Title
Cryptographic Protocols with Strong Security: Non-Malleable Commitments, Concurrent Zero-Knowledge and Topology-Hiding Multi-Party Computation

Author
Richelson, Silas

Publication Date
2014

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA

Los Angeles

Cryptographic Protocols with Strong Security: Non-Malleable Commitments, Concurrent Zero-Knowledge and Topology-Hiding Multi-Party Computation

A dissertation submitted in partial
satisfaction of the requirements for the
degree Doctor of Philosophy in Mathematics

by

Silas Isaac Richelson

2014

ABSTRACT OF THE DISSERTATION

Cryptographic Protocols with Strong Security: Non-Malleable Commitments, Concurrent Zero-Knowledge and Topology-Hiding Multi-Party Computation

by

Silas Isaac Richelson

Doctor of Philosophy in Mathematics

University of California, Los Angeles, 2014

Professor Rafail Ostrovsky, Chair

Abstract. The idea of protocol security is fundamental in cryptography, and the cryptographic literature is full of different notions of security and different models in which these notions can be achieved. This thesis contains three distinct lines of work, connected by the commonality that the security that they achieve is relevant in today's world of highly interconnected and parallelized networking. We make advancements to the well-studied non-malleable and concurrent security models, and in addition we formulate a new security notion of our own called "topology hiding security." Specifically, we give a new protocol for non-malleable commitment, a new model for constant round concurrent zero knowledge, and a new multi-party computation protocol which achieves topology hiding security for a large family of underlying network graphs.

The dissertation of Silas Isaac Richelson is approved.

Haruzo Hida

Ciprian Manolescu

Vwani P. Roychowdhury

Rafail Ostrovsky, Committee Chair

University of California, Los Angeles

2014

Contents

List of Figures

Acknowledgements

First and foremost, I would like to thank my advisor Rafail Ostrovsky. Without Rafi's patience and generosity with his time and positive energy, I would not have been able to complete this thesis. I feel I owe Rafi an additional special thanks as it was because of his recommendation that I decided to pursue cryptography in the first place. In working with him for just four years I have learned so much, both as a cryptographer and as a person, and I look forward to continuing to work with him in the future.

Secondly I would like to thank Alon Rosen. I went to visit Alon at the Herzliya Interdisciplinary Center (IDC) for an incredibly fun and productive seven months. Alon opened me up to new ideas and an entirely new approach to cryptography. It was an honor to study under Alon as he is the protocol master. And his refreshing attitude toward research makes even the unsuccessful days and weeks enjoyable. I also look forward to a future full of interaction and collaboration with Alon.

Additionally, I would like to thank all the people who made working at UCLA and IDC so pleasant. Special thanks goes to Maggie Albert and Martha Contreras for helping me stay on track and for always being friendly faces around the department. I would also like to thank the group at IDC: Hai Brenner, Margarita Vald, Ilan Orlov, Tal Moran, Alon Rosen, and the many students whose visits overlapped with mine. Working with you has taught me the power and importance of collaboration.

I thank my coauthors, Vipul Goyal, Abhishek Jain, Tal Moran, Ilan Orlov, Rafail Ostrovsky, Alon Rosen, Margarita Vald and Ivan Visconti, as working with you has made this thesis possible. In particular, chapter three contains joint work with Goyal, Rosen and Vald entitled "An Algebraic Approach to Non-Malleability"; chapter four is a combination of the works:

- Vipul Goyal, Abhishek Jain, Rafail Ostrovsky, Silas Richelson, Ivan Visconti. "Concurrent Zero-Knowledge in the Bounded Player Model." TCC, pages 60-79. 2013.

- Vipul Goyal, Abhishek Jain, Rafail Ostrovsky, Silas Richelson, Ivan Visconti. "Constant Round Concurrent Zero-Knowledge in the Bounded Player Model." Asiacrypt, pages 20-41. 2013.

Finally, chapter five is joint work with Moran and Orlov entitled "Topology-Hiding MPC".

I thank my friends in the UCLA math department, as without you I never would have made it through. Because of all of you, Los Angeles feels like home. A special thanks goes to the legendary Jane Sherman. I also thank Jukka Virtanen for teaching me how to navigate the math department in a professional manner, and for sharing his love of poetry with me.

Finally, I would like to thank my family, whose love and support knows no bounds, and Natalia for being so awesome.

Bibliographical Sketch

I received my BA in 2008 from Harvard University. My publications include:

- Vipul Goyal, Abhishek Jain, Rafail Ostrovsky, Silas Richelson, Ivan Visconti. "Concurrent Zero-Knowledge in the Bounded Player Model." TCC, pages 60-79. 2013.
- Vipul Goyal, Abhishek Jain, Rafail Ostrovsky, Silas Richelson, Ivan Visconti. "Constant Round Concurrent Zero-Knowledge in the Bounded Player Model." Asiacrypt, pages 20-41. 2013.

Introduction

When arguing for the security of a cryptographic protocol, Π, one generally considers an adversary $\mathcal{A}$ who attacks Π. Π is considered secure if any $\mathcal{A}$ who can mount a successful attack, can also do something unlikely – like factor integers in polynomial time. Originally, security was considered in the standalone model where $\mathcal{A}$ was only allowed to execute the protocol in his attack. However, in an attempt to achieve security notions which are useful in today's massively interconnected and highly parallelized world, new and more stringent security models were considered. One famous example is concurrent security, which asks the question "what if $\mathcal{A}$ can run many executions of the protocol at the same time?" Non-malleability goes one step further, asking "what if $\mathcal{A}$ can run many protocol executions, and assume different roles in each? Can he use information obtained in one execution to affect his other executions?"

This thesis contains three distinct lines of work, connected by the common theme of achieving levels of security which are relevant to today's world. We make advancements to the well-studied non-malleable and concurrent security models, and in addition we formulate a new model of our own. Specifically, in chapter three we describe a new non-malleable commitment scheme which is vastly superior to all existing non-malleable commitment schemes in terms of round, communication and computation complexity as well as simplicity.

Then in chapter four we give new protocols for concurrent zero-knowledge. The problem of obtaining constant round zero-knowledge has been a prized open problem for more than

a decade and has led to advancement in the theory of protocols. A large body of work is devoted to various relaxations of the plain model in which concurrent zero-knowledge is possible. Our protocols are in another such model called the bounded player model, which is arguably closer to the plain model than all others. We achieve constant round concurrent zero-knowledge in the bounded player model.

Finally, we examine the security question "Can many parties who are connected to each other by some incomplete network graph G run multi-party computation protocols which hide the structure of G?" We feel this is an important security direction to pursue in today's world where the relationship between nodes in a network might reflect sensative information. This question has received very little attention in the literature, and to our knowledge has never been considered in the computational setting where the techniques of cryptography apply. We formulate the notion of *topology hiding security* and give a protocol which achieves it for a large family of network graphs.

Chapter 1

Preliminaries

1.1 Basic Notations

Throughout the rest of the paper we use the following notations. Let λ be the security parameter. For positive integer $n \in \mathbb{N}$, let $[n] = \{1, \ldots, n\}$. We use $\oplus$ to denote the bitwise XOR of strings operation. A function $\varepsilon : \mathbb{N} \to \mathbb{R}^+$ is *negligible* if it tends to 0 faster than any inverse polynomial, i.e., for every constant c there exists $n_c \in \mathbb{N}$ such that $\varepsilon(n) < n^{-c}$ for all $n > n_c$. We use $\mathbf{negl}(\cdot)$ to specify a generic negligible function. We say that an event occurs *with high probability* (whp) or *overwhelming probability* if its probability of occurring is $1 - \mathbf{negl}(\cdot)$. If a function is not negligible we say it is *noticeable*.

Let $\{X_n\}$ and $\{Y_n\}$ be families of probability distributions where for each $n \in \mathbb{N}$, X_n and Y_n are distributions over $\{0,1\}^{p(n)}$ for a polynomial p. We say that $\{X_n\}$ and $\{Y_n\}$ are *computationally indistinguishable*, written $\{X_n\} \approx_c \{Y_n\}$ (or just $X_n \approx_c Y_n$), if there exists a negligible function ε such that for all probabilistic polynomial time (PPT) algorithms A and large enough n,

$$\left| \Pr_{x \leftarrow X_n}\big(A(x) = 1\big) - \Pr_{y \leftarrow Y_n}\big(A(y) = 1\big) \right| < \varepsilon(n).$$

We will speak (usually informally) about interactive protocols. For our purposes, all protocols take place between two parties, modelled as interactive Turing machines. At any point during or after the protocol's execution, the view of a party consists of the party's input, randomness and all of the messages it has received, catalogued appropriately so that the order of messages is clear. The notion of a party's view extends to the setting where many executions of the protocol are executed in an arbitrarily (and possibly adversarially) selected ordering.

1.2 Cryptographic Building Blocks

One-Way Functions. We say that $f : \{0,1\}^* \to \{0,1\}^*$ is a *one-way function* (OWF) if it can be computed in polynomial time, but for any PPT algorithm $\mathcal{A}$,

$$\left|\Pr_{y \xleftarrow{R} f(U_\lambda)}\left(\mathcal{A}(y) \in f^{-1}(\{y\})\right)\right| = \mathbf{negl}(\lambda),$$

where U_λ is the uniform distribution on $\{0,1\}^\lambda$.

Pseudorandom Generators. We say that a deterministic polynomial time computable function $G : \{0,1\}^n \to \{0,1\}^N$ is a *pseudorandom generator* (PRG) if the following properties hold.

Length Extending: $N > n$;

Pseudorandomness: $G(U_n) \approx_c U_N$, where U_n and U_N are the uniform distributions on $\{0,1\}^n$ and $\{0,1\}^N$, respectively.

The notion of a PRG was put forward in [Yao82a]. PRGs exist if and only if one-way functions exist [HILL99].

Collision Resistant Hash Functions. An efficiently computable function ensemble $\mathcal{H} = \{h_\alpha\}_{\alpha\in\{0,1\}^*}$ where $h_\alpha : \{0,1\}^* \to \{0,1\}^{|\alpha|}$ is a family of collision resistant hash functions (CRHF) if for all PPT algorithms $\mathcal{A}$,

$$\left|\Pr_{\alpha\xleftarrow{R}\{0,1\}^\lambda}\big(\mathcal{A}(\alpha) = (x,y) \text{ st } h_\alpha(x) = h_\alpha(y)\big)\right| = \mathbf{negl}(\lambda).$$

Error Correcting Codes. Error correcting codes (ECCs) are fundamental primitives, generally formalized as functions which map a message to a codeword vector and have the property that any distinct messages map to codewords that differ on many coordinates. The ECCs we will use have message space $\{0,1\}^k$ and codeword space $\mathbb{F}^n$ for a finite field $\mathbb{F}$ and maps message $c \in \{0,1\}^k$ to codeword $\hat{c} \in \mathbb{F}^n$. Two important quantities of an ECC are the *rate*, $\frac{k}{n\log q}$, and the *distance*, defined by

$$\Delta := \min_{c\neq c'\in\{0,1\}^k} {}^{\#}\big\{i : \hat{c}_i \neq \hat{c}'_i\big\}.$$

Often the normalized distance $\frac{\Delta}{n}$ is used instead.

The rate and distance together determine the quality of a code. One generally wishes to maximize the distance while keeping the rate as close to 1 as possible. This problem is surprisingly challenging, and the construction of a family of codes whose rate and normalized distance simultaneously do not tend to 0 was a major achievement in the early theory of coding. We will use Reed-Muller codes, a well-known example of such codes which are the multivariate extensions of the classical Reed-Solomon codes [RS60].

When using a Reed-Muller code we will identify our usual message space $\{0,1\}^k$ with $\mathbb{F}[\mathbf{X}]^{\leq d} = \mathbb{F}[X_1,\ldots,X_s]^{\leq d}$, the set of $s-$variate polynomials of degree at most d over $\mathbb{F}$. This implicitly assumes that $k \leq \binom{d+s}{s}\log|\mathbb{F}|$. For a message $c \in \{0,1\}^k$ identified with the

polynomial $c(\mathbf{X})$, we define the codeword $\hat{c}$ to be the evaluation vector

$$\hat{c} = \big(c(\mathbf{a})\big)_{\mathbf{a}\in\mathbb{Z}_q^s} \in \mathbb{Z}_q^{q^s}.$$

Fact 1.1 (**Reed-Muller Distance**). *The code* $\mathbf{RM}[d,s]_q$ *has normalized distance at least* $1-\frac{d}{q}$.

Secret Sharing. A secret sharing scheme gives a way to break a secret into shares in such a way so that each individual share reveals nothing about the secret, but with all the shares together, one can reconstruct the secret. We will use a packed variant of Shamir's secret sharing scheme [Sha79] due to Franklin and Yung [FY92].

The Franklin-Yung packed secret sharing scheme is parametrized by the pair (d,n). The message space is $\mathbb{F}^\ell$ for some field $\mathbb{F}$ and $\ell \le d+1$. Given $\mathbf{m} = (m_1,\ldots,m_\ell) \in \mathbb{F}^\ell$ one constructs n shares of $\mathbf{m}$, denoted $[\mathbf{m}] = \big([\mathbf{m}]_1,\ldots,[\mathbf{m}]_n\big) \in \mathbb{F}^n$ by choosing a random degree d polynomial $f(x) \in \mathbb{F}[x]$ such that $f(\beta_j) = m_j$ for all $j \in [\ell]$, and setting the share $[\mathbf{m}]_i = f(\alpha_i)$ where $\beta_1,\ldots,\beta_\ell,\alpha_1,\ldots,\alpha_n \in \mathbb{F}$ are public values (roots of unity are often chosen in implementations). The security guarantee is that $d+1-\ell$ or fewer shares give no information about the secret $\mathbf{m}$, while $d+1$ or more allow one to reconstruct $\mathbf{m}$ entirely.

1.3 Commitment: "The Digital Analogue of a Sealed Envelope"

A commitment scheme is a two-phase interactive protocol between two parties, a committer C and receiver R. We denote R's view after the first phase (called the commit phase) by $\mathrm{Com}(m;r)$ where m and r are C's secret input and randomness, respectively. In the decommit phase, C reveals m in such a way that R can verify that $\mathrm{Com}(m;r)$ is a commitment to m. In

all of the commitment schemes we will encounter, the decommit phase consists simply of C sending (m, r) to R. The quantity $\mathrm{Com}(m; r)$ must satisfy two security properties: binding and hiding. Informally, binding requires that after the commit phase, C cannot decommit to any value other than m. Hiding stipulates that $\mathrm{Com}(m; r)$ gives R no information about m. As usual, these properties can hold computationally, statistically or perfectly. The statistical binding variant is defined formally below.

