Beimel, Eran Omri, and Ilan Orlov. Protocols for multiparty coin toss with dishonest majority. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 538–557, Santa Barbara, CA, USA, August 15–19, 2010. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-14623-7_29. 33, 98

[BS19] Saikrishna Badrinarayanan and Akshayaram Srinivasan. Revisiting non-malleable secret sharing. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019, Part I*, volume 11476 of *Lecture Notes in Computer Science*, pages 593–622, Darmstadt, Germany, May 19–23, 2019. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-17653-2_20. 31

[Can00] Ran Canetti. Security and composition of multiparty cryptographic protocols. *J. Cryptology*, 2000. 119

[CCG+20] Arka Rai Choudhuri, Michele Ciampi, Vipul Goyal, Abhishek Jain, and Rafail Ostrovsky. Round optimal secure multiparty computation from minimal assumptions. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020: 18th Theory of Cryptography Conference, Part II*, volume 12551 of *Lecture Notes in Computer Science*, pages 291–319, Durham, NC, USA, November 16–19, 2020. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-64378-2_11. 12

[CCHM19] Binyi Chen, Yilei Chen, Kristina Hostáková, and Pratyay Mukherjee. Continuous space-bounded non-malleable codes from stronger proofs-of-space. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019, Part I*, volume 11692 of *Lecture Notes in Computer Science*, pages 467–495, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-26948-7_17. 30

[CDF+08] Ronald Cramer, Yevgeniy Dodis, Serge Fehr, Carles Padró, and Daniel Wichs. Detection of algebraic manipulation with applications to robust secret sharing and fuzzy extractors. In Nigel P. Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 471–488, Istanbul, Turkey, April 13–17, 2008. Springer, Heidelberg, Germany. doi:10.1007/978-3-540-78967-3_27. 40

[CG13] Mahdi Cheraghchi and Venkatesan Guruswami. Capacity of non-malleable codes. *CoRR*, abs/1309.0458, 2013. URL: http://arxiv.org/abs/1309.0458, arXiv:1309.0458. 83

[CG14a] Mahdi Cheraghchi and Venkatesan Guruswami. Capacity of non-malleable codes. In Moni Naor, editor, *ITCS 2014: 5th Conference on Innovations in Theoretical Computer Science*, pages 155–168, Princeton, NJ, USA, January 12–14, 2014. Association for Computing Machinery. doi:10.1145/2554797.2554814. 15, 20, 22, 26, 41, 78, 82, 83

[CG14b] Mahdi Cheraghchi and Venkatesan Guruswami. Non-malleable coding against bit-wise and split-state tampering. In Yehuda Lindell, editor, *TCC 2014: 11th Theory of Cryptography Conference*, volume 8349 of *Lecture Notes in Computer Science*, pages 440–464, San Diego, CA, USA, February 24–26, 2014. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-54242-8_19. 23, 27, 28

[CGH+85] Benny Chor, Oded Goldreich, Johan Håstad, Joel Friedman, Steven Rudich, and Roman Smolensky. The bit extraction problem of t-resilient functions (preliminary version). In *26th Annual Symposium on Foundations of Computer Science*, pages 396–407, Portland, Oregon, October 21–23, 1985. IEEE Computer Society Press. doi:10.1109/SFCS.1985.55. 136

[CGL16] Eshan Chattopadhyay, Vipul Goyal, and Xin Li. Non-malleable extractors and codes, with their many tampered extensions. In Daniel Wichs and Yishay Mansour, editors, *48th Annual ACM Symposium on Theory of Computing*, pages 285–298, Cambridge, MA, USA, June 18–21, 2016. ACM Press. doi:10.1145/2897518.2897547. 23, 27, 30

[CGM+16] Nishanth Chandran, Vipul Goyal, Pratyay Mukherjee, Omkant Pandey, and Jalaj Upadhyay. Block-wise non-malleable codes. In Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, *ICALP 2016: 43rd International Colloquium on Automata, Languages and Programming*, volume 55 of *LIPIcs*, pages 31:1–31:14, Rome, Italy, July 11–15, 2016. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.ICALP.2016.31. 31, 76, 78, 82

[Chv79] Vasek Chvátal. The tail of the hypergeometric distribution. *Discrete Mathematics*, 25(3), 1979. 50

[CI93] Richard Cleve and Russell Impagliazzo. Martingales, collective coin flipping and discrete control processes. *In other words*, 1:5, 1993. 34, 39, 98, 99, 103, 136

[CKOS19] Eshan Chattopadhyay, Bhavana Kanukurthi, Sai Lakshmi Bhavana Obbattu, and Sruthi Sekar. Privacy amplification from non-malleable codes. In Feng Hao, Sushmita Ruj, and Sourav Sen Gupta, editors, *Progress in Cryptology - INDOCRYPT 2019: 20th International Conference in Cryptology in India*, volume 11898 of *Lecture Notes in Computer Science*, pages 318–337, Hyderabad, India, December 15–18, 2019. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-35423-7_16. 31

[CKR16] Nishanth Chandran, Bhavana Kanukurthi, and Srinivasan Raghuraman. Information-theoretic local non-malleable codes and their applications. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A: 13th Theory of Cryptography Conference, Part II*, volume 9563 of *Lecture Notes in Computer Science*, pages 367–392, Tel Aviv, Israel, January 10–13, 2016. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-49099-0_14. 30

[CL17] Eshan Chattopadhyay and Xin Li. Non-malleable codes and extractors for small-depth circuits, and affine functions. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *49th Annual ACM Symposium on Theory of Computing*, pages 1171–1184, Montreal, QC, Canada, June 19–23, 2017. ACM Press. doi:10.1145/3055399.3055483. 23, 25, 29, 158

[CL20] Eshan Chattopadhyay and Xin Li. Non-malleable codes, extractors and secret sharing for interleaved tampering and composition of tampering. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020: 18th Theory of Cryptography Conference, Part III*, volume 12552 of *Lecture Notes in Computer Science*, pages 584–613, Durham, NC, USA, November 16–19, 2020. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-64381-2_21. 27

[Cle86] Richard Cleve. Limits on the security of coin flips when half the processors are faulty (extended abstract). In *18th Annual ACM Symposium on Theory of Computing*, pages 364–369, Berkeley, CA, USA, May 28–30, 1986. ACM Press. doi:10.1145/12130.12168. 15, 16, 32, 33, 34, 98, 99

[Col71] James S Coleman. Control of collectivities and the power of a collectivity to act. *Social choice*, pages 269–300, 1971. 136

[COSV17a] Michele Ciampi, Rafail Ostrovsky, Luisa Siniscalchi, and Ivan Visconti. Delayed-input non-malleable zero knowledge and multi-party coin tossing in four rounds. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017: 15th Theory of Cryptography Conference, Part I*, volume 10677 of *Lecture Notes in Computer Science*, pages 711–742, Baltimore, MD, USA, November 12–15, 2017. Springer, Heidelberg, Germany. doi:10.1007/978-3-319-70500-2_24. 12

[COSV17b] Michele Ciampi, Rafail Ostrovsky, Luisa Siniscalchi, and Ivan Visconti. Round-optimal secure two-party computation from trapdoor permutations. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017: 15th Theory of Cryptography Conference, Part I*, volume 10677 of *Lecture Notes in Computer Science*, pages 678–710, Baltimore, MD, USA, November 12–15, 2017. Springer, Heidelberg, Germany. doi:10.1007/978-3-319-70500-2_23. 12

[CZ14] Eshan Chattopadhyay and David Zuckerman. Non-malleable codes against constant split-state tampering. In *55th Annual Symposium on Foundations of Computer Science*, pages 306–315, Philadelphia, PA, USA, October 18–21, 2014. IEEE Computer Society Press. doi:10.1109/FOCS.2014.40. 23, 27, 76, 79

[CZ16] Eshan Chattopadhyay and David Zuckerman. Explicit two-source extractors and resilient functions. In Daniel Wichs and Yishay Mansour, editors, *48th Annual ACM Symposium on Theory of Computing*, pages 670–683, Cambridge, MA, USA, June 18–21, 2016. ACM Press. doi:10.1145/2897518.2897528. 36

[DDN91] Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography (extended abstract). In *23rd Annual ACM Symposium on Theory of Computing*, pages 542–552, New Orleans, LA, USA, May 6–8, 1991. ACM Press. doi:10.1145/103418.103474. 31

[DK19] Dana Dachman-Soled and Mukul Kulkarni. Upper and lower bounds for continuous non-malleable codes. In Dongdai Lin and Kazue Sako, editors, *PKC 2019: 22nd International Conference on Theory and Practice of Public Key Cryptography, Part I*, volume 11442 of *Lecture Notes in Computer Science*, pages 519–548, Beijing, China, April 14–17, 2019. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-17253-4_18. 30

[DKO13] Stefan Dziembowski, Tomasz Kazana, and Maciej Obremski. Non-malleable codes from two-source extractors. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part II*, volume 8043 of *Lecture Notes in Computer Science*, pages 239–257, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-40084-1_14. 26, 76, 79

[DKS17] Dana Dachman-Soled, Mukul Kulkarni, and Aria Shahverdi. Tight upper and lower bounds for leakage-resilient, locally decodable and updatable non-malleable codes. In Serge Fehr, editor, *PKC 2017: 20th International Conference on Theory and Practice of Public Key Cryptography, Part I*, volume 10174 of *Lecture Notes in Computer Science*, pages 310–332, Amsterdam, The Netherlands, March 28–31, 2017. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-54365-8_13. 30

[DKS18] Dana Dachman-Soled, Mukul Kulkarni, and Aria Shahverdi. Local non-malleable codes in the bounded retrieval model. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018: 21st International Conference on Theory and Practice of Public Key Cryptography, Part II*, volume 10770 of *Lecture Notes in Computer Science*, pages 281–311, Rio de Janeiro, Brazil, March 25–29, 2018. Springer, Heidelberg, Germany. doi:10.1007/978-3-319-76581-5_10. 30

[DLMM11] Dana Dachman-Soled, Yehuda Lindell, Mohammad Mahmoody, and Tal Malkin. On the black-box complexity of optimally-fair coin tossing. In Yuval Ishai, editor, *TCC 2011: 8th Theory of Cryptography Conference*, volume 6597 of *Lecture Notes in Computer Science*, pages 450–467, Providence, RI, USA, March 28–30, 2011. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-19571-6_27. 34, 110, 112, 130, 136

[DLSZ15] Dana Dachman-Soled, Feng-Hao Liu, Elaine Shi, and Hong-Sheng Zhou. Locally decodable and updatable non-malleable codes and their applications. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015: 12th Theory of Cryptography Conference, Part I*, volume 9014 of *Lecture Notes in Computer Science*, pages 427–450, Warsaw, Poland, March 23–25, 2015. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-46494-6_18. 30

[DMM14] Dana Dachman-Soled, Mohammad Mahmoody, and Tal Malkin. Can optimally-fair coin tossing be based on one-way functions? In Yehuda Lindell, editor, *TCC 2014: 11th Theory of Cryptography Conference*, volume 8349 of *Lecture Notes in Computer Science*, pages 217–239, San Diego, CA, USA, February 24–26, 2014. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-54242-8_10. 34, 110, 112, 130, 136

[DMM18] Dimitrios I. Diochnos, Saeed Mahloujifar, and Mohammad Mahmoody. Adversarial risk and robustness: General definitions and implications for the uniform distribution. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*, pages 10380–10389, 2018. 136

[DORS08] Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM J. Comput.*, 38(1):97–139, 2008. 81

[DPW10] Stefan Dziembowski, Krzysztof Pietrzak, and Daniel Wichs. Non-malleable codes. In Andrew Chi-Chih Yao, editor, *ICS 2010: 1st Innovations in Computer Science*, pages 434–452, Tsinghua University, Beijing, China, January 5–7, 2010. Tsinghua University Press. 12, 13, 19, 22, 40