Definition 1.1 (Statistically Binding Commitment Scheme). *Let* $\langle \mathrm{C}(m), \mathrm{R} \rangle$ *be a two phase protocol between* C *and* R *where* m *is* C*'s secret input. Let* $z = \mathrm{Com}(m; r)$ *denote* R*'s view after the first phase. Let* $(m, w) = \mathrm{Decom}(m, r, z)$ *be* R*'s view after the second phase. We say that* $\langle \mathrm{C}(m), \mathrm{R} \rangle$ *is a* statistically binding commitment scheme *if the following properties hold:*

Correctness: If parties follow the protocol, then $\mathrm{R}(z, m, w) = 1$;

Binding: With high probability over R's randomness, there does not exist a (m', w') with $m' \neq m$ such that $\mathrm{R}(z, m', w') = 1$;

Hiding: For all $m_0 \neq m_1$, $\{\mathrm{Com}(m_0; r)\}_r \approx_c \{\mathrm{Com}(m_1; r)\}_r$.

The Hiding Game: The computational hiding property above is often formalized as a game between a challenger $\mathcal{C}$ and adversary $\mathcal{A}$, where $\mathcal{A}$ presents $\mathcal{C}$ with two messages, $\mathcal{C}$ commits to one of them and $\mathcal{A}$ tries to guess which one, winning if it guesses correctly. A more formal description is given in Figure 1.1. The equivalence of these notions is easy to see: an adversary who can win the game with probability noticeably better than $1/2$ can be used to break the hiding property in Definition 1.1. The upside, as we will see later on, is that the game based interpretation is very flexible and easy to work with. During the proof of our main result we will get a lot of milage out of considering different versions of the game, tailored for specific reductions.

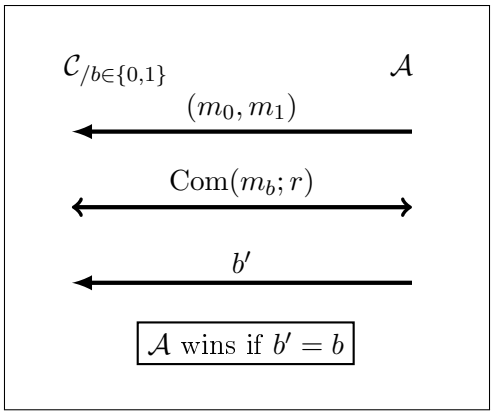

Figure 1.1: Hiding Game

2-Round, Statistically Binding Commitment from PRG: Naor's two-round, statistically binding bit commitment scheme [Nao91] is shown in Figure 1.2. This will be both a major building block in our non-malleable commitment scheme, and one of two focal examples for us throughout this thesis. Among the appealing features of this particular scheme are its simplicity and its reliance on the minimal assumption that one way functions exist.

Setup: Let $G : \{0,1\}^\lambda \to \{0,1\}^{3\lambda}$ be a length tripling PRG.

Commiter's Private Input: A bit $m \in \{0,1\}$ to commit to.

Commit Phase:

- R $\to$ C (Initialization Message): sample and send a random $\sigma \in \{0,1\}^{3\lambda}$.
- C $\to$ R (Commit Message): sample random $s \in \{0,1\}^\lambda$ and send $z = G(s) \oplus m\sigma$.

Decommit Phase:

- C $\to$ R (Decommit Message): send (m, s).
- (Verification): R checks the correctness of the commit message.

Figure 1.2: Naor's Statistically Binding Bit Commitment.

Binding: For z to be both a commitment to 0 and 1, there must exist $s, s' \in \{0,1\}^\lambda$ such that $G(s) \oplus G(s') = \sigma$. However, the probability that a random $3\lambda-$bit string is the XOR of two values in the image of G is at most $2^{2\lambda}/2^{3\lambda} = \mathbf{negl}(\lambda)$.

Hiding: The pseudorandomness of G ensures that z is indistinguishable from random, regardless of m.

One can use Naor's bit commitment scheme to commit to a string, bit by bit. Moreover, the same initialization message can be used across all instantiations.

Non-Interactive, Perfectly Binding Commitment from DDH: The Elgamal commitment scheme [Elg85] is shown in Figure 1.3. This scheme will serve as the main building block in our DDH-based instantiation. Elgamal commitment is perfectly binding, requires only one round of communication, and its algebraic nature makes it highly compatible with certain types of interactive proof systems called sigma protocols.

Setup: Let G be a DDH group with random public generators $g, h \in G$, but where x such that $h = g^x$ is secret. Let q be prime such that $q = |G|$, so $\mathbb{Z}_q$ is identified with the exponent group of G.

Commiter's Private Input: A value $m \in \mathbb{Z}_q$ to commit to.

Commit Phase:

- C $\rightarrow$ R (Commit Message): sample random $r \in \mathbb{Z}_q$ and send $(a, b) = (g^r, h^r g^m)$.

Decommit Phase:

- C $\rightarrow$ R (Decommit Message): send (m, r).
- (Verification): R checks the correctness of the commit message.

Figure 1.3: Elgamal Commitment.

Binding: Note that $(m, r) \mapsto (g^r, h^r g^m)$ is a bijection between $\mathbb{Z}_q^2$ and G^2. It follows that each (a, b) corresponds to exactly one pair (m, r).

Hiding: The DDH assumption ensures that $(g, h, a, b) = (g, g^x, g^r, g^{xr}) \cdot (1, 1, 1, g^m)$ is indistinguishable from random regardless of m.

1.4 Zero-Knowledge: "Proofs that Yield Nothing but their Validity"

Throughout this paper we make frequent use of zero-knowledge proofs. Our non-malleable commitment scheme will use them, and we will spend an entire chapter looking at a special type of zero-knowledge. In this section we introduce zero-knowledge, and several related notions.

An *interactive proof system* for a language L is a protocol, $\langle P, V \rangle$ in which a prover P tries to prove to a verifier V that an instance x is in L. After the protocol is complete V outputs either 1 or 0, indicating that it either accepts or rejects P's proof. Furthermore, two properties are required:

Completeness: If $x \in L$ and both parties follow the protocol then V accepts.

Soundness: If $x \notin L$ then even a cheating P^* who deviates arbitrarily from the protocol cannot cause V to accept, except with negligible probability.

We say that $\langle P, V \rangle$ is an *interactive argument* if soundness holds only against a computationally bounded P^*. In their groundbreaking work, Goldwasser, Micali and Rackoff [GMR89] put forth the notion of *zero-knowledge*, defined below. Informally, an interactive proof system is zero-knowledge if, after the protocol is complete, a dishonest verifier V^* has learned nothing except the validity of the statement $x \in L$. In particular, V learns nothing about

any witness to the truth of $x \in L$ that P might be holding. This is formalized by requiring the existence of a simulator who gets as input x and a cheating verifier V^*, and outputs a string that is indistinguishable from V^*'s view in the real interaction.

Definition 1.2 (Zero-Knowledge). *Let $\langle P, V \rangle$ be an interactive proof or argument system for a language $L \in \mathcal{NP}$. Fix any $x \in L$ with witness w, known to P. Let $\mathbf{VIEW}_{\langle P(w), V \rangle(x)}$ be the random variable denoting V's view after completing $\langle P, V \rangle$ on common input x and secret input w to P. We say that $\langle P, V \rangle$ is* zero-knowledge *if there exists a simulator $\mathcal{S}$ which on input x and a cheating verifier V^*, outputs a random variable $\mathbf{VIEW}_{\mathcal{S}(V^*, x)}$ such that*

$$\left\{\mathbf{VIEW}_{\langle P(w), V^* \rangle(x)}\right\} \approx_c \left\{\mathbf{VIEW}_{\mathcal{S}(V^*, x)}\right\}.$$

It is known that zero knowledge proofs for all languages $L \in \mathcal{NP}$ exist if and only if OWFs exist [GMW91, OW93].

We will also use the related but weaker notion of witness indistinguishability. We say that an interactive proof or argument system $\langle P, V \rangle$ is *witness indistinguishable* if for all $x \in L$ with witnesses w and w',

$$\left\{\mathbf{VIEW}_{\langle P(w), V^* \rangle(x)}\right\} \approx_c \left\{\mathbf{VIEW}_{\langle P(w'), V^* \rangle(x)}\right\}.$$

Finally, we say that $\langle P, V \rangle$ is a proof or argument *of knowledge* if P proves not just that $x \in L$ but that it knows some witness w to this fact. This is formalized by demanding that there exists an extractor E who takes as input a complete transcript of an accepting proof and the prover P, and outputs a witness to the fact $x \in L$.

Chapter 2

An Algebraic Approach to Non-Malleability

2.1 Introduction

The notion of non-malleability is central in cryptographic protocol design. Its objective is to protect against a man-in-the-middle (MIM) attacker that has the power to intercept messages and transform them in order to harm the security in other instantiations of the protocol. Commitment is often used as the paragon example for non-malleable primitives because of its ability to almost "universally" secure higher-level protocols against MIM attacks.

2.1.1 Non-Malleable Commitments

Commitments allow one party, called the committer, to probabilistically map a message m into a string, $\mathrm{Com}(m;r)$, which can be then sent to another party, called the receiver. In the statistically binding variant, the string $\mathrm{Com}(m;r)$ should be *binding*, in that it cannot be later "opened" into a message $m' \neq m$. It should also be *hiding*, meaning that for any pair of messages, m, m', the distributions $\mathrm{Com}(m;r)$ and $\mathrm{Com}(m';r')$ are computationally indistinguishable. The hiding property is formalized via a hiding "game" in which the adversary wins if he can distinguish between $\mathrm{Com}(m;r)$ and $\mathrm{Com}(m';r')$ with non-negligible

advantage.

A commitment scheme is said to be *non-malleable* if for every message m, no MIM adversary, intercepting a commitment $\mathrm{Com}(m;r)$ and modifying it at will, is able to efficiently generate a commitment $\mathrm{Com}(\tilde{m};\tilde{r})$ to a related message $\tilde{m}$. Interest in non-malleable commitments is motivated both by the central role that they play in securing cryptographic protocols under composition (see for example [CLOS02, LPV09]) and by the unfortunate reality that many widely used commitment schemes are actually highly *malleable*. Indeed, man-in-the-middle (MIM) attacks occur quite naturally when multiple concurrent executions of protocols are allowed, and can be quite devastating. Beyond protocol composition, non-malleable commitments are known to be applicable in secure multi-party computation [KOS03, Wee10, Goy11], authentication [NSS06], as well as a host of other non-malleable primitives (e.g., coin flipping, zero-knowledge, etc.), and even into applications as diverse as position based cryptography [CGMO09].

2.1.2 Prior Work

Since their conceptualization by Dolev, Dwork and Naor [DDN91], non-malleable commitments have been studied extensively, and with increasing success in terms of characterizing their round-efficiency and the underlying assumptions required. By now, we know how to construct constant-round non-malleable commitments based on any one-way function, and moreover the constructions are fully black-box. While this might give the impression that non-malleable commitments are well understood, each of the currently known constructions leaves something to be desired.

The first construction, due to DDN is perhaps the simplest and most efficient, mainly because it can in principle be instantiated with highly efficient cryptographic "sub-protocols". This, however, comes at the cost of round-complexity that is logarithmic in the maximum overall number of possible committers. Subsequent works, due to Barak [Bar02],

Pass [Pas04b], and, Pass and Rosen [PR05b] are constant-round, but rely on (highly inefficient) non-black box techniques. Wee [Wee10] (relying on [PW10]) gives a constant-round black-box construction under the assumption that sub-exponentially hard one-way functions exist. This construction employs a generic (and costly) transformation that is designed to handle general "non-synchronizing" MIM adversaries that do not schedule the messages in the different protocol executions synchronously.

Finally, recent works by Lin and Pass [LP11] and Goyal [Goy11] attain non-malleable commitment with constant round-complexity via the minimal assumption that polynomial-time hard to invert one-way functions exist. The Lin-Pass protocol makes highly non-black-box use of the underlying one-way function (though not of the adversary), along with a concept called signature chains; resulting in significant overhead. Goyal's protocol, using a later result of Goyal, Lee, Ostrovsky and Visconti [GLOV12], can be made fully black-box, with its only shortcomings being high-communication complexity and the use of the Wee transformation (or alternatively a similarly costly transformation due to Goyal [Goy11]) for handling non-synchronizing adversaries.

The current state of affairs is such that in spite of all the remarkable advances, the DDN construction and its analysis remain the simplest and arguably most appealing candidate for non-malleable commitments. This is both due to its black-boxness and because it does not require transformations for handling a non-synchronizing MIM (in fact, the protocol is purposefully designed to introduce asynchronicity in message scheduling, which can be then exploited in the analysis).

2.1.3 Our Results

In this work we introduce a new algebraic technique for obtaining non-malleability, resulting in a simple and elegant non-malleable commitment scheme.

The scheme's analysis contains many fundamentally new ideas allowing us to overcome

substantial obstacles without sacrificing efficiency. The protocol is constructed using any statistically binding commitment scheme as a building block, and hence requires the minimal assumption that one way functions exist.

Theorem. *Suppose the existence of 2-round statistically binding commitments (which is true if and only if one-way functions exist). Then there is a 4-round non-malleable commitment scheme.*

Our protocol enjoys the following appealing features, each of which makes it preferable in at least one way over any of the previously proposed protocols for non malleable commitment:

Simplicity. Compared to all previous protocols, ours is significantly simpler to describe and to instantiate (though not to analyze). The simplicity of the protocol also means that there is no need to introduce costly transformations for handling non-synchronizing adversaries.

Efficiency. In particular, ours is significantly more efficient than all prior protocols: it has only four communication rounds and makes use of a surprisingly small number of sub-protocols, each of which can be instantiated in a very efficient way (e.g. using standard sigma protocols).

Assumption. The assumption underlying our protocol is the existence of one-way functions, which is necessary for non-malleable commitments.

A direct consequence of our non-malleable protocol is a 4-round non-malleable zero-knowledge argument based on any one way function. This demonstrates that for zero-knowledge, non-malleability does not necessarily come at the cost of extra rounds of interaction or complexity assumption.

Theorem. *Suppose the existence of 2-round statistically binding commitments (which is true if an only if one-way functions exist). Then there is a 4-round black-box non-malleable zero-knowledge argument for every language in NP.*

Beyond the above virtues, we believe that our new techniques are actually the most significant contributions of this work. In addition to our use of algebra, we make novel combinatorial use of error correcting codes in order to ensure that different committers' tags differ in many coordinates (more on that later on). Whereas prior work relied on "worst-case" analysis of differences in committers' tags, ours follows from an "average-case" claim.

One way of viewing our construction is as a method for combining n atomic sub-protocols in a way that simultaneously amplifies their soundness and non-malleability properties, thus requiring much weaker soundness and non-malleability to begin with. We hope that this paradigm will become the norm for future work on in the area as, despite requiring more careful and strenuous analysis, it leads to pleasantly lightweight protocols. For example, this technique alone allows for an immediate linear reduction in communication complexity compared with its nearest relative, Goyal's protocol.

Another immediate payoff of the use of error correcting codes is that it allows for parallelizing the "two slots" technique of [Pas04b, PR05b]. With two possible exceptions, the so called two slot trick has become almost standard fare (in constant-round non-malleable protocols) as it creates a way to turn an asymmetry between different protocol instantiations that the MIM is involved in into two: one which is heavy on the right and one on the left. Running the two slots in parallel introduces several technical problems, most notably "if the two imbalances are side by side, won't they just cancel each other out?" Our analysis uses a cryptographic version of the "linear independence of polynomial evaluation" mantra in order to argue that the MIM cannot combine the two imbalances and must deal with each one separately.

We stress that the use of algebra and error correcting codes does not yield such reward

for free: the analysis required becomes substantially more difficult. In the next section we describe and briefly discuss our new protocol and extractor. We then outline our techniques, keeping it informal but pointing out several of the challenges faced and new ideas required to overcome them.

2.1.4 The New Protocol

Suppose that committer C wishes to commit to message m, and let $t_1, \ldots, t_n$ be a sequence of tags that uniquely correspond to C's identity. We assume that the receiver R is familiar with $t_1, \ldots, t_n$ (in particular the protocol's instructions depend on these tags). Let $\mathrm{Com}(m) = \mathrm{Com}(m; s)$ be a statistically binding commitment scheme (where s denotes the randomness used in the commitment), and suppose that $m \in \mathbb{F}_q$ where $q > \max_i 2^{t_i}$. The protocol $\langle \mathrm{C}, \mathrm{R} \rangle$ proceeds as follows:

1. C chooses random $\mathbf{r} = (r_1, \ldots, r_n) \in \mathbb{F}_q^n$ and sends $\mathrm{Com}(m)$ and $\{\mathrm{Com}(r_i)\}_{i=1}^n$ to R;
2. R chooses a vector $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)$ where each α_i is randomly chosen from $\left[2^{t_i}\right] \subset \mathbb{F}_q$;
3. C responds with $\mathbf{a} = (a_1, \ldots, a_n)$ where $a_i = r_i \alpha_i + m$;
4. C proves in ZK that $\mathbf{a}$ (from step 3) are consistent with m and $\mathbf{r}$ (from step 1).

The statistical binding property of the protocol follows directly from the binding property of the underlying commitment Com. The hiding property follows from the hiding of Com, the zero-knowledge property of the protocol used in step 4, and from the fact that for every i the receiver R observes only a single pair of the form (α_i, a_i), where $a_i = r_i \alpha_i + m$.

Note the role of C's tags in the protocol: t_i determines the size of the $i-$th coordinate's challenge space. Historically, non-malleable commitment schemes have used the tags as a way for the committer to encode its identity into the protocol as a mechanism to prevent M (whose tag is different from C's) from "mauling" C's commitment into its own. In our

protocol the tags play the same role, albeit rather passively. For example, though the size of the $i-$th challenge space depends on t_i, the size of the total challenge space depends only on the sum $\sum_{i=1}^{n} t_i$ of the tags. In particular, our scheme leaves open the possibility that the left and right challenge spaces might have the same size (as usual, we think of M as taking part in a left and right interaction with C and R respectively). This raises a red flag, as previous works go to great lengths to set up imbalances between the left and right challenge spaces in order to force M to "give more information than it gets". Nevertheless, we are able to prove that any mauling attack will fail.

At a very high level, our protocol can be seen as an algebraic abstraction of Goyal's protocol. However, the fundamental difference we should emphasize from [Goy11] is that he crucially relies on the challenge space in the left interaction being much smaller than the challenge space in the right. For us, the challenge spaces in the two interactions are exactly the same size and so the techniques of [Goy11] do not apply to our setting; at least at first. Our protocol does have small imbalances between the challenge spaces of individual coordinates, which is what we will eventually use to prove non-malleability. However, proving that the coordinates are sufficiently independent so that these imbalances accrue to something usable is completely new to this work.

2.1.5 Proving Non-Malleability

Consider a MIM adversary M that is playing the role of the receiver in a protocol using tags $t_1, \ldots, t_n$ while playing the role of the committer in a protocol using tags $\tilde{t}_1, \ldots, \tilde{t}_n$ (we describe explicitly how to construct the tags from C's identity in Section **??**). We refer to the former as the "left" interaction and to the latter as the "right" interaction. We let m and $\tilde{m}$ denote the messages committed to in the left and right interactions respectively. One nice feature of our protocol is that it is automatically secure against a non-synchronizing adversary, simply because there are so few rounds, there is no way for the MIM to benefit

by changing the message order: any scheduling but the synchronous one can be dealt with trivially. So the only scheduling our proof actually needs to handle is a synchronizing one, as depicted in Figure 2.1 below.

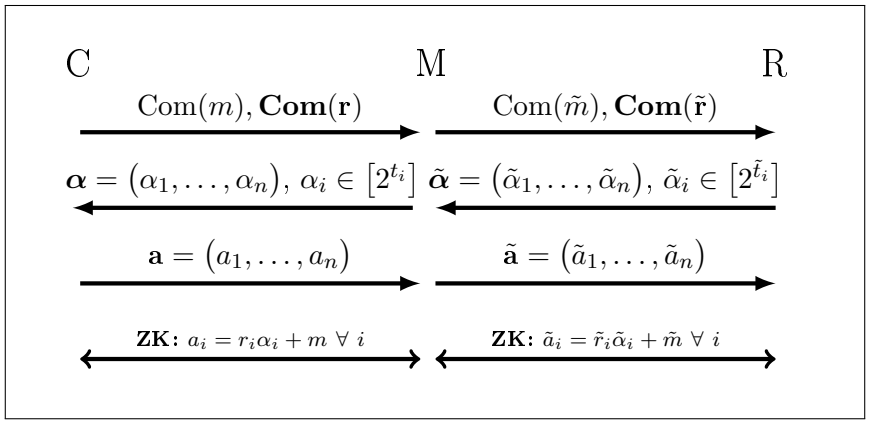

Figure 2.1: Protocol with Man-in-the-Middle

As is customary in proofs of non-malleability, we demonstrate the existence of an extractor, E, who is able to rewind M and extract $\tilde{m}$ without needing to rewind C in the left instantiation. Our extractor is modeled after Goyal's extractor which: (1) rewinds M to where $\tilde{\boldsymbol{\alpha}}$ was sent and asks a new query $\tilde{\boldsymbol{\beta}}$ instead, and (2) responds to M's left query randomly (it cannot do better without rewinding C as it does not know m), hoping that M answers correctly on the right.

In Goyal's protocol there is no way for E to know whether M answered correctly or not, and so it must have a verification message after the query response phase so E can compare M's answer with the main thread to verify correctness. We sidestep this necessity in the following way. We rewind to the beginning of step 2 twice and ask two new query vectors $\tilde{\boldsymbol{\beta}}$ and $\tilde{\boldsymbol{\gamma}}$, we answer randomly on the left obtaining $\{(\tilde{\boldsymbol{\alpha}}, \tilde{\mathbf{a}}), (\tilde{\boldsymbol{\beta}}, \tilde{\mathbf{b}}), (\tilde{\boldsymbol{\gamma}}, \tilde{\mathbf{c}})\}$, where $(\tilde{\boldsymbol{\alpha}}, \tilde{\mathbf{a}})$ is from the main thread. Comparing both $(\tilde{\beta}_i, b_i)$ and $(\tilde{\gamma}_i, c_i)$ with $(\tilde{\alpha}_i, a_i)$ will result in candidate values $\tilde{m}_i$ and $\tilde{m}'_i$, but with no verification message it is not clear how E should verify which one (if either) is correct.

We accomplish this with the following "collinearity test". If $\tilde{m}_i = \tilde{m}'_i$ then E checks whether the points $\{(\tilde{\alpha}_i, \tilde{a}_i), (\tilde{\beta}_i, \tilde{b}_i), (\tilde{\gamma}_i, \tilde{c}_i)\}$ are collinear. If so, E deems that $\tilde{m}_i$ was the correct value. This requires proving that M cannot answer "incorrectly but collinearly".

Tags in Error Corrected Form. As in previous non-malleable commitment schemes, our protocol consists of n "atomic subprotocols", one for each tag. Previous non-malleable commitment schemes use the so called DDN trick [DDN91] in order to turn C's $k-$bit identity into a list of n $(= k)$ tags $t_1, \ldots, t_n$, satisfying the properties: (1) each t_i is of length $\log n + 1$; and (2) if $\{t_i\}_i$ and $\{\tilde{t}_j\}_j$ are the tags resulting from two distinct identities then there exists some i such that t_i is completely distinct from $\{\tilde{t}_j\}_j$, meaning that $t_i \neq \tilde{t}_j$ for all j.

Previous schemes' security proofs require the extractor to be able to use any completely distinct left subprotocol (i.e., one whose tag is completely distinct from $\{\tilde{t}_j\}_j$) to extract M's commitment $\tilde{m}$ with high probability. This ensures that extraction is possible even in the worst case when there is a single such subprotocol. It also introduces a good deal of redundancy into the protocol.

While one would expect most pairs of distinct identities to result in pairs of tags such that property (2) holds for many i, all the DDN trick can guarantee in the worst case is that it holds for a single i. If however, one first applied an error correcting code to C's identity obtaining, say, a codeword in $\mathbb{F}_q^n$ for suitably chosen $q \leq \text{poly}(n)$, then applying the DDN trick to this codeword would yield tags such that (1) t_i is of length $\mathcal{O}(\log n)$; and (2) t_i is completely distinct from $\{\tilde{t}_j\}_j$ for a constant fraction of the $i \in \{1, \ldots, n\}$.

Our "completely distinct on average" property, on the other hand, requires only that extraction is possible from a completely distinct left subprotocol with constant probability. This allows us to remove much of the artificial redundancy resulting in an incredibly trim protocol.

Non-malleability against a copying M. To get a sense of why we might expect our scheme to be non-malleable, let us examine the situation against an M who attempts to maul C's commitment by simply copying its messages from the left interaction to the right. Let m be the message committed to on the left and let $\{t_i\}_{i=1}^n$ and $\{\tilde{t}_i\}_{i=1}^n$ be the corresponding tags.

After the first message, M will have copied C's commitments over to the right interaction, successfully committing to the coefficients of the linear polynomials $\tilde{f}_i(x) = r_i x + m$, $i = 1, \ldots, n$. The hiding of Com ensures it does not know the polynomials themselves, and so when it receives the right query vector $\tilde{\boldsymbol{\alpha}}$, its only hope of coming up with the correct valuations $\tilde{f}_i(\tilde{\alpha}_i)$ is to copy R's challenge to the left interaction and copy C's response back. However, it is unlikely that this will be possible. Indeed, M can only copy $\tilde{\alpha}_i$ over to the left when $\tilde{\alpha}_i \in \left[2^{t_i}\right]$. If $\tilde{t}_i > t_i$ then the $i-$th challenge space on the right is at least twice as big as the $i-$th challenge space on the left, which means that the probability $\tilde{\alpha}_i$ can be copied is at most $1/2$. We will use a code which ensures that $\tilde{t}_i > t_i$ for a constant fraction of the i, and makes the probability that M can copy every coordinate of R's query vector $\tilde{\boldsymbol{\alpha}}$ negligible. So M will not be able to successfully answer R's query and complete the proof when performing the "copying" attack.

Non-malleability against general M. Establishing security against a general man-in-the-middle adversary is significantly more challenging, and this is where the bulk of the new ideas are required. Our proof of non-malleability will require us to delve into the full range of possibilities for M's behavior. In each case, we will show that one of three things happen:

1. M does not correctly answer its queries with good enough probability;

2. E succeeds in extracting $\tilde{m}$ with sufficient probability;

3. an M with such behavior can be used to break the hiding of Com.

The core of our result can be seen as a reduction from a PPT M who correctly answers its queries with non-negligible probability and yet causes E to fail with overwhelming probability to a machine $\mathcal{A}$ who breaks the hiding of Com. The following is a very high level outline of our proof.

We define USEFUL to be the set of transcripts which do not lead to situation 1 above; that is, transcripts for which M has a good chance of completing the protocol given the prefix and challenges. This is important in order for E to have any chance of successfully extracting $\tilde{m}$. Indeed, if M just aborts in every rewind, E will have no chance. From this standpoint, USEFUL is the set of transcripts which give E "something to work with." We prove that most transcripts are in USEFUL in Claim 2.3.

We then define EXT, the set of "extractable" transcripts, on which E will succeed with high probability. These are the transcripts which lead to situation 2. Intuitively, EXT is the set of transcripts such that M has good probability of correctly answering a query in a rewind despite the fact that E provides random answers to M's queries. We prove that indeed, if a transcript is in EXT then E succeeds in extracting $\tilde{m}$.

Finally, we define INT, the set of "interesting" transcripts which are both useful and not extractable. Transcripts in INT are troublesome as on the one hand, usefulness ensures that the prefix is such that if (in a rewind) M receives correct responses to its queries on the left, it gives correct responses (with non-negligible probability) to the queries on the right. At the same time however, transcripts in INT are not extractable and so the prefix is also such that if M receives random responses to its queries on the left it answers the right queries incorrectly. Certainly, the hiding of Com ensures that M cannot *know* whether it receives correct or random responses to its queries on the left. So this difference in behavior suggests that we may be able to use M to violate the hiding of Com, leading to situation 3 above.