[DW09] Yevgeniy Dodis and Daniel Wichs. Non-malleable extractors and symmetric key cryptography from weak secrets. In Michael Mitzenmacher, editor, *41st Annual ACM Symposium on Theory of Computing*, pages 601–610, Bethesda, MD, USA, May 31 – June 2, 2009. ACM Press. doi:10.1145/1536414.1536496. 23, 31

[EGL82] Shimon Even, Oded Goldreich, and Abraham Lempel. A randomized protocol for signing contracts. In David Chaum, Ronald L. Rivest, and Alan T. Sherman, editors, *Advances in Cryptology – CRYPTO'82*, pages 205–210, Santa Barbara, CA, USA, 1982. Plenum Press, New York, USA. 98

[EMM20] Omid Etesami, Saeed Mahloujifar, and Mohammad Mahmoody. Computational concentration of measure: Optimal bounds, reductions, and more. In Shuchi Chawla, editor, *31st Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 345–363, Salt Lake City, UT, USA, January 5–8, 2020. ACM-SIAM. doi:10.1137/1.9781611975994.21. 136, 142

[Fei99] Uriel Feige. Noncryptographic selection protocols. In *40th Annual Symposium on Foundations of Computer Science*, pages 142–153, New York, NY, USA, October 17–19, 1999. IEEE Computer Society Press. doi:10.1109/SFFCS.1999.814586. 37

[FGJ+19] Nils Fleischhacker, Vipul Goyal, Abhishek Jain, Anat Paskin-Cherniavsky, and Slava Radune. Interactive non-malleable codes. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019: 17th Theory of Cryptography Conference, Part II*, volume 11892 of *Lecture Notes in Computer Science*, pages 233–263, Nuremberg, Germany, December 1–5, 2019. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-36033-7_9. 30

[FHH+19] Yuval Filmus, Lianna Hambardzumyan, Hamed Hatami, Pooya Hatami, and David Zuckerman. Biasing Boolean functions and collective coin-flipping protocols over arbitrary product distributions. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *ICALP 2019: 46th International Colloquium on Automata, Languages and Programming*, volume 132 of *LIPIcs*, pages 58:1–58:13, Patras, Greece, July 9–12, 2019. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.ICALP.2019.58. 36, 139

[FHMV17] Sebastian Faust, Kristina Hostáková, Pratyay Mukherjee, and Daniele Venturi. Non-malleable codes for space-bounded tampering. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part II*, volume 10402 of *Lecture Notes in Computer Science*, pages 95–126, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Heidelberg, Germany. doi:10.1007/978-3-319-63715-0_4. 30

[FMNV14] Sebastian Faust, Pratyay Mukherjee, Jesper Buus Nielsen, and Daniele Venturi. Continuous non-malleable codes. In Yehuda Lindell, editor, *TCC 2014: 11th Theory of Cryptography Conference*, volume 8349 of *Lecture Notes in Computer Science*, pages 465–488, San Diego, CA, USA, February 24–26, 2014. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-54242-8_20. 30

[FMVW14] Sebastian Faust, Pratyay Mukherjee, Daniele Venturi, and Daniel Wichs. Efficient non-malleable codes and key-derivation for poly-size tampering circuits. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 111–128, Copenhagen, Denmark, May 11–15, 2014. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-55220-5_7. 22, 41

[Fri92] Joel Friedman. On the bit extraction problem. In *33rd Annual Symposium on Foundations of Computer Science*, pages 314–319, Pittsburgh, PA, USA, October 24–27, 1992. IEEE Computer Society Press. doi:10.1109/SFCS.1992.267760. 136

[Fri04] Ehud Friedgut. Influences in product spaces: Kkl and bkkkl revisited. *Combinatorics Probability and Computing*, 13(1):17–29, 2004. 36

[FY92] Matthew K. Franklin and Moti Yung. Communication complexity of secure computation (extended abstract). In *24th Annual ACM Symposium on Theory of Computing*, pages 699–710, Victoria, BC, Canada, May 4–6, 1992. ACM Press. doi:10.1145/129712.129780. 42, 192

[GH15] Ran Gelles and Bernhard Haeupler. Capacity of interactive communication over erasure channels and channels with feedback. In Piotr Indyk, editor, *26th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1296–1311, San Diego, CA, USA, January 4–6, 2015. ACM-SIAM. doi:10.1137/1.9781611973730.86. 11

[GHKL08] S. Dov Gordon, Carmit Hazay, Jonathan Katz, and Yehuda Lindell. Complete fairness in secure two-party computation. In Richard E. Ladner and Cynthia Dwork, editors, *40th Annual ACM Symposium on Theory of Computing*, pages 413–422, Victoria, BC, Canada, May 17–20, 2008. ACM Press. doi:10.1145/1374376.1374436. 33, 98

[GK10] S. Dov Gordon and Jonathan Katz. Partial fairness in secure two-party computation. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 157–176, French Riviera, May 30 – June 3, 2010. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-13190-5_8. 98

[GK18a] Vipul Goyal and Ashutosh Kumar. Non-malleable secret sharing. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *50th Annual ACM Symposium on Theory of Computing*, pages 685–698, Los Angeles, CA, USA, June 25–29, 2018. ACM Press. doi:10.1145/3188745.3188872. 31, 40

[GK18b] Vipul Goyal and Ashutosh Kumar. Non-malleable secret sharing for general access structures. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part I*, volume 10991 of *Lecture Notes in Computer Science*, pages 501–530, Santa Barbara, CA, USA, August 19–23, 2018. Springer, Heidelberg, Germany. doi:10.1007/978-3-319-96884-1_17. 31, 40

[GKM+00] Yael Gertner, Sampath Kannan, Tal Malkin, Omer Reingold, and Mahesh Viswanathan. The relationship between public key encryption and oblivious transfer. In *41st Annual Symposium on Foundations of Computer Science*, pages 325–335, Redondo Beach, CA, USA, November 12–14, 2000. IEEE Computer Society Press. doi:10.1109/SFCS.2000.892121. 129

[GKP15] Shafi Goldwasser, Yael Tauman Kalai, and Sunoo Park. Adaptively secure coin-flipping, revisited. In Magnús M. Halldórsson, Kazuo Iwama, Naoki Kobayashi, and Bettina Speckmann, editors, *ICALP 2015: 42nd International Colloquium on Automata, Languages and Programming, Part II*, volume 9135 of *Lecture Notes in Computer Science*, pages 663–674, Kyoto, Japan, July 6–10, 2015. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-47666-6_53. 38, 39, 138, 139

[GMMM18] Sanjam Garg, Mohammad Mahmoody, Daniel Masny, and Izaak Meckler. On the round complexity of OT extension. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part III*, volume 10993 of *Lecture Notes in Computer Science*, pages 545–574, Santa Barbara, CA, USA, August 19–23, 2018. Springer, Heidelberg, Germany. doi:10.1007/978-3-319-96878-0_19. 11

[GMPP16] Sanjam Garg, Pratyay Mukherjee, Omkant Pandey, and Antigoni Polychroniadou. The exact round complexity of secure computation. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 448–476, Vienna, Austria, May 8–12, 2016. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-49896-5_16. 12

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th Annual ACM Symposium on Theory of Computing*, pages 218–229, New York City, NY, USA, May 25–27, 1987. ACM Press. doi:10.1145/28395.28420. 98

[GMW18] Divya Gupta, Hemanta K. Maji, and Mingyuan Wang. Non-malleable codes against lookahead tampering. In Debrup Chakraborty and Tetsu Iwata, editors, *Progress in Cryptology - INDOCRYPT 2018: 19th International Conference in Cryptology in India*, volume 11356 of *Lecture Notes in Computer Science*, pages 307–328, New Delhi, India, December 9–12, 2018. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-05378-9_17. 18, 27

[GMW19] Divya Gupta, Hemanta K. Maji, and Mingyuan Wang. Explicit rate-1 non-malleable codes for local tampering. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019, Part I*, volume 11692 of *Lecture Notes in Computer Science*, pages 435–466, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-26948-7_16. 18, 29

[Goy11] Vipul Goyal. Constant round non-malleable protocols using one way functions. In Lance Fortnow and Salil P. Vadhan, editors, *43rd Annual ACM Symposium on Theory of Computing*, pages 695–704, San Jose, CA, USA, June 6–8, 2011. ACM Press. doi:10.1145/1993636.1993729. 12

[GPR16] Vipul Goyal, Omkant Pandey, and Silas Richelson. Textbook non-malleable commitments. In Daniel Wichs and Yishay Mansour, editors, *48th Annual ACM Symposium on Theory of Computing*, pages 1128–1141, Cambridge, MA, USA, June 18–21, 2016. ACM Press. doi:10.1145/2897518.2897657. 31

[GS18] Sanjam Garg and Akshayaram Srinivasan. Two-round multiparty secure computation from minimal assumptions. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part II*, volume 10821 of *Lecture Notes in Computer Science*, pages 468–499, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Heidelberg, Germany. doi:10.1007/978-3-319-78375-8_16. 12

[GUV07] Venkatesan Guruswami, Christopher Umans, and Salil P. Vadhan. Unbalanced expanders and randomness extractors from parvaresh-vardy codes. In *22nd Annual IEEE Conference on Computational Complexity (CCC 2007), 13-16 June 2007, San Diego, California, USA*, pages 96–108, 2007. URL: https://doi.org/10.1109/CCC.2007.38. 81, 82

[Hae14] Bernhard Haeupler. Interactive channel capacity revisited. In *55th Annual Symposium on Foundations of Computer Science*, pages 226–235, Philadelphia, PA, USA, October 18–21, 2014. IEEE Computer Society Press. doi:10.1109/FOCS.2014.32. 11

[Har66] Lawrence H Harper. Optimal numberings and isoperimetric problems on graphs. *Journal of Combinatorial Theory*, 1(3):385–393, 1966. 38, 136, 137, 139

[Har99] LH Harper. On an isoperimetric problem for hamming graphs. *Discrete applied mathematics*, 95(1-3):285–309, 1999. 38, 139

[HK20] Iftach Haitner and Yonatan Karidi-Heller. A tight lower bound on adaptively secure full-information coin flip. In *FOCS*, 2020. 138, 139

[HKH20] Iftach Haitner and Yonatan Karidi-Heller. A tight lower bound on adaptively secure full-information coin flip. In *61st Annual Symposium on Foundations of Computer Science*, pages 1268–1276, Durham, NC, USA, November 16–19, 2020. IEEE Computer Society Press. doi:10.1109/FOCS46700.2020.00120. 39, 159

[HMO18] Iftach Haitner, Nikolaos Makriyannis, and Eran Omri. On the complexity of fair coin flipping. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018: 16th Theory of Cryptography Conference, Part I*, volume 11239 of *Lecture Notes in Computer Science*, pages 539–562, Panaji, India, November 11–14, 2018. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-03807-6_20. 34

[HO11] Iftach Haitner and Eran Omri. Coin flipping with constant bias implies one-way functions. In Rafail Ostrovsky, editor, *52nd Annual Symposium on Foundations of Computer Science*, pages 110–119, Palm Springs, CA, USA, October 22–25, 2011. IEEE Computer Society Press. doi:10.1109/FOCS.2011.29. 99

[Hoe63] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American statistical association*, 58(301), 1963. 50

[HOZ16] Iftach Haitner, Eran Omri, and Hila Zarosim. Limits on the usefulness of random oracles. *Journal of Cryptology*, 29(2):283–335, April 2016. doi:10.1007/s00145-014-9194-9. 136

[HR07] Iftach Haitner and Omer Reingold. Statistically-hiding commitment from any one-way function. In David S. Johnson and Uriel Feige, editors, *39th Annual ACM Symposium on Theory of Computing*, pages 1–10, San Diego, CA, USA, June 11–13, 2007. ACM Press. doi:10.1145/1250790.1250792. 34

[HT14] Iftach Haitner and Eliad Tsfadia. An almost-optimally fair three-party coin-flipping protocol. In David B. Shmoys, editor, *46th Annual ACM Symposium on Theory of Computing*, pages 408–416, New York, NY, USA, May 31 – June 3, 2014. ACM Press. doi:10.1145/2591796.2591842. 33, 98