For this, the most difficult part of our proof, we use the hiding of our protocol $\langle \mathrm{C}, \mathrm{R} \rangle$. We will be able to use an M with the above "unlikely seeming" behavior in order to construct

an adversary that breaks the hiding of $\langle C, R \rangle$. Our framework simultaneously simplifies and generalizes [Goy11]'s techniques and gives us a desirable level of flexibility in how we apply them, allowing us to overcome the substantial challenges which arise in our setting.

2.2 Non-Malleable Commitments

Non-malleability in cryptography refers to security against an adversary who participates in two or more protocols and tries to use information obtained in one protocol execution to influence another. The classic example is an adversary invoking two instantiations of a commitment scheme, playing one as the receiver, the other as the committer, and trying to use the commitment it receives in order to generate a commitment to a related value in the other execution. Such an adversary is generally referred to as a *man-in-the-middle* (MIM), and such an attack is called a *mauling* attack. At first glance, non-malleability seems impossible as surely nothing can be done to protect against a MIM who simply copies messages from one protocol execution to another. For this reason, non-malleable security offers protection only against any MIM who tries to change messages in a meaningful way.

In this work, just as in [DDN91, PR05b], we will assume that players have identities. In order to perform a successful mauling attack, a MIM has to maul a commitment corresponding to C's identity into a commitment of his own, distinct identity. Though this sounds like a strong assumption on the network, essentially requiring that "you know who you are talking to", for our purposes, it is actually equivalent to the requirement discussed above, that the MIM do something other than simply copy messages. This is because our protocol is interactive, and the first committer message contains a statistically binding commitment to m. This means that if we set the committer's identity to be the first committer message, C's and M's identities will be distinct unless M copied C's first message. Moving forward, we assume that players have identities and we require that non-malleability holds only in the

case when C and M's identities are different. We also assume for simplicity that player identities are known before the protocol begins, though strictly speaking this is not necessary, as the identities do not appear in the protocol until after the first committer message.

2.2.1 Definition of Non-Malleable Commitments

We wish for our commitment scheme to be impervious to a MIM adversary, M, who takes part in two protocol executions (in the left interaction M acts as the receiver while in the right, M plays the role of the committer), and tries to use the left interaction to affect the right. The security property we desire can be summarized:

> *For any MIM adversary* M*, there exists a standalone machine who plays only one execution as the committer, yet whose commitment is indistinguishable from* M*'s commitment on the right.*

We follow Lin *et al.* [LPV08] who formalize the above nicely using the real/ideal paradigm. In the real world, M interacts normally with C (who commits to an unknown message m) on the left, and with R on the right. Let $\mathbf{MIM}_{\langle \mathrm{C}(m),\mathrm{R}\rangle}$ be a random variable that describes the pair $(\tilde{m}, v)$ consisting of M's commitment in the right interaction and M's view after the commit phase of both executions is complete. The randomness is over C and R's random coins. Note that if M performs a mauling attack, it might not be possible to efficiently learn $\tilde{m}$ from v. In the ideal world, M interacts with a simulator $\mathcal{S}$ who plays the role of both the committer on the left and the receiver on the right. Let $\mathbf{MIM}_{\langle \mathcal{S},\mathcal{S}\rangle}$ be the random variable denoting M's right commitment and view after the commit phases are complete in the ideal world.

Definition 2.1 (**Non-Malleable Commitments**). *A commitment scheme* $\langle \mathrm{C}, \mathrm{R}\rangle$ *is* non-

malleable *if for every PPT MIM adversary* M*, there exists a PPT simulator* $\mathcal{S}$ *such that*

$$\{\mathbf{MIM}_{\langle \mathrm{C}(m),\mathrm{R}\rangle}\}_m \approx_c \{\mathbf{MIM}_{\langle \mathcal{S},\mathcal{S}\rangle}\}.$$

We wish the ideal world to model the case when M is a standalone machine taking part only in the right execution, and so our $\mathcal{S}$ will play honestly as R on the right, and act as a "dummy committer" on the left committing honestly to 0.

Concurrent Non-Malleability: One can extend non-malleability to the concurrent setting where M may participate in polynomially many left and right executions which may be interleaved arbitrarily. Concurrent non-malleability, also known as many-many non-malleability is defined below. To state the definition, we update the random variables $\mathbf{MIM}_{\langle \mathrm{C}(m),\mathrm{R}\rangle}$ and $\mathbf{MIM}_{\langle \mathcal{S},\mathcal{S}\rangle}$ appropriately.

Let $\langle \mathrm{C}, \mathrm{R}\rangle$ be a commitment scheme and fix polynomials $\ell = \ell(\lambda)$ and $\tilde{\ell} = \tilde{\ell}(\lambda)$. Consider a MIM adversary M who participates in ℓ (resp. $\tilde{\ell}$) left (resp. right) executions of $\langle \mathrm{C}, \mathrm{R}\rangle$ where it acts as the committer (resp. receiver). Let $\mathbf{m} = (m_1, \ldots, m_\ell)$ and $\tilde{\mathbf{m}} = (\tilde{m}_1, \ldots, \tilde{m}_{\tilde{\ell}})$ be the values committed to in the left and right executions. Let $\mathbf{MIM}^{\ell,\tilde{\ell}}_{\langle \mathrm{C}(\mathbf{m}),\mathrm{R}\rangle}$ be the random variable describing $(\tilde{\mathbf{m}}, v)$, M's commitments in the right executions and view upon completion of the commit phases of all executions. Similarly, for a simulator $\mathcal{S}$ who acts as C (resp. R) in all ℓ (resp. $\tilde{\ell}$) left (resp. right) executions, let $\mathbf{MIM}^{\ell,\tilde{\ell}}_{\langle \mathcal{S},\mathcal{S}\rangle}$ be the random variable describing $(\tilde{\mathbf{m}}, v)$, M's right commitments and view after the commit phases are complete.

Definition 2.2 (**Many-Many Non-Malleable Commitments**). *A commitment scheme* $\langle \mathrm{C}, \mathrm{R}\rangle$ *is* many-many non-malleable *if for every PPT MIM adversary* M*, there exists a PPT simulator* $\mathcal{S}$ *such that*

$$\{\mathbf{MIM}^{\ell,\tilde{\ell}}_{\langle \mathrm{C}(\boldsymbol{m}),\mathrm{R}\rangle}\}_{\boldsymbol{m}} \approx_c \{\mathbf{MIM}^{\ell,\tilde{\ell}}_{\langle \mathcal{S},\mathcal{S}\rangle}\}.$$

Additionally, we say that $\langle \mathrm{C}, \mathrm{R} \rangle$ *is* one-many *or* many-one *non-malleable if* $\ell = 1$ *or* $\tilde{\ell} = 1$.

Non-Malleability wrt **INSERT-WORD-HERE:** The numerous applications that non-malleable commitments and related primitives have found across cryptography have motivated several variants of the definition. The one given above is often refered to in the literature as *non-malleability with respect to commitment*. This is because any MIM adversary who performs a mauling attack will not be able to successfully complete the commit phase. An alternative weaker notion which appears occasionally in the literature is *non-malleability with respect to decommitment* which stipulates that a MIM adversary performing a mauling attack not be able to complete the decommit phase (but might manage to finish the commit phase successfully). Definitionally, the only differece between the two is that the view component of the random variables $\mathbf{MIM}_{\langle \mathrm{C}(m), \mathrm{R} \rangle}$ and $\mathbf{MIM}_{\langle \mathcal{S}, \mathcal{S} \rangle}$ refers to M's view after the decommit phases of both executions are complete.

Another alternative definition given in [Goy11] is *non-malleability with respect to replacement*. Though weaker than non-malleability with respect to commitment, it is sufficient for applications to MPC. Essentially the difference is that non-malleability with respect to replacement limits the adversary's power to abort, forcing him instead to *replace* the abort with a legitamite commitment to some value.

In this work we consider only non-malleability with respect to commitment; the version we have already defined. It is the most standard notion and has the richest history.

2.2.2 Malleability of Aforementioned Schemes

In order to show the fundamental importance of non-malleable security, we describe mauling attacks on the two commitment schemes we have already introduced, namely Naor's bit commitment and Elgamal commitment. In fact, non-malleability is somewhat of a rare bird

among enhanced security properties in the sense that if a scheme is not non-malleable it is often highly *malleable*, admitting devestating attacks to a MIM adversary.

Mauling Naor's Bit Commitment: Suppose M participates in two executions of Naor's bit commitment (described in Section 1.3), both instantiated with the same length tripling PRG G. Then M can maul a left commitment to m to a right commitment of $1-m$ as follows:

1. Upon receiving right initialization message $\tilde{\sigma}$, M sets $\sigma = \tilde{\sigma}$ and sends it to C;

2. Upon receiving left commitment message z, M sets $\tilde{z} = z \oplus \tilde{\sigma}$ and sends it to R.

If z is a commitment to 0 then $z = G(s)$ and so $\tilde{z} = G(s) \oplus \tilde{\sigma}$ is a commitment to 1. Similarly, if z is a commitment to 1 then $z = G(s) \oplus \sigma$ and so $\tilde{z} = G(s)$ is a commitment to 0.

Mauling Elgamal Commitment: Now suppose that M participates in two Elgamal commitments (described in Section 1.3), both instantiated with the same public parameters (G, g, h). Then M can maul a left commitment to m to a right commitment of $2m$ by simply squaring the left message. Indeed, if $(a, b) = (g^r, h^r g^m)$ is a commitment to m, then $(\tilde{a}, \tilde{b}) = (a^2, b^2) = (g^{2r}, h^{2r} g^{2m})$ is a commitment to $2m$.

2.3 A New Non-Malleable Commitment Scheme

2.3.1 Tags in Error Corrected Form

Let $x \in \{0,1\}^k$ be C's identity. All previous non-malleable commitment protocols use the so called "DDN trick" [DDN91] to generate tags $t_1, \ldots, t_n$ $(n = k)$ by setting $t_i = i \circ x_i = 2i + x_i$. Tags generated in this fashion satisfy the property:

- if $\{t_i\}_i$ and $\{\tilde{t}_j\}_j$ are tags resulting from distinct identities $x, \tilde{x} \in \{0,1\}^k$ then there exists some i such that $t_i \neq \tilde{t}_j$ for all j. In this case we say that t_i is *completely distinct* from $\{\tilde{t}_j\}_j$.

In this work we generate tags by first applying an error correcting code to x, obtaining codeword $\hat{x} \in \mathbb{F}^{n/2}$, for some finite field $\mathbb{F}$ and only then applying something like the DDN trick, obtaining tags $t_1, \ldots, t_n$ with

$$t_i = \begin{cases} 2i|\mathbb{F}| + \hat{x}_i, & i \leq n/2 \\ (2n+1)|\mathbb{F}| - t_{n-i}, & i > n/2 \end{cases}.$$

The properties we will need from our tags are listed below. Let $\{t_i\}_i$ and $\{\tilde{t}_j\}_j$ be the tags resulting from distinct $x, \tilde{x} \in \{0,1\}^k$.

0. Ordered: $t_1 < t_2 < \cdots < t_n$;

1. Well Spaced: $t_{i+1} - t_i = \omega(\log \lambda)$ for all $i \in [n]$;

2. Every Coordinate Matters: $\sum_j t_j - \sum_{j \neq i} \tilde{t}_j = \omega(\log \lambda)$ for all $i \in [n]$;

3. Completely Distinct on Average: if $i \neq j$ then $t_i \neq \tilde{t}_j$; moreover $t_i \neq \tilde{t}_i$ holds for a constant fraction of the $i \in [n]$;

4. Balanced: $t_i < \tilde{t}_i$ holds for a constant fraction of the $i \in [n]$.

Properties 0-2 follow immediately as long as $|\mathbb{F}| = \omega(\log \lambda)$. Property 3 follows as long as the error correcting code we choose has constant distance. Finally, property 4 holds because if $t_i \neq \tilde{t}_i$ then either $t_i < \tilde{t}_i$ or else $t_{n-i} < \tilde{t}_{n-i}$. This is analogous to the two slot trick of [Pas04b, PR05b].

It remains to select parameters. We make the conservative selection $n = \mathcal{O}(\lambda)$ and $q = \log^2(\lambda)$ to ensure both that the above properties hold and that all that is required of

the error correcting code is that it has constant distance and constant rate. Codes with such properties are known to exist. We could use, for example polynomial based codes such as Reed-Muller codes, the multivariate generalization of Reed-Solomon codes. This results in the overall communication complexity of our non-malleable commitment scheme being $\tilde{\mathcal{O}}(\lambda^2)$. Slightly better communication complexity might be available through more agressive choices of parameters or better codes. We do not press the issue further.

2.3.2 The Protocol

In this section, we describe our protocol. Let $t_1, \ldots, t_n$ be tags in error corrected form, as described in Section 2.3.1. Let Com denote Naor's commitment scheme. Let $\mathrm{Com}_\sigma(m)$ denote a commitment to the message m with initialization message σ. Whenever we need to be explicit about the randomness used to generate the commitment, we denote it as $\mathrm{Com}_\sigma(m; s)$ where s is the randomness. We use boldface to denote vectors; in particular a challenge vector $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)$ and a response vector $\mathbf{a} = (a_1, \ldots, a_n)$. We write **Com** for the entire commit message. The commitment scheme $\langle \mathrm{C}, \mathrm{R} \rangle$ appears in Figure 2.2.

Proposition 2.1. *The commitment scheme $\langle \mathrm{C}, \mathrm{R} \rangle$ is computationally hiding and statistically binding.*

Proof. Statistical binding follows immediately from the statistical binding of Naor's commitment scheme. To prove computational hiding, we consider the following hybrid games.