[IKNP03] Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending oblivious transfers efficiently. In Dan Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 145–161, Santa Barbara, CA, USA, August 17–21, 2003. Springer, Heidelberg, Germany. doi:10.1007/978-3-540-45146-4_9. 11

[Imp95] Russell Impagliazzo. A personal view of average-case complexity. In *Proceedings of Structure in Complexity Theory. Tenth Annual IEEE Conference*, pages 134–147. IEEE, 1995. 99

[IR89] Russell Impagliazzo and Steven Rudich. Limits on the provable consequences of one-way permutations. In *21st Annual ACM Symposium on Theory of Computing*, pages 44–61, Seattle, WA, USA, May 15–17, 1989. ACM Press. doi:10.1145/73007.73012. 16, 108

[Jer85] Mark Jerrum. Random generation of combinatorial structures from a uniform distribution (extended abstract). In Wilfried Brauer, editor, *Automata, Languages and Programming, 12th Colloquium, Nafplion, Greece, July 15-19, 1985, Proceedings*, volume 194 of *Lecture Notes in Computer Science*, pages 290–299. Springer, 1985. doi:10.1007/BFb0015754. 138

[JVV86] Mark Jerrum, Leslie G. Valiant, and Vijay V. Vazirani. Random generation of combinatorial structures from a uniform distribution. *Theor. Comput. Sci.*, 43:169–188, 1986. doi:10.1016/0304-3975(86)90174-X. 138

[JW15] Zahra Jafargholi and Daniel Wichs. Tamper detection and continuous non-malleable codes. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015: 12th Theory of Cryptography Conference, Part I*, volume 9014 of *Lecture Notes in Computer Science*, pages 451–480, Warsaw, Poland, March 23–25, 2015. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-46494-6_19. 22, 30

[Kat68] G Katona. A theorem for finite sets, theory of graphs (p. erdös and g. katona, eds.), 1968. 136, 137, 139

[Kil00] Joe Kilian. More general completeness theorems for secure two-party computation. In *32nd Annual ACM Symposium on Theory of Computing*, pages 316–324, Portland, OR, USA, May 21–23, 2000. ACM Press. doi:10.1145/335305.335342. 17, 107, 120, 121

[KKL88] Jeff Kahn, Gil Kalai, and Nathan Linial. The influence of variables on Boolean functions (extended abstract). In *29th Annual Symposium on Foundations of Computer Science*, pages 68–80, White Plains, NY, USA, October 24–26, 1988. IEEE Computer Society Press. doi:10.1109/SFCS.1988.21923. 36

[KKR18] Yael Tauman Kalai, Ilan Komargodski, and Ran Raz. A lower bound for adaptively-secure collective coin-flipping protocols. In Ulrich Schmid and Josef Widder, editors, *32nd International Symposium on Distributed Computing, DISC 2018, New Orleans, LA, USA, October 15-19, 2018*, volume 121 of *LIPIcs*, pages 34:1–34:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. 39, 138, 139

[KMM19] Hamidreza Amini Khorasgani, Hemanta K. Maji, and Tamalika Mukherjee. Estimating gaps in martingales and applications to coin-tossing: Constructions and hardness. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019: 17th Theory of Cryptography Conference, Part II*, volume 11892 of *Lecture Notes in Computer Science*, pages 333–355, Nuremberg, Germany, December 1–5, 2019. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-36033-7_13. 39, 103, 136, 139, 144

[KMMW21] Hamidreza Amini Khorasgani, Hemanta K. Maji, Himanshi Mehta, and Mingyuan Wang. Efficient distributed coin-tossing protocols. In *IEEE International Symposium on Information Theory ISIT*, 2021. 39

[KMW20] Hamidreza Amini Khorasgani, Hemanta K. Maji, and Mingyuan Wang. Coin tossing with lazy defense: Hardness of computation results. *IACR Cryptol. ePrint Arch.*, 2020:131, 2020. URL: https://eprint.iacr.org/2020/131. 101, 136, 144

[KMW21] Hamidreza Amini Khorasgani, Hemanta K. Maji, and Mingyuan Wang. Optimally-secure coin-tossing against a byzantine adversary. In *IEEE International Symposium on Information Theory ISIT 2021*, 2021. 18

[KO04] Jonathan Katz and Rafail Ostrovsky. Round-optimal secure two-party computation. In Matthew Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 335–354, Santa Barbara, CA, USA, August 15–19, 2004. Springer, Heidelberg, Germany. doi:10.1007/978-3-540-28628-8_21. 12

[KOS03] Jonathan Katz, Rafail Ostrovsky, and Adam Smith. Round efficiency of multiparty computation with a dishonest majority. In Eli Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 578–595, Warsaw, Poland, May 4–8, 2003. Springer, Heidelberg, Germany. doi:10.1007/3-540-39200-9_36. 12

[KOS17] Bhavana Kanukurthi, Sai Lakshmi Bhavana Obbattu, and Sruthi Sekar. Four-state non-malleable codes with explicit constant rate. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017: 15th Theory of Cryptography Conference, Part II*, volume 10678 of *Lecture Notes in Computer Science*, pages 344–375, Baltimore, MD, USA, November 12–15, 2017. Springer, Heidelberg, Germany. doi:10.1007/978-3-319-70503-3_11. 27, 76, 77, 79

[KOS18] Bhavana Kanukurthi, Sai Lakshmi Bhavana Obbattu, and Sruthi Sekar. Non-malleable randomness encoders and their applications. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part III*, volume 10822 of *Lecture Notes in Computer Science*, pages 589–617, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Heidelberg, Germany. doi:10.1007/978-3-319-78372-7_19. 27, 40, 77, 79

[KR13] Gillat Kol and Ran Raz. Interactive channel capacity. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th Annual ACM Symposium on Theory of Computing*, pages 715–724, Palo Alto, CA, USA, June 1–4, 2013. ACM Press. doi:10.1145/2488608.2488699. 11

[Kru63] Joseph B Kruskal. The number of simplices in a complex. *Mathematical optimization techniques*, 10:251–278, 1963. 136, 137, 139

[Li17] Xin Li. Improved non-malleable extractors, non-malleable codes and independent source extractors. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *49th Annual ACM Symposium on Theory of Computing*, pages 1144–1156, Montreal, QC, Canada, June 19–23, 2017. ACM Press. doi:10.1145/3055399.3055486. 23, 27, 52, 76, 79

[Li18] Xin Li. Pseudorandom correlation breakers, independence preserving mergers and their applications. *ECCC*, 25, 2018. 52, 77

[Li19] Xin Li. Non-malleable extractors and non-malleable codes: Partially optimal constructions. In Amir Shpilka, editor, *34th Computational Complexity Conference, CCC 2019, July 18-20, 2019, New Brunswick, NJ, USA*, volume 137 of *LIPIcs*, pages 28:1–28:49. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.CCC.2019.28. 23, 27

[Lin17] Yehuda Lindell. How to simulate it - A tutorial on the simulation proof technique. In *Tutorials on the Foundations of Cryptography*. 2017. 119

[LLS89] David Lichtenstein, Nathan Linial, and Michael Saks. Some extremal problems arising from discrete control processes. *Combinatorica*, 9(3):269–287, 1989. 39, 137, 138

[LOZ12] Yehuda Lindell, Eran Omri, and Hila Zarosim. Completeness for symmetric two-party functionalities - revisited. In Xiaoyun Wang and Kazue Sako, editors, *Advances in Cryptology – ASIACRYPT 2012*, volume 7658 of *Lecture Notes in Computer Science*, pages 116–133, Beijing, China, December 2–6, 2012. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-34961-4_9. 129

[Mak14] Nikolaos Makriyannis. On the classification of finite Boolean functions up to fairness. In Michel Abdalla and Roberto De Prisco, editors, *SCN 14: 9th International Conference on Security in Communication Networks*, volume 8642 of *Lecture Notes in Computer Science*, pages 135–154, Amalfi, Italy, September 3–5, 2014. Springer, Heidelberg, Germany. doi:10.1007/978-3-319-10879-7_9. 33, 98

[MDM19] Saeed Mahloujifar, Dimitrios I. Diochnos, and Mohammad Mahmoody. The curse of concentration in robust learning: Evasion and poisoning attacks from concentration of measure. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 4536–4543. AAAI Press, 2019. doi:10.1609/aaai.v33i01.33014536. 136

[Mek17] Raghu Meka. Explicit resilient functions matching ajtai-linial. In Philip N. Klein, editor, *28th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1132–1148, Barcelona, Spain, January 16–19, 2017. ACM-SIAM. doi:10.1137/1.9781611974782.73. 36

[MM19] Saeed Mahloujifar and Mohammad Mahmoody. Can adversarially robust learning leveragecomputational hardness? In Aurélien Garivier and Satyen Kale, editors, *Algorithmic Learning Theory, ALT 2019, 22-24 March 2019, Chicago, Illinois, USA*, volume 98 of *Proceedings of Machine Learning Research*, pages 581–609. PMLR, 2019. URL: http://proceedings.mlr.press/v98/mahloujifar19a.html. 136, 142

[MMP14] Mohammad Mahmoody, Hemanta K. Maji, and Manoj Prabhakaran. On the power of public-key encryption in secure computation. In Yehuda Lindell, editor, *TCC 2014: 11th Theory of Cryptography Conference*, volume 8349 of *Lecture Notes in Computer Science*, pages 240–264, San Diego, CA, USA, February 24–26, 2014. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-54242-8_11. 99, 107, 108, 129, 131, 132, 133

[MNS09] Tal Moran, Moni Naor, and Gil Segev. An optimally fair coin toss. In Omer Reingold, editor, *TCC 2009: 6th Theory of Cryptography Conference*, volume 5444 of *Lecture Notes in Computer Science*, pages 1–18. Springer, Heidelberg, Germany, March 15–17, 2009. doi:10.1007/978-3-642-00457-5_1. 16, 17, 33, 98, 99, 120, 121, 129

[MPR09] Hemanta K. Maji, Manoj Prabhakaran, and Mike Rosulek. Complexity of multi-party computation problems: The case of 2-party symmetric secure function evaluation. In Omer Reingold, editor, *TCC 2009: 6th Theory of Cryptography Conference*, volume 5444 of *Lecture Notes in Computer Science*, pages 256–273. Springer, Heidelberg, Germany, March 15–17, 2009. doi:10.1007/978-3-642-00457-5_16. 121

[MW20] Hemanta K. Maji and Mingyuan Wang. Black-box use of one-way functions is useless for optimal fair coin-tossing. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology – CRYPTO 2020, Part II*, volume 12171 of *Lecture Notes in Computer Science*, pages 593–617, Santa Barbara, CA, USA, August 17–21, 2020. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-56880-1_21. 6, 18, 34, 130, 131, 133, 134, 136, 144

[MW21] Hemanta K. Maji and Mingyuan Wang. Computational hardness of optimal fair computation: Beyond minicrypt. In *CRYPTO 2021*, 2021. 18, 34

[Nao91] Moni Naor. Bit commitment using pseudorandomness. *J. Cryptology*, 4(2):151–158, 1991. doi:10.1007/BF00196774. 34

[Nis90] Noam Nisan. Psuedorandom generators for space-bounded computation. In *22nd Annual ACM Symposium on Theory of Computing*, pages 204–212, Baltimore, MD, USA, May 14–16, 1990. ACM Press. doi:10.1145/100216.100242. 42, 47, 53

[NOVY98] Moni Naor, Rafail Ostrovsky, Ramarathnam Venkatesan, and Moti Yung. Perfect zero-knowledge arguments for *NP* using any one-way permutation. *J. Cryptology*, 11(2):87–108, 1998. doi:10.1007/s001459900037. 34

[OPVV18] Rafail Ostrovsky, Giuseppe Persiano, Daniele Venturi, and Ivan Visconti. Continuously non-malleable codes in the split-state model from minimal assumptions. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part III*, volume 10993 of *Lecture Notes in Computer Science*, pages 608–639, Santa Barbara, CA, USA, August 19–23, 2018. Springer, Heidelberg, Germany. doi:10.1007/978-3-319-96878-0_21. 30