$\mathbf{G}_0$: This is the hiding game, described in Figure 1.1. An adversary $\mathcal{A}$ presents two messages $m_0, m_1 \in \mathbb{Z}_q$ to a challenger $\mathcal{C}$. $\mathcal{C}$ then chooses one of them, m_b at random and commits to it using $\langle \mathrm{C}, \mathrm{R} \rangle$. Specifically, upon receiving initialization message σ from $\mathcal{A}$, $\mathcal{C}$ chooses random $r_1, \ldots, r_n \in \mathbb{Z}_q$ and sends $\mathbf{Com} = \big(\mathrm{Com}_\sigma(m_b), \mathrm{Com}_\sigma(r_1), \ldots, \mathrm{Com}_\sigma(r_n)\big)$ to $\mathcal{A}$. $\mathcal{A}$ then sends random query vector $\boldsymbol{\alpha}$ and $\mathcal{C}$ returns response vector $\mathbf{a}$ where $a_i = r_i \alpha_i + m_b$. Finally, $\mathcal{C}$ proves in zero-knowledge that the commit message **Com** is

Public Parameters: Tags $t_1, \ldots, t_n$ and a large prime q such that $q > 2^{t_i}$ for all i.

Commiter's Private Input: Message $m \in \mathbb{F}_q$ to be committed to.

Commit Phase:

0. R $\to$ C **Initialization message:** Send the first message σ of the Naor commitment scheme.

1. C $\to$ R **Commit message:** Sample random $r_1, \ldots, r_n \in \mathbb{F}_q$.

   - Define linear functions $f_1, \ldots, f_n$ by $f_i(x) = r_i x + m$.
   - Send commitments $\mathbf{Com} = \big(\mathrm{Com}_\sigma(m), \mathrm{Com}_\sigma(r_1), \ldots, \mathrm{Com}_\sigma(r_n)\big)$.

2. R $\to$ C **Query:**

   - Send random challenge vector $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)$, $\alpha_i \in [2^{t_i}] \subset \mathbb{F}_q$.

3. C $\to$ R **Response:**

   - Send evaluation vector $\mathbf{a} = (a_1, \ldots, a_n)$, $a_i = f_i(\alpha_i)$.

4. C $\longleftrightarrow$ R **Consistency proof:** Parties engage in a zero-knowledge argument protocol where C proves to R that the commit message is well formed and that the response is correct.

Decommit Phase:

C $\to$ R **Decommit Message:** Send $(m; r_1, \ldots, r_n)$.

Verification: R checks the correctness of the commit and response messages.

Figure 2.2: Non-malleable commitment scheme $\langle \mathrm{C}, \mathrm{R} \rangle$.

well formed and that the response vector is correct. At this point, $\mathcal{A}$ outputs a guess b' and wins if $b' = b$. We must show that $\mathcal{A}$'s chances of winning are no greater than $1/2 + \mathbf{negl}(\lambda)$.

$\mathbf{G}_1$: This is the same as the above game except that $\mathcal{C}$ simulates the zero-knowledge proof at the end using its knowledge of $\mathcal{A}$. The zero-knowledge property ensures that

$$\Big|\Pr\big(\mathcal{A} \text{ wins } \mathbf{G}_1\big) - \Pr\big(\mathcal{A} \text{ wins } \mathbf{G}_0\big)\Big| = \mathbf{negl}(\lambda).$$

$\mathbf{G}_2$: This is the same as $\mathbf{G}_1$ except that $\mathcal{C}$ sends a random response vector $\mathbf{a}'$. To see that $\mathcal{A}$'s chance of winning this game is essentially the same as its chances of winning $\mathbf{G}_1$, suppose $\mathcal{C}$ acts as an adversary against another challenger $\mathcal{C}'$ in a different hiding game as follows. Upon receiving $(m_0, m_1; \sigma)$ from $\mathcal{A}$, $\mathcal{C}$ chooses random values $r_i, s_i \in \mathbb{F}_q$ for $i = 1, \ldots, n$ and sends $(r_i, s_i)_{i=1,\ldots,n}$ and σ to $\mathcal{C}'$ who returns either $z_i = \mathrm{Com}_\sigma(r_i)$ or $z_i = \mathrm{Com}_\sigma(s_i)$ for $i = 1, \ldots, n$, each with probability $1/2$. Now $\mathcal{C}$ returns $\mathrm{Com}_\sigma(m_b)$ for random $b \in \{0, 1\}$ and z_i for $i = 1, \ldots, n$. When $\mathcal{C}$ receives $\boldsymbol{\alpha}$, it returns $r_i\alpha_i + m_b$ for all i. Note that if $z_i = \mathrm{Com}_\sigma(r_i)$ then $\mathcal{A}$ is playing game $\mathbf{G}_1$, while if $z_i = \mathrm{Com}_\sigma(s_i)$ it is playing $\mathbf{G}_2$. Therefore, if $\mathcal{A}$'s chances of winning $\mathbf{G}_1$ and $\mathbf{G}_2$ differ by a noticeable amount, then $\mathcal{C}$ can win its own hiding game with $\mathcal{C}'$ with noticeable advantage over $1/2$. So we see that

$$\Big|\Pr\big(\mathcal{A} \text{ wins } \mathbf{G}_2\big) - \Pr\big(\mathcal{A} \text{ wins } \mathbf{G}_1\big)\Big| = \mathbf{negl}(\lambda).$$

Finally, the hiding of Com ensures that $\mathcal{A}$'s chances of winning $\mathbf{G}_2$ are $1/2 + \mathbf{negl}(\lambda)$. To see this, suppose again that $\mathcal{C}$ acts as adversary in its own hiding game against challenger $\mathcal{C}^*$ as follows. Upon receiving $(m_0, m_1; \sigma)$ from $\mathcal{A}$, $\mathcal{C}$ forwards it on to $\mathcal{C}^*$ and receives $z = \mathrm{Com}_\sigma(m_b)$ for some b. $\mathcal{C}$ then chooses random $r_i \in \mathbb{Z}_q$ for

$i = 1, \ldots, n$ and returns z along with $\mathrm{Com}_\sigma(r_i)$. When $\mathcal{A}$ sends $\boldsymbol{\alpha}$, $\mathcal{C}$ chooses random response vector $\mathbf{a}'$ and then simulates the zero-knowledge proof. Finally, $\mathcal{C}$ forwards $\mathcal{A}$'s choice b' to $\mathcal{C}^*$. As $\mathcal{C}$ wins its hiding game if and only if $\mathcal{A}$ wins $\mathbf{G}_2$, the hiding of Naor's bit commitment ensures that

$$\left|\Pr\big(\mathcal{A} \text{ wins } \mathbf{G}_2\big) - \frac{1}{2}\right| = \mathbf{negl}(\lambda),$$

thus proving the hiding of $\langle \mathrm{C}, \mathrm{R} \rangle$.

□

Theorem 2.1 (Main theorem)**.** *The commitment scheme* $\langle \mathrm{C}, \mathrm{R} \rangle$ *is non-malleable.*

2.4 Proof of Non-Malleability

In this section we prove most of Theorem 2.1. Specifically, we prove standalone non-malleability of $\langle \mathrm{C}, \mathrm{R} \rangle$ against a synchronizing adversary. This is a MIM who plays corresponding messages of the left and right sessions one after another. We will show how to handle non-synchronizing adversaries in the final, round-optimized version of our protocol in Section 2.6. We remark here only that the simplicity of our protocol allows one to handle any adversary except the synchronizing one essentially trivially.

Recall from Definition 2.1 that we must show for any MIM M, there exists a simulator $\mathcal{S}$ such that

$$\{\mathbf{MIM}_{\langle \mathrm{C}(m), \mathrm{R} \rangle}\}_m \approx_c \{\mathbf{MIM}_{\langle \mathcal{S}, \mathcal{S} \rangle}\},$$

where the distributions are over $(\tilde{m}, v)$: M's commitment in the right interaction and view after the commit phases of both executions are complete in the real and ideal worlds, respectively. As we have already mentioned, our simulator simply commits honestly to $0 \in \mathbb{Z}_q$

on the left and plays honestly as R on the right. We prove indistinguishability of the above distributions for any M by constructing an extractor E which takes M's view after the commit phases of the left and right executions are complete and outputs its commitment $\tilde{m}$ in the right execution whp. It follows that an algorithm which distinguishes $\mathbf{MIM}_{\langle \mathrm{C}(m),\mathrm{R}\rangle}$ from $\mathbf{MIM}_{\langle \mathcal{S},\mathcal{S}\rangle}$ can be used to break the hiding of $\langle \mathrm{C},\mathrm{R}\rangle$ in the following way: 1) let v be M's view after completing the commit phases of the left and right executions in either the real or ideal world; 2) use E to obtain the pair $(\tilde{m}, v)$; 3) use the distinguisher to determine whether M's interaction took place in the real or ideal world. This breaks the hiding of the left commitment as the only difference between the worlds is that in the real, C commits to m while in the ideal, $\mathcal{S}$ commits to 0.

2.4.1 The Extractor E

The high level description of our extractor (described formally in Figure 2.3) is quite simple. Intuitively, our protocol begins by C committing to n, threashold 2, Shamir secret sharings [Sha79] of m; R then asks for one random share from each sharing, which C gives. All E does is rewind M to the beginning of the right session's query phase ask for a new random share. Since E gets one share as part of its input, this will allow E to reconstruct $\tilde{m}$.

The problem with this approach is that E does not know the value C has commit to in the left interaction and so it does not know how to answer M's query on the left correctly. The best E can do is give M a random response on the left and hope that M will give a correct response on the right anyway. On the one hand, the hiding of Com dictates that M cannot distinguish a correct response from a random one. On the other hand, M doesn't actually need to know whether the response on the left is correct or not in order to perform a successful mauling attack. Imagine, for example, the MIM who mauls R's challenge to the left execution and mauls C's response back. Such an M will prevent E from extracting $\tilde{m}$

because M only correctly answers E's query if given a correct response to its own left query, which E cannot give. Of course we will prove that no M with such behavior can exist, but this proof is highly non-trivial.

Another question which our extractor raises is "how can E tell a correct response from an incorrect one?" As we have described it, the hiding of Com ensures that it cannot. However, a small modification to E fixes this. Instead of asking for one new share, E rewinds twice to the beginning of the right query phase and asks for two different new shares. The key observation is that if M answers both queries correctly then the three shares it holds (the two it receives plus the one it gets as input) are collinear, whereas if M answers at least one incorrectly they are overwhelmingly likely to NOT be collinear. This is the first appearance of a tangeable payoff of the algebraicity of our protocol. For example, the protocol of [Goy11] (which is similar to ours, but strictly combinatorial in nature) does not have this algebraic verification technique at its disposal and must introduce use extra rounds into the protocol to ensure its extractor can reconstruct $\tilde{m}$.

We now prepare to formally describe E. E is given as input a transcript of a complete commit phase in both the left and right interactions. We denote the transcript with the letter $\mathbb{T}$. Specifically,

$$\mathbb{T} = \left(\mathbf{Com}, \tilde{\mathbf{Com}}, \boldsymbol{\alpha}, \tilde{\boldsymbol{\alpha}}, \mathbf{a}, \tilde{\mathbf{a}}, \pi, \tilde{\pi}\right).$$

Since E will not be interested in the proofs $(\pi, \tilde{\pi})$, and since without loss of generality M is deterministic (which means that $\tilde{\mathbf{Com}}$, $\boldsymbol{\alpha}$, $\tilde{\mathbf{a}}$ are uniquely determined by $\mathbf{Com}, \tilde{\boldsymbol{\alpha}}$, and $\mathbf{a}$) we will often just write $\mathbb{T} = \left(\mathbf{Com}, \tilde{\boldsymbol{\alpha}}, \mathbf{a}\right)$.

Definition 2.3 (Accepting Transcript). *We say that* $\mathbb{T} \in \mathsf{ACC}$ *if* R *accepts* M*'s proof* $\tilde{\pi}$*.*

We say that M aborts if M behaves in such a way as to make $\mathbb{T} \notin \mathsf{ACC}$. Note this includes the case when M acts in an obviously corrupt fashion, causing C or R to abort.

The extractor E gets $\mathbb{T} \in \mathsf{ACC}$ as input so the probabilities which arise in our analysis often are conditioned on the event $\mathbb{T} \in \mathsf{ACC}$. We denote this with the convenient shorthand $\Pr_{\mathbb{T}\in\mathsf{ACC}}(\cdots)$ instead of $\Pr_{\mathbb{T}}(\cdots \,|\, \mathbb{T} \in \mathsf{ACC})$. Note that for fixed **Com**, M can be thought of as a deterministic map, mapping right query vectors to left ones. We write $\boldsymbol{\alpha} = \mathrm{M}(\tilde{\boldsymbol{\alpha}})$ to be consistent with this point of view. We assume that the transcript E gets as input is consistent with exactly one right commitment $\tilde{m}$. As $\langle \mathrm{C}, \mathrm{R} \rangle$ is statistically binding, this happens with overwhelming probability. See Figure 2.3 for a formal description of the extractor.

Tags: Let $\{t_i\}_i$ and $\{\tilde{t}_i\}_i$ be the left and right tags, respectively, in error corrected form.

Input: $\mathbb{T} = \left(\mathbf{Com}, \tilde{\boldsymbol{\alpha}}, \mathbf{a}\right) \in \mathsf{ACC}$, and a large value $N = \mathrm{poly}(\lambda)$. E is given oracle access to M.

Extraction procedure: For $j \in [N]$:

1. Rewind M to the beginning of step 2 of the protocol:
   - generate a random right challenge vector $\tilde{\boldsymbol{\beta}}_j = (\tilde{\beta}_{1,j}, \ldots, \tilde{\beta}_{n,j})$, where $\tilde{\beta}_{i,j} \in [2^{\tilde{t}_i}]$.
   - Feed M with $\tilde{\boldsymbol{\beta}}_j$ and receive challenge $\boldsymbol{\beta}_j = (\beta_{1,j}, \ldots, \beta_{n,j})$ for left interaction.
2. Generate and send $\mathbf{b}_j = (b_{1,j}, \ldots, b_{n,j})$ to M where $b_{i,j} = \begin{cases} a_i, & \beta_{i,j} = \alpha_i \\ r \xleftarrow{\mathrm{R}} \mathbb{Z}_q, & \beta_{i,j} \neq \alpha_i \end{cases}$.
   Receive response $\tilde{\mathbf{b}}_j = (\tilde{b}_{1,j}, \ldots, \tilde{b}_{n,j})$.
3. For each $i \in [n]$ use $\left\{(\tilde{\alpha}_i, \tilde{a}_i), (\tilde{\beta}_{i,j}, \tilde{b}_{i,j})\right\}$ to interpolate a line and recover candidate message $\tilde{m}_{i,j}$.
4. Repeat steps 1-3. Let $\tilde{\boldsymbol{\gamma}}_j = (\tilde{\gamma}_{1,j}, \ldots, \tilde{\gamma}_{n,j})$ be the new right challenge vector and $\tilde{\mathbf{c}}_j = (\tilde{c}_{1,j}, \ldots, \tilde{c}_{n,j})$ be the corresponding response. Let $(\tilde{m}'_{1,j}, \ldots, \tilde{m}'_{n,j})$ be the recovered candidate messages.
5. If for some $i \in [n]$, $\tilde{m}_{i,j} = \tilde{m}'_{i,j}$ and $\left\{(\tilde{\alpha}_i, \tilde{a}_i), (\tilde{\beta}_{i,j}, \tilde{b}_{i,j}), (\tilde{\gamma}_{i,j}, \tilde{c}_{i,j})\right\}$ are collinear output $\tilde{m}_{i,j}$ and halt.