[OS08] Ryan O'Donnell and Rocco A. Servedio. The Chow parameters problem. In Richard E. Ladner and Cynthia Dwork, editors, *40th Annual ACM Symposium on Theory of Computing*, pages 517–526, Victoria, BC, Canada, May 17–20, 2008. ACM Press. doi:10.1145/1374376.1374450. 136

[OS11] Ryan O'Donnell and Rocco A. Servedio. The chow parameters problem. *SIAM J. Comput.*, 40(1):165–199, 2011. doi:10.1137/090756466. 136

[Pap83] Christos H. Papadimitriou. Games against nature (extended abstract). In *24th Annual Symposium on Foundations of Computer Science*, pages 446–450, Tucson, Arizona, November 7–9, 1983. IEEE Computer Society Press. doi:10.1109/SFCS.1983.20. 98

[Pas04] Rafael Pass. Bounded-concurrent secure multi-party computation with a dishonest majority. In László Babai, editor, *36th Annual ACM Symposium on Theory of Computing*, pages 232–241, Chicago, IL, USA, June 13–16, 2004. ACM Press. doi:10.1145/1007352.1007393. 12

[PW10] Rafael Pass and Hoeteck Wee. Constant-round non-malleable commitments from sub-exponential one-way functions. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 638–655, French Riviera, May 30 – June 3, 2010. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-13190-5_32. 12

[Rab81] Michael O. Rabin. How to exchange secrets by oblivious transfer. *Technical Memo TR-81*, 1981. 98

[Rab05] Michael O. Rabin. How to exchange secrets with oblivious transfer. Cryptology ePrint Archive, Report 2005/187, 2005. https://eprint.iacr.org/2005/187. 98

[RS20] Peter Michael Reichstein Rasmussen and Amit Sahai. Expander graphs are non-malleable codes. In Yael Tauman Kalai, Adam D. Smith, and Daniel Wichs, editors, *ITC 2020: 1st Conference on Information-Theoretic Cryptography*, pages 6:1–6:10, Boston, MA, USA, June 17–19, 2020. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.ITC.2020.6. 28

[RSZ99] Alexander Russell, Michael E. Saks, and David Zuckerman. Lower bounds for leader election and collective coin-flipping in the perfect information model. In *31st Annual ACM Symposium on Theory of Computing*, pages 339–347, Atlanta, GA, USA, May 1–4, 1999. ACM Press. doi:10.1145/301250.301337. 37, 38

[RZ98] Alexander Russell and David Zuckerman. Perfect information leader election in log*n + O(1) rounds. In *39th Annual Symposium on Foundations of Computer Science*, pages 576–583, Palo Alto, CA, USA, November 8–11, 1998. IEEE Computer Society Press. doi:10.1109/SFCS.1998.743508. 37

[Sak89] Michael E. Saks. A robust noncryptographic protocol for collective coin flipping. *SIAM J. Discrete Math.*, 2(2):240–244, 1989. 37

[Sch96] Leonard J Schulman. Coding for interactive communication. *IEEE transactions on information theory*, 42(6):1745–1756, 1996. 11, 30

[Sch17] René L Schilling. *Measures, integrals and martingales*. Cambridge University Press, 2017. 100

[Sha79] Adi Shamir. How to share a secret. *Communications of the Association for Computing Machinery*, 22(11):612–613, November 1979. 42, 192

[SV84] Miklos Santha and Umesh V. Vazirani. Generating quasi-random sequences from slightly-random sources (extended abstract). In *25th Annual Symposium on Foundations of Computer Science*, pages 434–440, Singer Island, Florida, October 24–26, 1984. IEEE Computer Society Press. doi:10.1109/SFCS.1984.715945. 136

[Vad12] S.P. Vadhan. *Pseudorandomness*. Foundations and trends in theoretical computer science. Now Publishers, 2012. URL: https://books.google.com/books?id=iam4lAEACAAJ. 81, 82

[Vaz85] Umesh V. Vazirani. Towards a strong communication complexity theory or generating quasi-random sequences from two communicating slightly-random sources (extended abstract). In *17th Annual ACM Symposium on Theory of Computing*, pages 366–378, Providence, RI, USA, May 6–8, 1985. ACM Press. doi:10.1145/22145.22186. 136

[Vio11] Emanuele Viola. Extractors for circuit sources. In Rafail Ostrovsky, editor, *52nd Annual Symposium on Foundations of Computer Science*, pages 220–229, Palm Springs, CA, USA, October 22–25, 2011. IEEE Computer Society Press. doi:10.1109/FOCS.2011.20. 46, 48

[Win71] Robert O. Winder. Chow parameters in threshold logic. *J. ACM*, 18(2):265–289, 1971. doi:10.1145/321637.321647. 136

[Yao82] Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science*, pages 160–164, Chicago, Illinois, November 3–5, 1982. IEEE Computer Society Press. doi:10.1109/SFCS.1982.38. 98

# A. APPENDIX FOR CHAPTER 3

## A.1 Instantiation of Error-Correcting Secret-Sharing Schemes

In this section, we will use Shamir's secret-sharing scheme [Sha79], with a standard share-packing technique [BM84, FY92], to give a construction for Lemma 3.4.2. The error-correcting secret-sharing scheme can be formally defined as follows.

Let $\mathbb{F}$ to be a field of characteristic 2 such that $\log|\mathbb{F}| = \varphi$. We view message $m$ as a concatenation of binary representations of elements from $\mathbb{F}$, i.e., $m = (\alpha_1, \alpha_2, \ldots, \alpha_\ell)$, where $\alpha_1, \ldots, \alpha_\ell \in \mathbb{F}$. For any $\zeta > 0$, pick $n$ such that $n - \ell = n^{1-\zeta/2}$. Let $\beta_1, \beta_2, \ldots, \beta_\ell$ and $\gamma_1, \gamma_2, \ldots, \gamma_n$ be arbitrary elements from field $\mathbb{F}$. Pick a polynomial $p(x) \in \mathbb{F}(x)$ of degree $\ell + (n-\ell)/2$ such that $p(\beta_i) = \alpha_i$ for $i \in [\ell]$. The encoding of $m$ is defined as $\mathsf{Enc}(m) := (p(\gamma_1), p(\gamma_2), \ldots, p(\gamma_n))$. By standard argument, this is a $(n', \ell', d, t)$ error-correcting secret-sharing scheme with $n' = n\varphi$ and $\ell' = \ell\varphi$. And when $\varphi = \Theta(\log n)$, we have $d, t = \widetilde{\Theta}(n^{1-\zeta/2}) \geqslant (n')^{1-\zeta}$. This construction satisfies Lemma 3.4.2.

## A.2 Proof of Lemma 3.6.1

*Proof.* We provide the proof for (1) and $c^L$ case of (2). Proofs for $c^R$ case of (2) are analogous to the $c^L$ case. And (3) is straightforward from (2). We first show that if $G(s^L)$ and $G(s^R)$ are truly uniformly random strings, then the lemma is correct. Then, we show that if this lemma is incorrect with pseudorandom bits, there is a FSM $Q$ that breaks the underlying PRG $G$.

(1) Firstly, since the output locality is bounded by $\delta$, the number of input neighbors of $\widetilde{\alpha^R}$ is bounded by $\delta \cdot |\widetilde{\alpha^R}| = \delta n^\Lambda$. So the number of bits from $\alpha^L$ who have input locality higher than $4\delta/\mu$ onto $\widetilde{\alpha^R}$ is at most $\mu n^\Lambda/4$. So, the expected number of those high input locality bits that are from $c^L$ is at most $(\mu n^\Lambda/4) \cdot 2n^{-(\Lambda-\beta_1)} = \mu n^{\beta_1}/2$ (using bias of $\rho^L$). By Chernoff bound, with probability at least $1 - \exp(-\Theta(\mu n^{\beta_1}))$, at most $\mu n^{\beta_1}$ many bits from $c^L$ will have input locality higher than $4\delta/\mu$ onto $\widetilde{\alpha^R}$.

(2) Secondly, because of output locality bound $\delta$, the number of input neighbors of $\widetilde{c}$ is bounded by $\delta n$ and the number of bits from $\alpha^L$ with input locality higher than $n^{1-\gamma-\tau}$ is at most $\delta n^{\gamma+\tau}$. Therefore, the expected number of those high input locality bits that are

from $c^L$ is at most $\delta n^{\gamma+\tau} \cdot 2n^{-(\Lambda-\beta_1)} = 2\delta n^{\beta_1-\tau}$. By Chernoff bound, with probability at least $1 - \exp(-\Theta(\delta n^{\beta_1-\tau}))$[1], at most $4\delta n^{\beta_1-\tau}$ bits from $c^L$ will have input locality higher than $n^{1-\gamma-\tau}$ onto $\widetilde{c}$.

Now we show that these are correct even when we use PRG $G$ that fools FSMs with space $\kappa \log^2 n$. Suppose some tampering function $f$ succeeds with probability more than $2^{-\Omega(\log^2 n)}$ in violating condition (1) or (2l). Then we will use $f$ to construct FSM $Q$ that breaks pseudorandomness of $G$. First, we note that the function $f$ structurally determines the positions in $\alpha^L$ that have higher than $4\delta/\mu$ input locality onto $\widetilde{\alpha^R}$ and that have higher than $n^{1-\gamma-\tau}$ input locality onto $\widetilde{c}$. Now, we hardcode these locations with high input locality, say $\mathcal{I}$ and $\mathcal{J}$, in FSM $Q$. It takes $y_1, y_2, \ldots, y_{n^\Lambda}$ as input. Now a state in $Q$ stores $(\ell, A, B)$ where $\ell$ denotes the number of symbols read so far, $A$ denotes the number of indices $i$ so far such that $\rho^L(y_i) = 1$ and $i \in \mathcal{I}$. (Note that $A$ denotes the number of indices in $c^L$ that have high input locality onto $\widetilde{\alpha^R}$.) Similarly, $B$ denotes the number of indices $j$ so far such that $\rho^L(y_j) = 1$ and $j \in \mathcal{J}$. (Note that $B$ denotes the number of indices in $c^L$ that have high input locality onto $\widetilde{c}$.) The final output of $Q$ will be 1 when $A \leqslant \mu n^{\beta_1}$ and $B \leqslant 4\delta n^{\beta_1-\tau}$.

Clearly, by our argument above, on a true uniform string, $Q$ will output 1 with probability $1 - \exp(-\Omega(n^{\beta_1-\tau}))$. If this lemma is incorrect, on a pseudorandom string, $Q$ will output 1 with probability less than $1 - 2^{-\Omega(\log^2 n)}$ and hence $Q$ will break the underlying PRG with success probability higher than $2^{-\Omega(\log^2 n)}$. Finally, note that $Q$ only needs $O(\log n)$ space to record $(j, A, B)$. This completes the proof. □

### A.3 Construction with modified parameters

In this section, we show that we can slightly tweak our compiler to get the following theorem.

**Theorem A.3.1.** *Let $\{0,1\}^\ell$ be the message space and $\delta = o(\log n)$. There exists an explicit and efficient rate-1 NMC against* $\mathrm{Local}^\delta$ *with simulation error that is negligible in $n$ and uses the following primitives in a black-box manner.*

1. *For appropriate $\zeta > 0$, an $(n, \ell, d, t)$-ECSS scheme with $d, t \geqslant n^{1-\zeta}$ and $n = (1+o(1))\ell$.*

[1]↑Since we have the freedom to pick $\tau$ small enough, we can make sure $\tau < \beta_1$.