Output: Output **FAIL**.

Figure 2.3: The Extractor E.

Note that there are two ways for E to fail to output $\tilde{m}$. The first is if E fails to extract any value and outputs **FAIL**. The other is if E accidentally extracts an incorrect value $\tilde{m}' \neq \tilde{m}$. The second way requires M to answer a pair of queries incorrectly but in such a way so that they yield the same candidate message and they "pass the collinearity test". In this case we say that M answers *incorrectly but collinearly.*

Definition 2.4 (Incorrect but Collinear). *Fix* $i \in \{1, \ldots, n\}$. *Let* $(\tilde{\boldsymbol{\beta}}, \tilde{\boldsymbol{b}})$ *and* $(\tilde{\boldsymbol{\gamma}}, \tilde{\boldsymbol{c}})$ *denote two query/response pairs arising during the execution of* E *while rewinding* M. *Suppose that interpolating* $(\tilde{\beta}_i, \tilde{b}_i)$ *and* $(\tilde{\gamma}_i, \tilde{c}_i)$ *against the main thread's point* $(\tilde{\alpha}_i, \tilde{a}_i)$ *produces the same candidate message* $\tilde{m}'$. *We say that* M *answers* $(\tilde{\beta}_i, \tilde{\gamma}_i)$ incorrectly but collinearly *if:*

1. $\tilde{m}' \neq \tilde{m}$; *and*
2. $\big\{(\tilde{\alpha}_i, \tilde{a}_i), (\tilde{\beta}_i, \tilde{b}_i), (\tilde{\gamma}_i, \tilde{c}_i)\big\}$ *are collinear.*

We define the set $\mathsf{IBC}^i(\tilde{\alpha}_i) = \big\{(\tilde{\boldsymbol{\beta}}, \tilde{\boldsymbol{\gamma}}) : \text{M } \textit{answers } (\tilde{\beta}_i, \tilde{\gamma}_i) \textit{ incorrectly but collinearly}\big\}$. *Finally, define*

$$\mathsf{IBC} = \big\{\mathbb{T} \in \mathsf{ACC} : (\tilde{\boldsymbol{\beta}}, \tilde{\boldsymbol{\gamma}}) \in \mathsf{IBC}^i(\tilde{\alpha}_i) \textit{ for some } i\big\}.$$

Note that $\mathsf{IBC}^i(\tilde{\alpha}_i)$ is well defined given $\mathbb{T}$ and E's randomness. Intuitively IBC is the set of transcripts for which E might fail because M answers incorrectly but collinearly. The following claim shows that these transcripts rarely occur.

Claim 2.1. $\Pr_{\mathbb{T} \in \mathsf{ACC}}\big(\mathbb{T} \in \mathsf{IBC}\big) = \mathbf{negl}(\lambda)$.

Proof. Fix $i \in \{1, \ldots, n\}$ and let $\mathbb{T}, \mathbb{T}' \in \mathsf{ACC}$ be main threads with the same prefix and $i-$th right queries $\tilde{\alpha}_i$ and $\tilde{\alpha}'_i$, respectively. Moreover, we fix E's randomness arbitrarily making it deterministic, so that the sets $\mathsf{IBC}^i(\tilde{\alpha}_i)$ and $\mathsf{IBC}^i(\tilde{\alpha}'_i)$ are defined. Note that $\mathsf{IBC}^i(\tilde{\alpha}_i)$ and $\mathsf{IBC}^i(\tilde{\alpha}'_i)$ are disjoint. Indeed, suppose $(\tilde{\boldsymbol{\beta}}, \tilde{\boldsymbol{\gamma}}) \in \mathsf{IBC}^i(\tilde{\alpha}_i) \cap \mathsf{IBC}^i(\tilde{\alpha}'_i)$. Then the four points

$$\big\{(\tilde{\alpha}_i, \tilde{a}_i), (\tilde{\alpha}'_i, \tilde{a}'_i), (\tilde{\beta}_i, \tilde{b}_i), (\tilde{\gamma}_i, \tilde{c}_i)\big\}$$

are collinear. This means that the line they all lie on is correct because $(\tilde{\alpha}_i, \tilde{a}_i)$ and $(\tilde{\alpha}'_i, \tilde{a}'_i)$ are correct $(\mathbb{T}, \mathbb{T}' \in \mathsf{ACC})$ and so $(\tilde{\boldsymbol{\beta}}, \tilde{\boldsymbol{\gamma}}) \notin \mathsf{IBC}^i(\tilde{\alpha}_i) \cup \mathsf{IBC}^i(\tilde{\alpha}'_i)$ as M answered $\tilde{\beta}_i$ and $\tilde{\gamma}_i$ correctly. Therefore, for a fixed prefix **Com** and extractor queries $(\tilde{\boldsymbol{\beta}}, \tilde{\boldsymbol{\gamma}})$, there is at most one value of $\tilde{\alpha}_i$ such that $(\tilde{\boldsymbol{\beta}}, \tilde{\boldsymbol{\gamma}}) \in \mathsf{IBC}^i(\tilde{\alpha}_i)$. As the set of possible $\tilde{\alpha}_i$ is superpolynomial, the chances that R's query $\tilde{\boldsymbol{\alpha}}$ in $\mathbb{T}$ is such that $(\tilde{\boldsymbol{\beta}}, \tilde{\boldsymbol{\gamma}}) \in \bigcup_i \mathsf{IBC}^i(\tilde{\alpha}_i)$ for any extractor query $(\tilde{\boldsymbol{\beta}}, \tilde{\boldsymbol{\gamma}})$ is negligible. The result follows. □

So we see that if our extractor never outputs the wrong message $\tilde{m}' \neq \tilde{m}$ and so if E outputs **FAIL**, it does so because it fails to receive correct answers to its queries. This allows us to formulate conditions which make E's chances of success overwhelmingly high in terms of the behavior of M. This will be useful moving forward.

Next, we define EXT, the set of "extractable" transcripts, on which M has a non-negligible chance of answering a query correctly even given random responses to its queries on the left.

Definition 2.5 (Extractable Transcripts). *Fix* $\varepsilon^* = \big(\lambda/N\big)^{1/2}$. *We define*

$$\mathsf{EXT}_i = \Big\{\mathbf{Com} : \Pr_{\tilde{\boldsymbol{\beta}}}\big(\text{M } \textit{correctly answers } \tilde{\beta}_i \big| \mathbf{Com} \ \&\ \text{M}\textit{'s queries answered randomly}\big) \geq \varepsilon^*\Big\}.$$

Set $\mathsf{EXT} = \big\{\mathbb{T} \in \mathsf{ACC} : \mathbf{Com} \in \mathsf{EXT}_i \textit{ for some } i\big\}$.

Intuitively, EXT is the set of transcripts such that M has good probability of providing at least one correct answer to a query in a rewind despite the fact that E provides random answers to M's queries. We now prove that if a transcript is in EXT then E succeeds in extracting $\tilde{m}$ whp.

Claim 2.2. $\Pr\big(\mathrm{E}(\mathbb{T}) \neq \tilde{m} \big| \mathbb{T} \in \mathsf{EXT}\big) = \mathbf{negl}(\lambda)$, *where the probability is over* $\mathbb{T}$ *and the randomness of* E.

Proof. Let $\mathbf{E}_{i,j}$ be the event that M answers both $i-$th queries correctly in rewind j. Since $\mathbb{T} \in \mathsf{EXT}$, there exists some i such that $\Pr(\mathbf{E}_{i,j}) \geq (\varepsilon^*)^2 = \lambda/N$ for all j. As the $\mathbf{E}_{i,j}$ are independent, the expected number of $\mathbf{E}_{i,j}$ which occur during the lifetime of E is at least λ. So we see that

$$\Pr\big(\mathrm{E}(\mathbb{T}) \neq \tilde{m}\big) = \Pr\big(\mathbb{T} \in \mathsf{IBC}\big) + \Pr(\text{no } \mathbf{E}_{i,j} \text{ occur}\big) = \mathbf{negl}(\lambda),$$

by Claim 2.1 and the Chernoff bound. □

At this point we formally make the assumption that there exists a PPT algorithm that can distinguish $\{\mathbf{MIM}_{\langle \mathrm{C}(m),\mathrm{R}\rangle}\}_m$ and $\{\mathbf{MIM}_{\langle \mathcal{S},\mathcal{S}\rangle}\}$ with probability at least $2p$ for some non-negligible $p = p(\lambda)$. It then suffices to prove that E succeeds with probability at least $1 - p$ since this will imply that E extracts $\tilde{m}$ AND the distinguisher correctly determines whether $(\tilde{m}, v)$ comes from the real or ideal world with probability at least p. This breaks the hiding of Com. The required theorem is stated below.

Theorem 2.2 (Sufficient for Theorem 2.1). *Let* E *be the extractor described in Figure 2.3, and let* $\mathbb{T}$ *be the transcript it is given as input. Let* $\tilde{m}$ *be* M*'s commitment in the right interaction of* $\mathbb{T}$*. Then*

$$\Pr\big(\mathrm{E}(\mathbb{T}) \neq \tilde{m}\big) \leq p,$$

where the probability is over the randomness of E*.*

2.4.2 Useful and Interesting Transcripts

In this section we define USEFUL to be the set of transcripts for which M has a good chance of completing given the prefix and challenges. This requirement on M's behavior is important in order for E to have any hope of successfully extracting $\tilde{m}$. Indeed, if M just aborts in every rewind, E will not stand a chance. From this standpoint, USEFUL is the set of transcripts

which give E "something to work with."

Definition 2.6 (**Useful Transcripts**). *Fix non-negligible* $\delta < \frac{1}{3}$. *We define*

1. $\mathsf{BAD}_1 = \left\{\mathbf{Com} : \Pr_{\mathbb{T}}\left(\mathbb{T} \in \mathsf{ACC} \middle| \mathbf{Com}\right) \leq \frac{\delta p^2}{3}\right\}$;
2. $\mathsf{BAD}_2 = \left\{(\mathbf{Com}, \boldsymbol{\alpha}) : \Pr_{\mathbb{T}}\left(\mathbb{T} \in \mathsf{ACC} \middle| \mathbf{Com}\ \&\ \mathrm{M}(\tilde{\boldsymbol{\alpha}}) = \boldsymbol{\alpha}\right) \leq \frac{\delta p^2}{3}\right\}$;
3. $\mathsf{BAD}_3 = \left\{(\mathbf{Com}, \tilde{\boldsymbol{\alpha}}) : \Pr_{\mathbb{T}}\left(\mathbb{T} \in \mathsf{ACC} \middle| \mathbf{Com}\ \&\ \tilde{\alpha}_i\right) \leq \frac{\delta p^2}{3n}\ \textit{for some}\ i \in \{1, \ldots, n\}\right\}$.

Set $\mathsf{USEFUL} := \left\{\mathbb{T} \in \mathsf{ACC} : \mathbf{Com} \notin \mathsf{BAD}_1\ \&\ (\mathbf{Com}, \boldsymbol{\alpha}) \notin \mathsf{BAD}_2\ \&\ (\mathbf{Com}, \tilde{\boldsymbol{\alpha}}) \notin \mathsf{BAD}_3\right\}$.

Informally, the BAD_i are sets of partial transcripts for which M is unlikely to complete the protocol. BAD_1 is a set of prefixes while BAD_2 (resp. BAD_3) are sets of prefixes and left (resp. right) query vectors. We start by proving that most transcripts are indeed useful. The proof involves little more than conditional probability and so may be freely bypassed by an informal reader.

Claim 2.3. $\Pr_{\mathbb{T} \in \mathsf{ACC}}\left(\mathbb{T} \notin \mathsf{USEFUL}\right) \leq \delta p$.

Proof. It suffices to prove that the quantities

$$\Pr_{\mathbb{T} \in \mathsf{ACC}}\left(\mathbf{Com} \in \mathsf{BAD}_1\right);\ \Pr_{\mathbb{T} \in \mathsf{ACC}}\left((\mathbf{Com}, \boldsymbol{\alpha}) \in \mathsf{BAD}_2\right);\ \Pr_{\mathbb{T} \in \mathsf{ACC}}\left((\mathbf{Com}, \tilde{\boldsymbol{\alpha}}) \in \mathsf{BAD}_3\right)$$

are all less than or equal to $\frac{\delta p}{3}$. All three are proven using conditional probability. We have

$$\Pr_{\mathbb{T} \in \mathsf{ACC}}\left(\mathbf{Com} \in \mathsf{BAD}_1\right) = \Pr_{\mathbb{T}}\left(\mathbf{Com} \in \mathsf{BAD}_1 \middle| \mathbb{T} \in \mathsf{ACC}\right) \leq \frac{\Pr_{\mathbb{T}}\left(\mathbb{T} \in \mathsf{ACC} \middle| \mathbf{Com} \in \mathsf{BAD}_1\right)}{\Pr_{\mathbb{T}}(\mathbb{T} \in \mathsf{ACC})} \leq \frac{\delta p}{3},$$

using the definition of BAD_1 and the fact that $\Pr_{\mathbb{T}}(\mathbb{T} \in \mathsf{ACC}) \geq p$. Similarly,

$$\Pr_{\mathbb{T} \in \mathsf{ACC}}\left((\mathbf{Com}, \boldsymbol{\alpha}) \in \mathsf{BAD}_2\right) \leq \frac{\Pr_{\mathbb{T}}\left(\mathbb{T} \in \mathsf{ACC} \middle| (\mathbf{Com}, \boldsymbol{\alpha}) \in \mathsf{BAD}_2\right)}{\Pr_{\mathbb{T}}(\mathbb{T} \in \mathsf{ACC})} \leq \frac{\delta p}{3}.$$

And finally, for BAD_3, set $Z_i = \big\{(\mathbf{Com}, \tilde{\alpha}_i) : \Pr_{\mathbb{T}}\big(\mathbb{T} \in \mathsf{ACC} \big| \mathbf{Com}\ \&\ \tilde{\alpha}_i\big) \leq \frac{\delta p^2}{3n}\big\}$. We have

$$\begin{aligned}
\Pr_{\mathbb{T}\in\mathsf{ACC}}\big((\mathbf{Com}, \tilde{\boldsymbol{\alpha}}) \in \mathsf{BAD}_3\big) &= \Pr_{\mathbb{T}}\big(\exists\ i \text{ st } (\mathbf{Com}, \tilde{\alpha}_i) \in Z_i \big| \mathbb{T} \in \mathsf{ACC}\big) \\
&\leq \sum_{i=1}^{n} \Pr_{\mathbb{T}}\big((\mathbf{Com}, \tilde{\alpha}_i) \in Z_i \big| \mathbb{T} \in \mathsf{ACC}\big) \\
&\leq \sum_{i=1}^{n} \frac{\Pr_{\mathbb{T}}\big(\mathbb{T} \in \mathsf{ACC} \big| (\mathbf{Com}, \tilde{\alpha}_i) \in Z_i\big)}{\Pr_{\mathbb{T}}(\mathbb{T} \in \mathsf{ACC})} \leq \frac{\delta p}{3}.
\end{aligned}$$

□

The transcripts in EXT are those for which M is likely to correctly answer a right query even given incorrect responses to its own left queries. On the other hand, USEFUL can be thought of as the transcripts for which M answers the right queries correctly if given correct answers to its left queries. This leads us to the following definition.