2. *For some constant* $\lambda, \mu$ *and* $\eta = n^{\Theta(1)}$*, a* $(\lambda, \mu, \ell_i, \ell_o)$*-NMC against leaky input and output local tampering for messages in* $\{0,1\}^\eta$*, rate* $1/\mathsf{poly}(\eta)$*,* $\ell_o = O(\log \eta)$*,* $\ell_i = O(\log \eta)$*, simulation error negligible in* $\eta$*.*
3. *For some constant* $\Lambda > 0$*, a PRG* $G : (\{0,1\}^{\log^2 n})^{3\Lambda \log n} \longrightarrow (\{0,1\}^{\log^2 n})^{n^\Lambda}$ *that is secure against FSM with alphabet size* $\log^2 n$ *and space* $\Theta(\log^2 n)$ *with error that is negligible in* $n$*.*

**Comparison with Theorem 3.5.1.** This theorem only requires the base NMC to have $1/\mathsf{poly}(\eta)$ rate, instead of $1/\eta^{o(1)}$ as in Theorem 3.5.1. It gives this relaxation at the cost of achieving a slightly worse locality guarantee. Note that this compiler requires significantly lower rate for the base NMC. However, it yields an NMC against $o(\log n)$-local tampering functions, which already yields an NMC against $\mathsf{NC}^0$ tampering functions.

**Construction and parameter setting.** The construction for this theorem is the same as Theorem 3.5.1 (cf. Figure 3.2). The only difference is that because the rate of our base NMC is very poor, we no longer enjoy the freedom to pick as many as $n^{1-\varepsilon_1}$ error bits for any $\varepsilon_1 > 0$. Otherwise, it is possible that $\mathrm{NMEnc}_0(E, e)$ will be of length $\Theta(n)$ or even $\omega(n)$ and the construction is not rate-1. So depending on the exact rate of the base NMC, we will pick $1 - \varepsilon_1$ small enough such that the length of $c^L, c^{R2}$ is of $n^{\beta_1}, n^{\beta_2}$ with $\beta_1, \beta_2 < 1$. At this point, we can set $\lambda, \mu$ and $\Lambda, \gamma, \tau$ and $\zeta$ the same way as paramater settings in Theorem 3.5.1. As for $\varepsilon_2$, we will pick it such that $1 - \varepsilon_1 - 2\varepsilon_2 > 0$.

**Proof Sketch.** The proof of this theorem use exactly the same hybrid argument as the proof of Theorem 3.5.1. The only difference is to note that when $\delta = o(\log n)$, $(1 - 1/2^\delta)^{\frac{n^{1-\varepsilon_1-2\varepsilon_2}}{4\delta^2}}$ is negligible as long as $1 - \varepsilon_1 - 2\varepsilon_2 > 0$. Since the proof is simply a repetition of Section 3.6, we omit it for the ease of presentation.

[2]↑Recall $(c^L, c^R) = \mathrm{NMEnc}_0(E, e)$.

# B. APPENDIX FOR CHAPTER 4

## B.1 Message Authentication Code: Choice of Parameters

**Lemma B.1.1.** *Suppose* $\{h_k : \{0,1\}^\alpha \longrightarrow \{0,1\}^\beta\}$ *is a* $\mu$*-almost pairwise independent hash family. Then the following family of pair of functions is a* $\mu$*-secure message authentication code.*

$$\left\{\begin{array}{c} \mathrm{Tag}_k(x) = h_k(x) \\ \mathrm{Verify}_k(x,y) = 1 \text{ if and only if } y = h_k(x) \end{array}\right\}_{k\in K}$$

*Proof.* Obviously, for all $m, k$, $\mathrm{Verify}(m, h_k(m)) = 1$. Also, for all $m \neq m'$ and $t, t'$,

$$\Pr_{k\leftarrow U_K}\Big[\,\mathrm{Tag}_k(m') = t' \Big|\, \mathrm{Tag}_k(m) = t\Big] = \frac{\Pr_{k\leftarrow U_K}\left[\,\mathrm{Tag}_k(m') = t' \wedge \mathrm{Tag}_k(m) = t\right]}{\Pr_{k\leftarrow U_K}\left[\,\mathrm{Tag}_k(m) = t\right]} \leqslant \frac{\mu\cdot 2^{-\beta}}{2^{-\beta}} = \mu$$

□

**Lemma B.1.2.** *Suppose* $\alpha = \ell\cdot\beta$ *and write* $m$ *as* $(m_1, m_2, \ldots, m_\ell)$ *where* $m_i \in \{0,1\}^\beta$*. Let* $K = \{0,1\}^\beta \times \{0,1\}^\beta$ *and write* $k$ *as* $(k_1, k_2)$*. Define* $h_{k_1,k_2}(m) = k_1 + m_1k_2 + m_2k_2^2 + \cdots + m_\ell k_2^\ell$*, which is seen as a polynomial in* $\mathbb{GF}[2^\beta]$*. Then* $\{h_k\}$ *is a* $\frac{\alpha}{\beta\cdot 2^\beta}$*-almost pairwise independent hash family.*

*Proof.* For all $m, t$,

$$\Pr_{k\leftarrow U_{2\beta}}[h_k(m) = t] = \Pr_{k_2\leftarrow U_\beta}\left[\Pr_{k_1\leftarrow U_\beta}\left[k_1 + m_1k_2 + m_2k_2^2 + \cdots + m_\ell k_2^\ell = t\right]\right] = \Pr_{k_2\leftarrow U_\beta}\left[2^{-\beta}\right] = 2^{-\beta}$$

For all $m \neq m'$ and $t, t'$,

$$\begin{aligned}
&\Pr_{k\leftarrow U_{2\beta}}[h_k(m) = t \wedge h_k(m') = t'] \\
&= \Pr_{k_1\leftarrow U_\beta, k_2\leftarrow U_\beta}\left[k_1 + m_1k_2 + m_2k_2^2 + \cdots + m_\ell k_2^\ell = t \wedge k_1 + m_1'k_2 + m_2'k_2^2 + \cdots + m_\ell' k_2^\ell = t'\right] \\
&= \Pr_{k_2\leftarrow U_\beta}\left[\sum_{i=1}^{\ell}(m_i - m_i')k_2^i = t - t'\right]\cdot \Pr_{k_1\leftarrow U_\beta}\left[k_1 + m_1k_2 + m_2k_2^2 + \cdots + m_\ell k_2^\ell = t\right] \\
&\leqslant \frac{\ell}{2^\beta}\cdot 2^{-\beta}
\end{aligned}$$

where the last inequality is because a degree $\ell$ polynomial in a field can have at most $\ell$ many zeros. Since $\ell = \alpha/\beta$, this completes the proof. □

**Corollary B.1.1.** *For all message length $\alpha$ and Tag length $\beta$, there exists a $\frac{\alpha}{2^\beta}$-secure message authentication code scheme with key length $2\beta$.*

## B.2 Proof of 3-Split-State Non-malleability (Theorem 4.1.3)

Here we will prove that the encoding scheme shown in Figure 4.4 is secure against 3-split-state tampering. More formally, we will show that there exists a simulator $\mathsf{Sim}_{f,g,h}$ such that

$$\left\{\begin{array}{c} \Big(c,(w,L),R\Big) \leftarrow \mathsf{Enc}(m) \\ \widetilde{c} = f(c),\ \ (\widetilde{w},\widetilde{L}) = g(w,L),\ \ \widetilde{R} = h(R) \\ \text{Output: } \tilde{m} = \mathsf{Dec}\Big(\widetilde{c},(\widetilde{w},\widetilde{L}),\widetilde{R}\Big) \end{array}\right\} = \mathsf{Tamper}^{m}_{f,g,h} \approx \mathsf{Copy}\Big(\mathsf{Sim}_{f,g,h}, m\Big)$$

Our first hybrid is exactly the same as $\mathsf{Tamper}^{m}_{f,g,h}$. We just open up the definition of $\mathsf{Enc}$ and $\mathsf{Dec}$.

$H_0(f,g,h,m)$:

1. $w \leftarrow U_n,\ s \leftarrow U_d,\ k_1 \leftarrow U_\gamma,\ k_2 \leftarrow U_{\gamma'}$
2. $r = \mathrm{Ext}(w,s)$, $c = m \oplus r$, $t_1 = \mathrm{Tag}_{k_1}(c)$, $t_2 = \mathrm{Tag}'_{k_2}(w)$
3. $(L,R) \leftarrow \mathsf{Enc}^+(k_1,k_2,t_1,t_2,s)$
4. $\widetilde{c} = f(c)$, $(\widetilde{w},\widetilde{L}) = g(w,L)$, $\widetilde{R} = h(R)$
5. $(\widetilde{k_1},\widetilde{k_2},\widetilde{t_1},\widetilde{t_2},\widetilde{s}) = \mathsf{Dec}^+(\widetilde{L},\widetilde{R})$
6. If $(\widetilde{k_1},\widetilde{k_2},\widetilde{t_1},\widetilde{t_2},\widetilde{s}) = \perp$, output $\perp$
7. Else if $\Big(\mathrm{Verify}_{\widetilde{k_1}}(\widetilde{c},\widetilde{t_1})=0$ or $\mathrm{Verify}'_{\widetilde{k_2}}(\widetilde{w},\widetilde{t_2}) = 0\Big)$, output $\perp$
8. Else output $\widetilde{c} \oplus \mathrm{Ext}(\widetilde{w},\widetilde{s})$

In the next hybrid, we re-write $(\widetilde{w},\widetilde{L}) = g(w,L)$ as $\widetilde{w} = g_L(w)$ and $\widetilde{L} = g_w(L)$. The hybrids $H_0(f,g,h,m)$ and $H_1(f,g,h,m)$ are identical.

$H_1(f, g, h, m)$:

1. $w \leftarrow U_n,\ s \leftarrow U_d,\ k_1 \leftarrow U_\gamma,\ k_2 \leftarrow U_{\gamma'}$
2. $r = \mathrm{Ext}(w, s)$, $c = m \oplus r$, $t_1 = \mathrm{Tag}_{k_1}(c)$, $t_2 = \mathrm{Tag}'_{k_2}(w)$
3. $\widetilde{c} = f(c)$
4. $(L, R) \leftarrow \mathsf{Enc}^+(k_1, k_2, t_1, t_2, s)$
5. $\widetilde{L} = g_w(L)$, $\widetilde{R} = h(R)$
6. $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s}) = \mathsf{Dec}^+(\widetilde{L}, \widetilde{R})$
7. $\widetilde{w} = g_L(w)$
8. If $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s}) = \perp$, output $\perp$
9. Else if $\Big(\mathrm{Verify}_{\widetilde{k_1}}(\widetilde{c}, \widetilde{t_1})=0$ or $\mathrm{Verify}'_{\widetilde{k_2}}(\widetilde{w}, \widetilde{t_2}) = 0\Big)$, output $\perp$
10. Else output $\widetilde{c} \oplus \mathrm{Ext}(\widetilde{w}, \widetilde{s})$

Notice that step 4,5,6 in $H_1(f, g, h, m)$ is exactly $\mathrm{TamperPlus}^{(k_1,k_2,t_1,t_2,s)}_{g_w,h}$, replace this with simulator $\mathrm{SimPlus}_{g_w,h}$ gives us $H_2(f, g, h, m)$. We note that hybrids $H_1(f, g, h, m)$ and $H_2(f, g, h, m)$ are $\varepsilon^+$-close. If not, we can use the tampering function $(g_w, h)$ and message $(k_1, k_2, t_1, t_2, s)$ to break the $\varepsilon^+$ augmented non-malleability of $(\mathrm{Enc}^+, \mathrm{Dec}^+)$.

$H_2(f, g, h, m)$:

1. $w \leftarrow U_n,\ s \leftarrow U_d,\ k_1 \leftarrow U_\gamma,\ k_2 \leftarrow U_{\gamma'}$
2. $r = \mathrm{Ext}(w, s)$, $c = m \oplus r$, $t_1 = \mathrm{Tag}_{k_1}(c)$, $t_2 = \mathrm{Tag}'_{k_2}(w)$
3. $\widetilde{c} = f(c)$
4. $(L, \mathrm{Ans}) \leftarrow \mathrm{SimPlus}_{g_w,h}$
5. $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s}) = \mathsf{Copy}\Big(\mathrm{Ans}, (k_1, k_2, t_1, t_2, s)\Big)$
6. $\widetilde{w} = g_L(w)$
7. If $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s}) = \perp$, output $\perp$
8. Else if $\Big(\mathrm{Verify}_{\widetilde{k_1}}(\widetilde{c}, \widetilde{t_1})=0$ or $\mathrm{Verify}'_{\widetilde{k_2}}(\widetilde{w}, \widetilde{t_2}) = 0\Big)$, output $\perp$
9. Else output $\widetilde{c} \oplus \mathrm{Ext}(\widetilde{w}, \widetilde{s})$

Now, we open up the different cases of Ans. This hybrid is identical to the previous one.

$H_3(f, g, h, m)$:

1. $w \leftarrow U_n,\ s \leftarrow U_d,\ k_1 \leftarrow U_\gamma,\ k_2 \leftarrow U_{\gamma'}$
2. $r = \mathrm{Ext}(w, s)$, $c = m \oplus r$, $t_1 = \mathrm{Tag}_{k_1}(c)$, $t_2 = \mathrm{Tag}'_{k_2}(w)$
3. $\widetilde{c} = f(c)$
4. $(L, \mathrm{Ans}) \leftarrow \mathrm{SimPlus}_{g_w, h}$
5. $\widetilde{w} = g_L(w)$
6. If Ans =
   - Case $\perp$: Output $\perp$
   - Case same$^*$: If $\Big(\mathrm{Verify}_{k_1}(\widetilde{c}, t_1){=}0$ or $\mathrm{Verify}'_{k_2}(\widetilde{w}, t_2) = 0\Big)$, output $\perp$
     Else output $\widetilde{c} \oplus \mathrm{Ext}(\widetilde{w}, s)$
   - Case $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s})$: If $\Big(\mathrm{Verify}_{\widetilde{k_1}}(\widetilde{c}, \widetilde{t_1}){=}0$ or $\mathrm{Verify}'_{\widetilde{k_2}}(\widetilde{w}, \widetilde{t_2}) {=}0\Big)$, output $\perp$
     Else output $\widetilde{c} \oplus \mathrm{Ext}(\widetilde{w}, \widetilde{s})$

Now we use the properties of message authentication codes.

$H_4(f, g, h, m)$:

Copy

$\Big($

1. $w \leftarrow U_n,\ s \leftarrow U_d,\ k_1 \leftarrow U_\gamma,\ k_2 \leftarrow U_{\gamma'}$
2. $r = \mathrm{Ext}(w, s)$, $c = m \oplus r$, $t_1 = \mathrm{Tag}_{k_1}(c)$, $t_2 = \mathrm{Tag}'_{k_2}(w)$
3. $\widetilde{c} = f(c)$
4. $(L, \mathrm{Ans}) \leftarrow \mathrm{SimPlus}_{g_w, h}$
5. $\widetilde{w} = g_L(w)$
6. If Ans =
   - Case $\perp$: Output $\perp$
   - Case same$^*$: If $\Big(\widetilde{c} = c$ and $\widetilde{w} = w\Big) = 1$, output same$^*$
     Else output $\perp$
   - Case $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s})$: If $\Big(\mathrm{Verify}_{\widetilde{k_1}}(\widetilde{c}, \widetilde{t_1}){=}0$ or $\mathrm{Verify}'_{\widetilde{k_2}}(\widetilde{w}, \widetilde{t_2}) = 0\Big)$, output $\perp$
     Else output $\widetilde{c} \oplus \mathrm{Ext}(\widetilde{w}, \widetilde{s})$

$, m\Big)$

Clean up and remove the redundant steps.

$H_5(f, g, h, m)$:

Copy

$\Big($

1. $w \leftarrow U_n,\ s \leftarrow U_d,\ r = \mathrm{Ext}(w, s),\ c = m \oplus r$
2. $(L, \mathrm{Ans}) \leftarrow \mathrm{SimPlus}_{g_w, h}$
3. $\widetilde{c} = f(c),\ \widetilde{w} = g_L(w)$
4. If Ans $=$
   - Case $\perp$: Output $\perp$
   - Case $\mathsf{same}^*$: If $\Big(\widetilde{c} = c$ and $\widetilde{w} = w\Big) = 1$, output $\mathsf{same}^*$
     Else output $\perp$
   - Case $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s})$: If $\Big(\mathrm{Verify}_{\widetilde{k_1}}(\widetilde{c}, \widetilde{t_1})$=0 or $\mathrm{Verify}'_{\widetilde{k_2}}(\widetilde{w}, \widetilde{t_2}) = 0\Big)$, output $\perp$
     Else output $\widetilde{c} \oplus \mathrm{Ext}(\widetilde{w}, \widetilde{s})$

$, m\Big)$

Now, compute the leakage about $w$ we need in the first part of the hybrid.

$H_6(f, g, h, m)$:

Copy

$\Big($

1. $w \leftarrow U_n$
2. $(L, \mathrm{Ans}) \leftarrow \mathrm{SimPlus}_{g_w, h},\ \widetilde{w} = g_L(w)$
3. If Ans $=$
   - Case $\mathsf{same}^*$: $\mathrm{flag}_1 = 1$ iff $(\widetilde{w} = w)$
   - Case $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s})$: $\mathrm{flag}_2 = 1$ iff $\mathrm{Verify}'_{\widetilde{k_2}}(\widetilde{w}, \widetilde{t_2}) = 1$.
     Set mask $= \mathrm{Ext}(\widetilde{w}, \widetilde{s})$.
4. $s \leftarrow U_d,\ r = \mathrm{Ext}(w, s),\ c = m \oplus r,\ \widetilde{c} = f(c)$
5. If Ans $=$
   - Case $\perp$: Output $\perp$
   - Case $\mathsf{same}^*$: If $\Big(\widetilde{c} = c$ and $\mathrm{flag}_1\Big) = 1$, output $\mathsf{same}^*$
     Else output $\perp$
   - Case $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s})$: If $\Big(\mathrm{Verify}_{\widetilde{k_1}}(\widetilde{c}, \widetilde{t_1})$=0 or $\mathrm{flag}_2 = 0\Big)$ output $\perp$
     Else output $\widetilde{c} \oplus$ mask

$, m\Big)$

Formally define the information as a leakage function of $w$.

$H_7(f, g, h, m)$:

Copy

$\Big($

1. $w \leftarrow U_n$
2. For the tampering function $(g, h)$ we define the following leakage function $\mathcal{L}(w) : \{0,1\}^n \longrightarrow \{0,1\}^{\beta+\beta'+\gamma+\gamma'+d+1} \times \{0,1\} \times \{0,1\} \times \{0,1\}^\ell$
   (a) $(L, \text{Ans}) \leftarrow \text{SimPlus}_{g_w, h}$, $\widetilde{w} = g_L(w)$
   (b) If Ans =
   - Case same*: $\text{flag}_1 = 1$ iff $(\widetilde{w} = w)$
   - Case $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s})$: $\text{flag}_2 = 1$ iff $\text{Verify}'_{\widetilde{k_2}}(\widetilde{w}, \widetilde{t_2}) = 1$
     Set mask $= \text{Ext}(\widetilde{w}, \widetilde{s})$

   (c) $\mathcal{L}(w) := (\text{Ans}, \text{flag}_1, \text{flag}_2, \text{mask})$
3. $s \leftarrow U_d$, $r = \text{Ext}(w, s)$, $c = m \oplus r$, $\widetilde{c} = f(c)$
4. If Ans =
   - Case $\perp$: Output $\perp$
   - Case same*: If $\Big(\widetilde{c} = c$ and $\text{flag}_1\Big) = 1$, output same*
     Else output $\perp$
   - Case $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s})$: If $\Big($ $\text{Verify}_{\widetilde{k_1}}(\widetilde{c}, \widetilde{t_1})$=0 or $\text{flag}_2 = 0\Big)$, output $\perp$
     Else output $\widetilde{c} \oplus$ mask

$\Big), m\Big)$

Using the property of average min-entropy extractor to replace the extraction step with uniform random bits.

$H_8(f, g, h, m)$:

Copy

1. $w \leftarrow U_n$
2. For the tampering function $(g, h)$ we define the following leakage function $\mathcal{L}(w)$ : $\{0,1\}^n \longrightarrow \{0,1\}^{\beta+\beta'+\gamma+\gamma'+d+1} \times \{0,1\} \times \{0,1\} \times \{0,1\}^\ell$
   (a) $(L, \text{Ans}) \leftarrow \text{SimPlus}_{g_w, h}$, $\widetilde{w} = g_L(w)$
   (b) If Ans =
   - Case $\mathsf{same}^*$: $\text{flag}_1 = 1$ iff $(\widetilde{w} = w)$
   - Case $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s})$: $\text{flag}_2 = 1$ iff $\text{Verify}'_{\widetilde{k_2}}(\widetilde{w}, \widetilde{t_2}) = 1$
     Set mask $= \text{Ext}(\widetilde{w}, \widetilde{s})$

   (c) $\mathcal{L}(w) := (\text{Ans}, \text{flag}_1, \text{flag}_2, \text{mask})$

$\Big($ , $m\Big)$

3. $r \leftarrow U_\ell$, $c = m \oplus r$, $\widetilde{c} = f(c)$
4. If Ans =
   - Case $\perp$: Output $\perp$
   - Case $\mathsf{same}^*$: If $\Big(\widetilde{c} = c$ and $\text{flag}_1\Big) = 1$, output $\mathsf{same}^*$
     Else output $\perp$
   - Case $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s})$: If $\Big(\text{Verify}_{\widetilde{k_1}}(\widetilde{c}, \widetilde{t_1})$=0 or $\text{flag}_2 = 0\Big)$, output $\perp$
     Else output $\widetilde{c} \oplus$ mask

Finally, fixing the message to $0^\ell$ would not affect the distribution of the output of our hybrid. This last hybrid is our simulator.

$H_9(f, g, h, m)$:

Copy

$\Big($

1. $w \leftarrow U_n$
2. For the tampering function $(g, h)$ we define the following leakage function $\mathcal{L}(w)$ : $\{0,1\}^n \longrightarrow \{0,1\}^{\beta+\beta'+\gamma+\gamma'+d+1} \times \{0,1\} \times \{0,1\} \times \{0,1\}^\ell$
   (a) $(L, \text{Ans}) \leftarrow \text{SimPlus}_{g_w, h}$, $\widetilde{w} = g_L(w)$
   (b) If Ans =
   - Case $\mathsf{same}^*$: $\text{flag}_1 = 1$ iff $(\widetilde{w} = w)$
   - Case $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s})$: $\text{flag}_2 = 1$ iff $\text{Verify}'_{\widetilde{k_2}}(\widetilde{w}, \widetilde{t_2}) = 1$
     Set mask $= \text{Ext}(\widetilde{w}, \widetilde{s})$

   (c) $\mathcal{L}(w) := (\text{Ans}, \text{flag}_1, \text{flag}_2, \text{mask})$
3. $r \leftarrow U_\ell$, $c = 0^\ell \oplus r$, $\widetilde{c} = f(c)$
4. If Ans =
   - Case $\perp$: Output $\perp$
   - Case $\mathsf{same}^*$: If $\Big(\widetilde{c} = c$ and $\text{flag}_1\Big) = 1$, output $\mathsf{same}^*$; else output $\perp$
   - Case $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s})$: If $\Big(\text{Verify}_{\widetilde{k_1}}(\widetilde{c}, \widetilde{t_1})=0$ or $\text{flag}_2 = 0\Big)$, output $\perp$
     Else output $\widetilde{c} \oplus$ mask

$\Big), m\Big)$

## B.3 Proof of Non-malleability against Forgetful Functions (Theorem 4.5.1)

In this section, we shall prove Theorem 4.5.1.

Now we divide the proof of non-malleability into two parts. In Appendix B.3.1, we show our coding scheme is non-malleable against tampering from $\mathcal{FOR}_{n_1,n_2,n_3,n_4-\{1\}} \cup \mathcal{FOR}_{n_1,n_2,n_3,n_4-\{3\}}$. In Appendix B.3.2, we show non-malleability against $\mathcal{LA}_{n_1,n_2} \times \mathcal{LA}_{n_3,n_4}$. Together they prove the non-malleability of our coding scheme.