Definition 2.7 (Interesting Transcripts). *We define* $\mathsf{INT} = \mathsf{USEFUL} \setminus \mathsf{EXT}$.

Transcripts in INT are troublesome as essentially, they are transcripts for which M answers the right queries correctly if given correct answers to its left queries, but incorrectly if given incorrect answers to its left queries. Certainly, the hiding of Com ensures that M cannot *know* whether it receives correct or random responses to its queries on the left. So this difference in behavior suggests that we may be able to use M to break the hiding of Com. However, it is not so easy. Keep in mind, M does not have to know whether it is giving a correct or incorrect answer on the left. Indeed, almost all mauling attacks one could think of would have the property that M answers correctly on the right if and only if it gets correct answers on the left. The following lemma comprises the heart of our analysis.

Lemma 2.1. *If* Com *is computationally hiding then there exists a constant* $\delta' < \frac{1}{3}$ *such that*

$$\Pr_{\mathbb{T}\in\mathsf{ACC}}\big(\mathbb{T} \in \mathsf{INT}\big) \leq \delta' p.$$

Lemma 2.1 combined with Claims 2.2 and 2.3 give us

$$\Pr_{\mathbb{T}\in\mathsf{ACC}}\big(\mathrm{E}(\mathbb{T}) \neq \tilde{m}\big) \leq \delta p + \delta' p + \mathbf{negl}(\lambda) < p,$$

proving Theorem 2.2.

2.4.3 Proof of Lemma 2.1 − Part 0: Proof Overview

At a very high level the proof proceeds by considering the possible different ways in which M's left queries $\boldsymbol{\alpha}$ can "depend" on right queries $\tilde{\boldsymbol{\alpha}}$ given to M. Intuitively, $\alpha_{i'}$ being dependent on $\tilde{\alpha}_i$ is the result of M performing a mauling attack. Suppose that M mauls $\mathrm{Com}(f_{i'})$ in order to obtain $\mathrm{Com}(\tilde{f}_i)$. Then M does not know $\tilde{f}_i$ and so cannot hope to answer the query $\tilde{\alpha}_i$ except by mauling C's answer to query $\alpha_{i'}$. Therefore, if M is rewound to the beginning of step 2 and asked a different query vector $\tilde{\boldsymbol{\beta}}$ such that $\tilde{\beta}_i = \tilde{\alpha}_i$, M will have to ask $\boldsymbol{\beta}$ such that $\beta_{i'} = \alpha_{i'}$ if it wants to answer successfully. In this way, dependence can be thought of as a quantitative outcome of a mauling attack.

Very roughly speaking we then classify the set of possible transcripts into three categories based on their dependencies (the actual proof will involve parameterized variants):

- $1-2$: There exist (i_1, i_2, i') such that $\alpha_{i'}$ depends on both $\tilde{\alpha}_{i_1}$ and $\tilde{\alpha}_{i_2}$.
- UNBAL: There exist $i' > i$ such that $\alpha_{i'}$ depends on $\tilde{\alpha}_i$.
- IND: There exists i such that each $\alpha_{i'}$ does *not* depend on $\tilde{\alpha}_i$.

The main line of reasoning of our proof will consist of arguing that the above types of depenencies are impossible. We will show that if (1) either one of them or, (2) none of them occur then either the extraction procedure E will succeed or M can be used to violate hiding.

To see why this is true consider first the case where $1-2$-type dependencies exist.

This means that M is using $f_{i'}(\alpha_{i'})$ in order to obtain both $\tilde{f}_{i_1}(\tilde{\alpha}_{i_1})$ and $\tilde{f}_{i_2}(\tilde{\alpha}_{i_2})$, which is impossible given that $\tilde{\alpha}_{i_1}$ and $\tilde{\alpha}_{i_2}$ are chosen at random. This corresponds to Claim 2.5 below.

Next, consider UNBAL-type dependencies. In this case, M is using $f_{i'}(\alpha_{i'})$ to answer $\tilde{\alpha}_i$ with $\tilde{f}_i(\tilde{\alpha}_i)$. But because $i' > i$, we have that the challenge space $[2^{t_{i'}}]$ for $\alpha_{i'}$ is significantly larger than the challenge space $[2^{\tilde{t}_i}]$ for $\tilde{\alpha}_i$, which intuitively means that M is "wasting challenge space". This will mean that the residual right challenge space is super-polynomially bigger than the residual left challenge space. This corresponds to Claim 2.6 below.

To take advantage of the observation that the residual right challenge space is much larger than the residual left challenge space, we define $\mathsf{SUPER-POLY}$ to be the set of transcripts such that the number of $\tilde{\boldsymbol{\alpha}}$ for which $M(\tilde{\boldsymbol{\alpha}}) = \boldsymbol{\alpha}$ is super-polynomial. We show that if a transcript is in $\mathsf{SUPER-POLY}$ then E will succeed in extracting. Roughly speaking, this is because M cannot hope to answer the right challenge $\tilde{\boldsymbol{\alpha}}$ from the information contained in C's answer to the left challenge $\boldsymbol{\alpha}$ alone; it must know some "extra information" about the polynomials $\tilde{\mathbf{f}}$. This extra information is precisely what will let us extract. The above argument corresponds to Claim 2.8 below.

The proof of Claim 2.8 relies on the computational hiding of the underlying commitment. To enable the proof we use a variant of the standard hiding game, shown in Figure 1.1. This game is at the heart of the "computational" component of our analysis.

Given that neither $1-2$ nor UNBAL hold, we are left with IND. Consider first the case in which IND does not hold. Then, for all i there will exist an i' such that $\alpha_{i'}$ depends on $\tilde{\alpha}_i$. Because neither $1-2$ nor UNBAL hold this actually implies that for all i α_i depends on $\tilde{\alpha}_i$. To see this, note that α_1 must depend on $\tilde{\alpha}_1$. Indeed, something must depend on $\tilde{\alpha}_1$, and it cannot be $\alpha_{i'}$ for $i' > 1$. Similarly, either α_1 or α_2 must be dependent on $\tilde{\alpha}_2$. But since α_1 is

dependent on $\tilde{\alpha}_1$ it must be that α_2 is dependent on $\tilde{\alpha}_2$ and so on. Using the fact that many of the tags on the left differ from all of the tags on the right we can show that when restricted appropriately (namely to the coordinates where the right tags are larger than the left tags) the challenge space on the right will be super polynomially bigger than the challenge space on the left (bringing us back to the case described above). This corresponds to Claim 2.7 below.

Finally, we are left with the case in which IND holds. In this case, there exists some i such that none of the left challenges are dependent on $\tilde{\alpha}_i$ (a non-malleable cryptographer's fantasy). Intuitively this means that M does not need any of the left challenges in order to correctly return $\tilde{f}_i(\tilde{\alpha}_i)$, implying that it knows some information about the polynomial $\tilde{f}_i$, allowing our extractor E to succeed. This corresponds to Claim 2.9.

2.4.4 Proof of Lemma 2.1 – Part 1: Analyzing Dependencies

Given a first message **Com** (which implicitly determines $\tilde{\mathbf{Com}} = \mathrm{M}(\mathbf{Com})$), we say that a right query vector $\tilde{\boldsymbol{\alpha}}$ is *honest* if M answers $\tilde{\boldsymbol{\alpha}}$ honestly in the right interaction given correct responses to its queries $\boldsymbol{\alpha} = \mathrm{M}(\tilde{\boldsymbol{\alpha}})$ in the left interaction.

We denote the set of honest right query vectors by $\mathsf{HON}_{\mathbf{Com}}$, or just HON when **Com** is clear from context (as in the case when, for example, M is rewound to the beginning of step 2 and asked a new query vector). We write $\Pr_{\tilde{\boldsymbol{\alpha}} \in \mathsf{HON}}(\cdots)$ as shorthand for $\Pr_{\tilde{\boldsymbol{\alpha}}}(\cdots \,|\, \tilde{\boldsymbol{\alpha}} \in \mathsf{HON})$. The following claim is a direct corollary of Claim 2.3.

Claim 2.4. *If* $\mathbb{T} \in \mathsf{INT}$ *then for a new right query vector* $\tilde{\boldsymbol{\beta}}$ *in a rewind,*

$$\Pr_{\tilde{\boldsymbol{\beta}}}(\tilde{\boldsymbol{\beta}} \in \mathsf{HON}) \geq \frac{\delta p^2}{3} \text{ and } \Pr_{\tilde{\boldsymbol{\beta}}}(\tilde{\boldsymbol{\beta}} \in \mathsf{HON} \,|\, \tilde{\beta}_i = \tilde{\alpha}_i) \geq \frac{\delta p^2}{3n}$$

for all $i = 1, \ldots, n$.

The remainder of our proof assumes that $\Pr_{\mathbb{T}\in\mathsf{ACC}}\big(\mathbb{T}\in\mathsf{INT}\big)\geq\delta' p$, and from this lower bound we are able to lower bound the probability that M behaves in certain ways. Our goal is to ensure that by setting N, the number of times E rewinds, to a sufficiently large polynomial either E will successfully extract $\tilde{m}$ or M breaks the hiding of Com. We have already introduced non-negligible p, constants $\delta,\delta'<1/3$, and $N=\text{poly}(\lambda)$ for a yet unspecified polynomial. Shortly we will introduce parameters $\varepsilon=1/n-\varepsilon'$ where $\varepsilon'=1/2n^2$, and also $\ell=N/(2\lambda)$. We will require that $(\varepsilon'\delta\delta' p^4)/(24n)\geq 2/\ell$ and $(\sigma\delta^2 p^5)/18\geq 2/\ell$ where $\sigma=(\varepsilon'\delta' p)^2/(65n^6)$ is defined for convenience. All in all, setting $N=\omega\big(\lambda n^{10}p^{-7}\big)$ will suffice.

We now turn to formally define $\varepsilon-$dependence.

Definition 2.8 ($\varepsilon-$**dependence**). *For fixed* $\mathbb{T}\in\mathsf{ACC}$ *and* $i,i'\in\{1,\ldots,n\}$, *we say* $\alpha_{i'}$ *is* $\varepsilon-$dependent *on* $\tilde{\alpha}_i$ *if* $\Pr_{\tilde{\boldsymbol{\beta}}\in\mathsf{HON}}\big(\beta_{i'}=\alpha_{i'}\big|\tilde{\beta}_i=\tilde{\alpha}_i\big)\geq\varepsilon$. *We say that* $\tilde{\alpha}_i$ *is* $\varepsilon-$*independent if none of the* $\alpha_{i'}$ *are* $\varepsilon-$*dependent on* $\tilde{\alpha}_i$.

Note that if $\varepsilon>\varepsilon'$ and $\alpha_{i'}$ is $\varepsilon-$dependent on $\tilde{\alpha}_i$, then $\alpha_{i'}$ is also $\varepsilon'-$dependent on $\tilde{\alpha}_i$. Additionally, notice that though our definition does leave open the possibility that there could be more than one value which is $\varepsilon-$dependent on $\tilde{\alpha}_i$, there can only be polynomially many (at most ε^{-1} to be exact). We call these values the $\varepsilon-$*dependencies* of $\tilde{\alpha}_i$. This notion is different from $\varepsilon-$dependence defined above only because the $\varepsilon-$dependencies are defined regardless of what queries were asked in $\mathbb{T}$, whereas we only say that $\alpha_{i'}$ is $\varepsilon-$dependent on $\tilde{\alpha}_i$ if both $\tilde{\alpha}_i$ and $\alpha_{i'}$ appear in $\mathbb{T}$. For the remainder of the proof we fix non-negligible values ε and ε' such that $\varepsilon=\frac{1}{n}-\varepsilon'$ and $\varepsilon'<\frac{1}{n^2}$.

Definition 2.9 (**Special Sets of Transcripts**). *Fix* $\omega=\omega(1)$. *Define the following sets of transcripts:*

1. $\mathsf{1{-}2}:=\big\{\mathbb{T}\in\mathsf{ACC}:\exists\ (i_1,i_2,i')\ st\ \alpha_{i'}\ is\ \varepsilon'-dependent\ on\ both\ \tilde{\alpha}_{i_1}\ and\ \tilde{\alpha}_{i_2}\big\}$;
2. $\mathsf{UNBAL}:=\big\{\mathbb{T}\in\mathsf{ACC}:\exists\ i'>i\ st\ \alpha_{i'}\ is\ \varepsilon'-dependent\ on\ \tilde{\alpha}_i\big\}$;

3. $\mathsf{IND} := \big\{\mathbb{T} \in \mathsf{ACC} : \exists\ i\ st\ \tilde{\alpha}_i\ is\ \varepsilon - independent\big\}$;

4. $\mathsf{SUPER{-}POLY} := \big\{\mathbb{T} \in \mathsf{ACC} : {}^{\#}\{\tilde{\boldsymbol{\alpha}} : \mathrm{M}(\tilde{\boldsymbol{\alpha}}) = \boldsymbol{\alpha}\} \geq \lambda^{\omega}\big\}$.