### B.3.1 Non-malleability against $\mathcal{FOR}_{n_1,n_2,n_3,n_4-\{1\}} \cup \mathcal{FOR}_{n_1,n_2,n_3,n_4-\{3\}}$

In this section, for codeword $c = (c_1, c_2, \ldots, c_k)$, we write $c_{-i}$ to denote $(c_1, \ldots, c_{i-1}, c_{i+1}, \ldots, c_k)$. Intuitively, our scheme is secure when the tampering function forget about the first or third state because forgetting any one of those two states essentially means forgetting about the

message. Specifically, if we use $c^m$ to denote the random variable $\mathsf{Enc}(m)$, we are going to show that for all $m \neq m'$,

$$c^m_{-i} \approx_{\varepsilon_1} c^{m'}_{-i} \quad i = 1 \text{ or } 3 \tag{B.1}$$

Recall $\varepsilon_1$ is the error of our extractor Ext. This would immediately imply non-malleability because for all $f \in \mathcal{FOR}_{n_1,n_2,n_3,n_4-\{i\}}$, we could write (see Section 4.5 for definition of forgetful family)

$$\mathsf{Dec}(f(\mathsf{Enc}(m))) = \mathsf{Dec}(g(c^m_{-i})) \approx_{\varepsilon_1} \mathsf{Dec}(g(c^{m'}_{-i})) = \mathsf{Dec}(f(\mathsf{Enc}(m')))$$

We shall prove Equation B.1 for $i = 1$ next. Fix keys $k_1, k_2$, if given leakage $t_2$ and $w_2$, we still have $\widetilde{H}_\infty(w|t_2, w_2) \geqslant k$, by the property of our strong average min-entropy extractor, we have

$$k_1, k_2, t_2, w_2, s, \mathrm{Ext}(w, s) \approx_{\varepsilon_1} k_1, k_2, t_2, w_2, s, U_\ell$$

Therefore, we have (recall we use $r$ to denote $\mathrm{Ext}(w, s)$)

$$k_1, k_2, t_2, w_2, s, r \oplus m \approx_{\varepsilon_1} k_1, k_2, t_2, w_2, s, r \oplus m'$$

which leads to (since $t_1$ is a deterministic function of $k_1$ and $c = r \oplus m$)

$$(k_1, k_2, t_1, t_2, s), w_2, r \oplus m \approx_{\varepsilon_1} (k_1, k_2, t_1, t_2, s), w_2, r \oplus m'$$

which implies

$$R, (w_2, L), r \oplus m \approx_{\varepsilon_1} R, (w_2, L), r \oplus m'$$

which is equivalent to

$$c^m_{-1} \approx_{\varepsilon_1} c^{m'}_{-1}$$

Using similar arguments, as long as $\widetilde{H}_\infty(w|t_2, w_1) \geqslant k$, we have

$$c^m_{-3} \approx c^{m'}_{-3}$$

Note that this also requires $w_2$ to have length $\ell + o(\ell)$.

### B.3.2 Non-malleability against $\mathcal{LA}_{n_1,n_2} \times \mathcal{LA}_{n_3,n_4}$

In order to prove non-malleability, we need to show that for all tampering $(f, g) \in \mathcal{LA}_{n_1,n_2} \times \mathcal{LA}_{n_3,n_4}$, where $f = (f^{(1)}, f^{(2)})$ and $g = (g^{(1)}, g^{(2)})$, there exists a simulator $\mathsf{Sim}_{f,g}$ such that for all $m$,

$$\left\{\begin{array}{c} \Big((w_1, R, (w_2, L), c)\Big) \leftarrow \mathsf{Enc}(m) \\ \widetilde{w_1} = f^{(1)}(w_1),\ \widetilde{R} = f^{(2)}(w_1, R) \\ (\widetilde{w_2}, \widetilde{L}) = g^{(1)}(w_2, L),\ \widetilde{c} = g^{(2)}(w_2, L, c) \\ \text{Output: } \tilde{m} = \mathsf{Dec}\Big(\widetilde{w_1}, \widetilde{R}, (\widetilde{w_2}, \widetilde{L}), \widetilde{c}\Big) \end{array}\right\} = \mathsf{Tamper}^m_{f,g} \approx \mathsf{Copy}\Big(\mathsf{Sim}_{f,g}\ ,\ m\Big)$$

The following hybrids will lead us from tampering experiment to the simulator.

$H_0(f, g, m)$:

1. $w_1 \leftarrow U_n,\ w_2 \leftarrow U_{n'}\ s \leftarrow U_d,\ k_1 \leftarrow U_\gamma,\ k_2 \leftarrow U_{\gamma'}$. Let $w := (w_1, w_2)$
2. $r = \mathrm{Ext}(w, s)$, $c = m \oplus r$, $t_1 = \mathrm{Tag}_{k_1}(c)$, $t_2 = \mathrm{Tag}'_{k_2}(w)$
3. $(L, R) \leftarrow \mathsf{Enc}^+(k_1, k_2, t_1, t_2, s)$
4. $\widetilde{w_1} = f^{(1)}(w_1)$, $\widetilde{R} = f^{(2)}(w_1, R),\ (\widetilde{w_2}, \widetilde{L}) = g^{(1)}(w_2, L)$, $\widetilde{c} = g^{(2)}(w_2, L, c)$. Let $\widetilde{w} = (\widetilde{w_1}, \widetilde{w_2})$
5. $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s}) = \mathsf{Dec}^+(\widetilde{L}, \widetilde{R})$
6. If $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s}) = \perp$, output $\perp$
7. Else if $\Big(\mathrm{Verify}_{\widetilde{k_1}}(\widetilde{c}, \widetilde{t_1})$=0 or $\mathrm{Verify}'_{\widetilde{k_2}}(\widetilde{w}, \widetilde{t_2}) = 0\Big)$, output $\perp$
8. Else output $\widetilde{c} \oplus \mathrm{Ext}(\widetilde{w}, \widetilde{s})$

Decompose the shaded equation into individual tampering equations.

$H_1(f, g, m)$:

1. $w_1 \leftarrow U_n,\ w_2 \leftarrow U_{n'},\ s \leftarrow U_d,\ k_1 \leftarrow U_\gamma,\ k_2 \leftarrow U_{\gamma'}$. Let $w := (w_1, w_2)$
2. $r = \mathrm{Ext}(w, s)$, $c = m \oplus r$, $t_1 = \mathrm{Tag}_{k_1}(c)$, $t_2 = \mathrm{Tag}'_{k_2}(w)$
3. $\widetilde{w_1} = f^{(1)}(w_1)$
4. $(L, R) \leftarrow \mathsf{Enc}^+(k_1, k_2, t_1, t_2, s)$
5. $\widetilde{L} = g^{(1)}_{w_2}(L)$, $\widetilde{R} = f^{(2)}_{w_1}(R)$
6. $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s}) = \mathsf{Dec}^+(\widetilde{L}, \widetilde{R})$
7. $\widetilde{c} = g^{(2)}_{w_2, L}(c)$, $\widetilde{w_2} = g^{(1)}_L(w_2)$. Let $\widetilde{w} := (\widetilde{w_1}, \widetilde{w_2})$
8. If $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s}) = \perp$, output $\perp$
9. Else if $\Big(\mathrm{Verify}_{\widetilde{k_1}}(\widetilde{c}, \widetilde{t_1})=0$ or $\mathrm{Verify}'_{\widetilde{k_2}}(\widetilde{w}, \widetilde{t_2}) = 0\Big)$, output $\perp$
10. Else output $\widetilde{c} \oplus \mathrm{Ext}(\widetilde{w}, \widetilde{s})$

Use SimPlus to replace the tampering experiment of augmented 2-state non-malleable code.

$H_2(f, g, m)$:

1. $w_1 \leftarrow U_n,\ w_2 \leftarrow U_{n'},\ s \leftarrow U_d,\ k_1 \leftarrow U_\gamma,\ k_2 \leftarrow U_{\gamma'}$. Let $w := (w_1, w_2)$
2. $r = \mathrm{Ext}(w, s)$, $c = m \oplus r$, $t_1 = \mathrm{Tag}_{k_1}(c)$, $t_2 = \mathrm{Tag}'_{k_2}(w)$
3. $\widetilde{w_1} = f^{(1)}(w_1)$
4. $(L, \mathrm{Ans}) \leftarrow \mathrm{SimPlus}_{g^{(1)}_{w_2}, f^{(2)}_{w_1}}$
5. $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s}) = \mathsf{Copy}\Big(\mathrm{Ans}, (k_1, k_2, t_1, t_2, s)\Big)$
6. $\widetilde{c} = g^{(2)}_{w_2, L}(c)$, $\widetilde{w_2} = g^{(1)}_L(w_2)$. Let $\widetilde{w} := (\widetilde{w_1}, \widetilde{w_2})$
7. If $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s}) = \perp$, output $\perp$
8. Else if $\Big(\mathrm{Verify}_{\widetilde{k_1}}(\widetilde{c}, \widetilde{t_1})=0$ or $\mathrm{Verify}'_{\widetilde{k_2}}(\widetilde{w}, \widetilde{t_2}) = 0\Big)$, output $\perp$
9. Else output $\widetilde{c} \oplus \mathrm{Ext}(\widetilde{w}, \widetilde{s})$

Rearrange steps.

$H_3(f, g, m)$:

1. $w_1 \leftarrow U_n,\ w_2 \leftarrow U_{n'},\ s \leftarrow U_d,\ k_1 \leftarrow U_\gamma,\ k_2 \leftarrow U_{\gamma'}$ Let $w := (w_1, w_2)$
2. $r = \mathrm{Ext}(w, s)$, $c = m \oplus r$, $t_1 = \mathrm{Tag}_{k_1}(c)$, $t_2 = \mathrm{Tag}'_{k_2}(w)$
3. $\widetilde{w_1} = f^{(1)}(w_1)$
4. $(L, \mathrm{Ans}) \leftarrow \mathrm{SimPlus}_{g^{(1)}_{w_2}, f^{(2)}_{w_1}}$
5. $\widetilde{c} = g^{(2)}_{w_2, L}(c)$, $\widetilde{w_2} = g^{(1)}_{L}(w_2)$. Let $\widetilde{w} := (\widetilde{w_1}, \widetilde{w_2})$
6. If Ans $=$
   - Case $\perp$: Output $\perp$
   - Case same*: If $\Big(\mathrm{Verify}_{k_1}(\widetilde{c}, t_1){=}0$ or $\mathrm{Verify}'_{k_2}(\widetilde{w}, t_2) = 0\Big)$, output $\perp$
     Else output $\widetilde{c} \oplus \mathrm{Ext}(\widetilde{w}, s)$
   - Case $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s})$: If $\Big(\mathrm{Verify}_{\widetilde{k_1}}(\widetilde{c}, \widetilde{t_1}){=}0$ or $\mathrm{Verify}'_{\widetilde{k_2}}(\widetilde{w}, \widetilde{t_2}) =$, output $\perp$
     Else output $\widetilde{c} \oplus \mathrm{Ext}(\widetilde{w}, \widetilde{s})$

Use the property of message authentication codes.

$H_4(f, g, m)$:

Copy

$\Big($

1. $w_1 \leftarrow U_n,\ w_2 \leftarrow U_{n'},\ s \leftarrow U_d,\ k_1 \leftarrow U_\gamma,\ k_2 \leftarrow U_{\gamma'}$ Let $w := (w_1, w_2)$
2. $r = \mathrm{Ext}(w, s)$, $c = m \oplus r$, $t_1 = \mathrm{Tag}_{k_1}(c)$, $t_2 = \mathrm{Tag}'_{k_2}(w)$
3. $\widetilde{w_1} = f^{(1)}(w_1)$
4. $(L, \mathrm{Ans}) \leftarrow \mathrm{SimPlus}_{g^{(1)}_{w_2}, f^{(2)}_{w_1}}$
5. $\widetilde{c} = g^{(2)}_{w_2, L}(c)$, $\widetilde{w_2} = g^{(1)}_{L}(w_2)$. Let $\widetilde{w} := (\widetilde{w_1}, \widetilde{w_2})$
6. If Ans $=$
   - Case $\perp$: Output $\perp$
   - Case same*: If $\Big(\widetilde{c} = c$ and $\widetilde{w} = w\Big) = 1$, output same*
     Else output $\perp$
   - Case $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s})$: If $\Big(\mathrm{Verify}_{\widetilde{k_1}}(\widetilde{c}, \widetilde{t_1}){=}0$ or $\mathrm{Verify}'_{\widetilde{k_2}}(\widetilde{w}, \widetilde{t_2}) = 0\Big)$, output $\perp$
     Else output $\widetilde{c} \oplus \mathrm{Ext}(\widetilde{w}, \widetilde{s})$

$, m\Big)$

Remove the redundant steps.

$H_5(f, g, m)$:

Copy

$\Big($

1. $w_1 \leftarrow U_n,\ w_2 \leftarrow U_{n'},\ s \leftarrow U_d,\ r = \mathrm{Ext}(w, s),\ c = m \oplus r$ Let $w := (w_1, w_2)$
2. $(L, \mathrm{Ans}) \leftarrow \mathrm{SimPlus}_{g^{(1)}_{w_2}, f^{(2)}_{w_1}}$
3. $\widetilde{w_1} = f^{(1)}(w_1),\ \widetilde{w_2} = g^{(1)}_L(w_2)$ Let $\widetilde{w} := (\widetilde{w_1}, \widetilde{w_2}),\ \widetilde{c} = g^{(2)}_{w_2, L}(c)$
4. If Ans $=$
   - Case $\perp$: Output $\perp$
   - Case same*: If $\Big(\widetilde{c} = c$ and $\widetilde{w} = w\Big) = 1$, output same*
     Else output $\perp$
   - Case $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s})$: If $\Big(\mathrm{Verify}_{\widetilde{k_1}}(\widetilde{c}, \widetilde{t_1})$=0 or $\mathrm{Verify}'_{\widetilde{k_2}}(\widetilde{w}, \widetilde{t_2}) = 0\Big)$, output $\perp$
     Else output $\widetilde{c} \oplus \mathrm{Ext}(\widetilde{w}, \widetilde{s})$

$, m\Big)$

Process the leakage on $w$ in the first part of our hybrid and only use the leakage in the remainder of our hybrid.

$H_6(f, g, m)$:

Copy

$\Big($

1. $w_1 \leftarrow U_n,\ w_2 \leftarrow U_{n'}$. Let $w := (w_1, w_2)$
2. $(L, \mathrm{Ans}) \leftarrow \mathrm{SimPlus}_{g^{(1)}_{w_2}, f^{(2)}_{w_1}},\ \widetilde{w_1} = f^{(1)}(w_1),\ \widetilde{w_2} = g^{(1)}_L(w_2)$ Let $\widetilde{w} := (\widetilde{w_1}, \widetilde{w_2})$
3. If Ans $=$
   - Case same*: If $(\widetilde{w} = w)$, $\mathrm{flag}_1 = 1$; Else $\mathrm{flag}_1 = 0$
   - Case $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s})$: If $\Big(\mathrm{Verify}_{\widetilde{k_1}}(\widetilde{w}, \widetilde{t_2})\Big) = 1, \mathrm{flag}_2 = 1$, Else $\mathrm{flag}_2 = 0$
     Let mask $= \mathrm{Ext}(\widetilde{w}, \widetilde{s})$
4. $s \leftarrow U_d,\ r = \mathrm{Ext}(w, s),\ c = m \oplus r,\ \widetilde{c} = g^{(2)}_{w_2, L}(c)$
5. If Ans $=$
   - Case $\perp$: Output $\perp$
   - Case same*: If $\Big(\widetilde{c} = c$ and $\mathrm{flag}_1\Big) = 1$, output same*; Else output $\perp$
   - Case $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s})$: If $\Big(\mathrm{Verify}_{\widetilde{k_1}}(\widetilde{c}, \widetilde{t_1})$=0 or $\mathrm{flag}_2 = 0\Big)$, output $\perp$
     Else output $\widetilde{c} \oplus$ mask

$, m\Big)$

Formally define the leakage function.

$H_7(f, g, m)$:

Copy

1. $w_1 \leftarrow U_n,\ w_2 \leftarrow U_{n'}$ Let $w := (w_1, w_2)$
2. Define leakage function $\mathcal{L}(w) : \{0,1\}^n \longrightarrow \{0,1\}^{n'} \times \{0,1\}^{n_1^+} \times \{0,1\}^{\beta+\beta'+\gamma+\gamma'+d+1} \times \{0,1\} \times \{0,1\} \times \{0,1\}^\ell$ as the following function:
   (a) $(L, \text{Ans}) \leftarrow \text{SimPlus}_{g^{(1)}_{w_2}, f^{(2)}_{w_1}},\ \widetilde{w_1} = f^{(1)}(w_1),\ \widetilde{w_2} = g^{(1)}_L(w_2)$ Let $\widetilde{w} := (\widetilde{w_1}, \widetilde{w_2})$
   (b) If Ans =
   - Case $\perp$: $\text{flag}_1 = 0,\ \text{flag}_2 = 0,\ \text{mask} = 0^\ell$
   - Case same*: If $(\widetilde{w} = w)$, $\text{flag}_1 = 1$; Else $\text{flag}_1 = 0$
     $\text{flag}_2 = 0,\ \text{mask} = 0^\ell$
   - Case $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s})$: $\text{flag}_1 = 0$
     If $\left(\text{Verify}'_{\widetilde{k_2}}(\widetilde{w}, \widetilde{t_2})\right) = 1$, $\text{flag}_2 = 1$; Else $\text{flag}_2 = 0$
     Let $\text{mask} = \text{Ext}(\widetilde{w}, \widetilde{s})$

   (c) $\mathcal{L}(w) := (w_2, L, \text{Ans}, \text{flag}_1, \text{flag}_2, \text{mask})$
3. $s \leftarrow U_d,\ r = \text{Ext}(w, s)$, $c = m \oplus r,\ \widetilde{c} = g^{(2)}_{w_2, L}(c)$
4. If Ans =
   - Case $\perp$: Output $\perp$
   - Case same*: If $\left(\widetilde{c} = c \text{ and } \text{flag}_1\right) = 1$, output same*; Else output $\perp$
   - Case $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s})$: If $\left(\text{Verify}_{\widetilde{k_1}}(\widetilde{c}, \widetilde{t_1})=0 \text{ or } \text{flag}_2 = 0\right)$, output $\perp$
     Else output $\widetilde{c} \oplus \text{mask}$

Use the property of min-entropy extractor to replace extraction step with true uniform bits.

$H_8(f, g, m)$:

Copy

1. $w_1 \leftarrow U_n,\ w_2 \leftarrow U_{n'}$ Let $w := (w_1, w_2)$
2. Define leakage function $\mathcal{L}(w) : \{0,1\}^n \longrightarrow \{0,1\}^{n'} \times \{0,1\}^{n_1^+} \times \{0,1\}^{\beta+\beta'+\gamma+\gamma'+d+1} \times \{0,1\} \times \{0,1\} \times \{0,1\}^\ell$ as the following function:
   (a) $(L, \text{Ans}) \leftarrow \text{SimPlus}_{g^{(1)}_{w_2}, f^{(2)}_{w_1}},\ \widetilde{w_1} = f^{(1)}(w_1),\ \widetilde{w_2} = g^{(1)}_L(w_2)$ Let $\widetilde{w} := (\widetilde{w_1}, \widetilde{w_2})$
   (b) If Ans =
   - Case $\perp$: $\text{flag}_1 = 0,\ \text{flag}_2 = 0,\ \text{mask} = 0^\ell$
   - Case same*: If $(\widetilde{w} = w)$, $\text{flag}_1 = 1$; Else $\text{flag}_1 = 0$
     $\text{flag}_2 = 0,\ \text{mask} = 0^\ell$
   - Case $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s})$: $\text{flag}_1 = 0$
     If $\left(\text{Verify}'_{\widetilde{k_2}}(\widetilde{w}, \widetilde{t_2})\right) = 1$, Else $\text{flag}_2 = 0$
     Let $\text{mask} = \text{Ext}(\widetilde{w}, \widetilde{s})$

   (c) $\mathcal{L}(w) := (w_2, L, \text{Ans}, \text{flag}_1, \text{flag}_2, \text{mask})$
3. $r \leftarrow U_\ell$, $c = m \oplus r$, $\widetilde{c} = g^{(2)}_{w_2, L}(c)$
4. If Ans =
   - Case $\perp$: Output $\perp$
   - Case same*: If $\left(\widetilde{c} = c \text{ and } \text{flag}_1\right) = 1$, output same*; Else output $\perp$
   - Case $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s})$: If $\left(\text{Verify}_{\widetilde{k_1}}(\widetilde{c}, \widetilde{t_1}) = 0 \text{ or } \text{flag}_2 = 0\right)$, output $\perp$
     Else output $\widetilde{c} \oplus \text{mask}$

$\Big($ $\Big)$, $m$

Now, we are finally ready to replace $m$ with $0^\ell$. And this give us the hybrid.

$H_9(f, g, m)$:

Copy

1. $w_1 \leftarrow U_n,\ w_2 \leftarrow U_{n'}$ Let $w := (w_1, w_2)$
2. Define leakage function $\mathcal{L}(w) : \{0,1\}^n \longrightarrow \{0,1\}^{n'} \times \{0,1\}^{n_1^+} \times \{0,1\}^{\beta+\beta'+\gamma+\gamma'+d+1} \times \{0,1\} \times \{0,1\} \times \{0,1\}^\ell$ as the following function:
   (a) $(L, \text{Ans}) \leftarrow \text{SimPlus}_{g^{(1)}_{w_2}, f^{(2)}_{w_1}},\ \widetilde{w_1} = f^{(1)}(w_1),\ \widetilde{w_2} = g^{(1)}_L(w_2)$ Let $\widetilde{w} := (\widetilde{w_1}, \widetilde{w_2})$
   (b) If Ans =
   - Case $\perp$: $\text{flag}_1 = 0,\ \text{flag}_2 = 0,\ \text{mask} = 0^\ell$
   - Case same*: If $(\widetilde{w} = w)$, $\text{flag}_1 = 1$; Else $\text{flag}_1 = 0$
     $\text{flag}_2 = 0,\ \text{mask} = 0^\ell$
   - Case $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s})$: $\text{flag}_1 = 0$
     If $\left(\text{Verify}'_{\widetilde{k_2}}(\widetilde{w}, \widetilde{t_2})\right) = 1$, Else $\text{flag}_2 = 0$
     Let $\text{mask} = \text{Ext}(\widetilde{w}, \widetilde{s})$

   (c) $\mathcal{L}(w) := (w_2, L, \text{Ans}, \text{flag}_1, \text{flag}_2, \text{mask})$
3. $r \leftarrow U_\ell,\ c = 0^\ell \oplus r,\ \widetilde{c} = g^{(2)}_{w_2, L}(c)$
4. If Ans =
   - Case $\perp$: Output $\perp$
   - Case same*: If $\left(\widetilde{c} = c \text{ and } \text{flag}_1\right) = 1$, output same*; Else output $\perp$
   - Case $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s})$: If $\left(\text{Verify}_{\widetilde{k_1}}(\widetilde{c}, \widetilde{t_1})=0 \text{ or } \text{flag}_2 = 0\right)$, output $\perp$
     Else output $\widetilde{c} \oplus \text{mask}$

$\Big($ $, m\Big)$

Notice that in our hybrid argument, we have some additional leakage $w_2$ of $w$, which is of length $\ell + o(\ell)$ by our analysis in Appendix B.3.1. Therefore, the total leakage of $w$ is $2\ell + o(\ell)$ and that makes $w$ of length $3\ell + o(\ell)$ in our construction.