Claim 2.5. *Fix* $\sigma = \frac{(\varepsilon'\delta'p)^2}{65n^6}$. *If* $\mathrm{Pr}_{\mathbb{T}\in\mathsf{ACC}}\big(\mathbb{T} \in \mathsf{1{-}2}\ \&\ \mathbb{T} \in \mathsf{INT}\big) \geq \frac{\delta'p}{4}$, *then*

$$\mathrm{Pr}_{\mathbb{T}\in\mathsf{ACC}}\big(\mathbb{T} \in \mathsf{SUPER{-}POLY}\ \&\ \mathbb{T} \in \mathsf{INT}\big) \geq \sigma.$$

notation: Let R and L be the sets of right and left query vectors respectively. We have already seen that for fixed **Com**, $\mathrm{M} : R \longrightarrow L$ deterministically maps $\tilde{\boldsymbol{\alpha}}$ to $\boldsymbol{\alpha}$. We write $R^i(\tilde{\alpha}_i)$ and $L^{i'}(\alpha_{i'})$ to mean the sets of right and left query vectors whose $i-$th and $i'-$th coordinates are fixed on $\tilde{\alpha}_i$ and $\alpha_{i'}$, respectively. We write $\mathrm{M} : R^i_\tau(\tilde{\alpha}_i) \longrightarrow L^{i'}(\alpha_{i'})$ if M maps a τ fraction of $R^i(\tilde{\alpha}_i)$ to $L^{i'}(\alpha_{i'})$.

Proof. Let (i_1, i_2, i') be such that $\mathrm{Pr}_{\mathbb{T}\in\mathsf{ACC}}\big(\alpha_{i'}$ is $\varepsilon' -$ dependent on $\tilde{\alpha}_{i_1}$ and $\tilde{\alpha}_{i_2}\big) \geq \frac{\delta'p}{4n^3}$. Such (i_1, i_2, i') must exist by hypothesis of Claim 2.5. Now fix **Com** and $\tilde{\alpha}_{i_1}$. With probability at least $\frac{\delta'p}{8n^3}$ over $(\mathbf{Com}, \tilde{\alpha}_{i_1})$, $\mathrm{Pr}_{\tilde{\alpha}_{i_2}}\big(\tilde{\alpha}_{i_1}$ and $\tilde{\alpha}_{i_2}$ share an $\varepsilon'-$dependency$\big) \geq \frac{\delta'p}{8n^3}$. However, since $\tilde{\alpha}_{i_1}$ has at most $(\varepsilon')^{-1}$ $\varepsilon'-$dependencies, there must exist some $\alpha_{i'}$ such that $\mathrm{Pr}_{\tilde{\alpha}_{i_2}}\big(\alpha_{i'}$ an $\varepsilon' -$ dependence of $\tilde{\alpha}_{i_2}\big) \geq \frac{\varepsilon'\delta'p}{8n^3}$. It follows that

$$\mathrm{Pr}(\alpha_{i'}) \geq \mathrm{Pr}\big(\alpha_{i'}\big|\alpha_{i'}\text{ an }\varepsilon'-\text{dependence of }\tilde{\alpha}_i\big)\cdot\mathrm{Pr}_{\tilde{\alpha}_i}(\alpha_{i'}\text{ an }\varepsilon'-\text{dependence of }\tilde{\alpha}_i) \geq \frac{(\varepsilon')^2\delta'p}{8n^3}.$$

Let $\tau = \frac{(\varepsilon')^2\delta'p}{8n^3}$. Then we see that $\mathrm{M} : R_\tau \to L^{i'}(\alpha_{i'})$, where $\big|R_\tau\big| = \tau\big|R\big| \geq \tau 2^{\lambda}\big|L^{i'}(\alpha_{i'})\big|$. If ω is such that $\big|R^i_\tau(\tilde{\alpha}_i)\big| = \lambda^{2\omega}\big|L^{i'}(\alpha_{i'})\big|$ then with probability at least $(1-\lambda^{-\omega})\tau$ over $\tilde{\boldsymbol{\alpha}} \in R^i(\tilde{\alpha}_i)$, ${}^{\#}\{\tilde{\boldsymbol{\alpha}} : \mathrm{M}(\tilde{\boldsymbol{\alpha}}) = \boldsymbol{\alpha}\} \geq \lambda^{\omega}$. So we see that

$$\mathrm{Pr}_{\mathbb{T}\in\mathsf{ACC}}\big(\mathbb{T} \in \mathsf{SUPER{-}POLY}\ \&\ \mathbb{T} \in \mathsf{INT}\big) \geq \frac{\tau\delta'p}{8n^3} - \mathbf{negl}(\lambda) \geq \frac{(\varepsilon'\delta'p)^2}{65n^6} = \sigma.$$

□

Claim 2.6. *Fix* $\sigma = \frac{(\varepsilon'\delta' p)^2}{65n^6}$. *If* $\Pr_{\mathbb{T}\in\mathsf{ACC}}\big(\mathbb{T} \in \mathsf{UNBAL} \;\&\; \mathbb{T} \in \mathsf{INT}\big) \geq \frac{\delta' p}{4}$, *then*

$$\Pr_{\mathbb{T}\in\mathsf{ACC}}\big(\mathbb{T} \in \mathsf{SUPER{-}POLY} \;\&\; \mathbb{T} \in \mathsf{INT}\big) \geq \sigma.$$

Proof. Fix a random first message **Com**. With probability at least $\frac{\delta' p}{8}$ over **Com**,

$$\Pr_{\tilde{\boldsymbol{\alpha}}\in\mathsf{HON}}\big(\exists\ i' > i \text{ st } \alpha_{i'} \text{ is } \varepsilon' - \text{dependent on } \tilde{\alpha}_i\big) \geq \frac{\delta' p}{8}.$$

Let $\tau' = \frac{\varepsilon'\delta' p}{8}$. It follows that $\mathrm{M} : R^i_{\tau'}(\tilde{\alpha}_i) \to L^{i'}(\alpha_{i'})$. But as $i' > i$, we have that

$$\big|R^i_{\tau'}(\tilde{\alpha}_i)\big| = \tau'\big|R^i(\tilde{\alpha}_i)\big| \geq \tau' 2^{\lambda}\big|L^{i'}(\alpha_{i'})\big|,$$

So when restricted appropriately, M is exponentially many to one on average. As in Claim 2.5, it follows that

$$\Pr_{\mathbb{T}\in\mathsf{ACC}}\big(\mathbb{T} \in \mathsf{SUPER{-}POLY} \;\&\; \mathbb{T} \in \mathsf{INT}\big) \geq \frac{\tau'\delta' p}{8} - \mathbf{negl}(\lambda) \geq \frac{\varepsilon'(\delta' p)^2}{65} \geq \sigma.$$

□

Claim 2.7. *If* $\Pr_{\mathbb{T}\in\mathsf{ACC}}\big(\mathbb{T} \notin \mathsf{1{-}2} \cup \mathsf{UNBAL} \cup \mathsf{IND} \;\&\; \mathbb{T} \in \mathsf{INT}\big) \geq \frac{\delta' p}{4}$, *then*

$$\Pr_{\mathbb{T}\in\mathsf{ACC}}\big(\mathbb{T} \in \mathsf{SUPER{-}POLY} \;\&\; \mathbb{T} \in \mathsf{INT}\big) \geq \sigma.$$

Proof. Consider the consequences of $\mathbb{T} \notin \mathsf{1{-}2} \cup \mathsf{UNBAL} \cup \mathsf{IND}$. If $\mathbb{T} \notin \mathsf{IND}$ then for every i, there exists at least one i' such that $\alpha_{i'}$ is $\varepsilon-$dependent on $\tilde{\alpha}_i$. However, if $\mathbb{T} \notin \mathsf{UNBAL}$, then for all $i' > i$, $\alpha_{i'}$ cannot be $\varepsilon-$dependent on $\tilde{\alpha}_i$ (since $\varepsilon \geq \varepsilon'$). Moreover, if $\mathbb{T} \notin \mathsf{1{-}2}$, then there do not exist (i_1, i_2, i') such that $\alpha_{i'}$ is $\varepsilon-$dependent on $\tilde{\alpha}_{i_1}$ and $\tilde{\alpha}_{i_2}$ (again using $\varepsilon' \geq \varepsilon'$). It follows that for each i, α_i must be $\varepsilon-$dependent on $\tilde{\alpha}_i$. Indeed, α_1 must be

$\varepsilon-$dependent on $\tilde{\alpha}_1$ as something must depend on $\tilde{\alpha}_1$ and it cannot be α_i for $i > 1$. Next, either α_1 or α_2 must be $\varepsilon-$dependent on $\tilde{\alpha}_2$ and it cannot be α_1 as that is already dependent on $\tilde{\alpha}_1$. Continuing in this fashion, we deduce that each α_i is $\varepsilon-$dependent on $\tilde{\alpha}_i$.

Now, since each α_i is $\varepsilon-$dependent on $\tilde{\alpha}_i$, Claim 2.5 ensures that for all $i' \neq i$, α_i is not $\varepsilon'-$dependent on $\tilde{\alpha}_{i'}$. It follows that for all i, $\Pr_{\tilde{\boldsymbol{\beta}} \in \mathsf{HON}}\big(\exists\ i' \neq i \text{ st } \beta_{i'} = \alpha_{i'} \big| \tilde{\beta}_i = \tilde{\alpha}_i\big) \leq \varepsilon' n$. As $\mathbb{T} \notin \mathsf{IND}$, we have that for all i,

$$\begin{aligned}
\Pr_{\tilde{\boldsymbol{\beta}} \in \mathsf{HON}}\big(\beta_i = \alpha_i \big| \tilde{\beta}_i = \tilde{\alpha}_i\big) &\geq \Pr_{\tilde{\boldsymbol{\beta}} \in \mathsf{HON}}\big(\exists\ i' \text{ st } \beta_{i'} = \alpha_{i'} \big| \tilde{\beta}_i = \tilde{\alpha}_i\big) \\
&- \Pr_{\tilde{\boldsymbol{\beta}} \in \mathsf{HON}}\big(\exists\ i' \neq i \text{ st } \beta_{i'} = \alpha_{i'} \big| \tilde{\beta}_i = \tilde{\alpha}_i\big). \\
&\geq \varepsilon n - \varepsilon' n = 1 - 2\varepsilon' n,
\end{aligned}$$

so we see that, in fact, each α_i is $(1 - 2\varepsilon' n)-$dependent on $\tilde{\alpha}_i$. As $2\varepsilon' n < \frac{1}{2}$, each $\tilde{\alpha}_i$ has a unique $(1 - 2\varepsilon' n)-$depencence.

Now, let $S = \{i : t_i \geq \tilde{t}_i\}$, and fix random $\mathbf{Com}$ and $\tilde{\boldsymbol{\alpha}}_S$, where $\tilde{\boldsymbol{\alpha}}_S = (\tilde{\alpha}_i)_{i \in S}$. Since we are given that $\Pr_{\mathbb{T} \in \mathsf{ACC}}\big(\mathbb{T} \notin \mathsf{1{-}2} \cup \mathsf{UNBAL} \cup \mathsf{IND}\ \&\ \mathbb{T} \in \mathsf{INT}\big) \geq \frac{\delta' p}{4}$, with probability at least $\frac{\delta' p}{8}$ over our choice of $(\mathbf{Com}, \tilde{\boldsymbol{\alpha}}_S)$, we will have

$$\Pr_{\tilde{\boldsymbol{\alpha}} \in \mathsf{HON}}\big(\text{each } \alpha_i \text{ is } (1 - 2\varepsilon' n) - \text{dependent on } \tilde{\alpha}_i \big| (\mathbf{Com}, \tilde{\boldsymbol{\alpha}}_S)\big) \geq \frac{\delta' p}{8}.$$

However, as each $\tilde{\alpha}_i$ has a unique $(1 - 2\varepsilon' n)-$dependence, it follows that

$$\Pr_{\tilde{\boldsymbol{\beta}} \in \mathsf{HON}}\big(\beta_{i'} = \alpha_{i'} \text{ for all } i' \in S \big| \tilde{\beta}_i = \tilde{\alpha}_i \text{ for all } i \in S\big) \geq \frac{\delta' p}{8}.$$

So we see that $\Pr_{\tilde{\boldsymbol{\beta}} \in \mathsf{HON}}\big(S \subset \{i' : \beta_{i'} = \alpha_{i'}\} \big| \tilde{\beta}_i = \tilde{\alpha}_i \text{ for all } i \in S\big) \geq \frac{\delta' p}{8}$ with probability at least $\frac{\delta' p}{8}$ over our choice of $(\mathbf{Com}, \tilde{\boldsymbol{\alpha}}_S)$.

Let $\tau'' = \frac{\delta' p}{8}$. We have shown that $\mathrm{M} : R^S_{\tau''}(\tilde{\boldsymbol{\alpha}}_S) \to L^S(\boldsymbol{\alpha}_S)$. But

$$\big|R^S_{\tau''}(\tilde{\boldsymbol{\alpha}}_S)\big| = \tau''\big|R^S(\tilde{\boldsymbol{\alpha}}_S)\big| \geq \tau'' 2^\lambda \big|L^S(\boldsymbol{\alpha}_S)\big|.$$

So M is exponentially many to one on average, which means that with all but negligible probability, $\boldsymbol{\alpha}$ has exponentially many preimages. It follows that

$$\Pr_{\mathbb{T}\in\mathsf{ACC}}\big(\mathbb{T} \in \mathsf{SUPER-POLY} \;\&\; \mathbb{T} \in \mathsf{INT}\big) \geq \frac{\tau''\delta' p}{8} - \mathbf{negl}(\lambda) \geq \frac{(\delta' p)^2}{65} \geq \sigma.$$

□

2.4.5 Proof of Lemma 2.1 – Part 2: Reductions to the Hiding of $\langle \mathrm{C}, \mathrm{R}\rangle$

In this section complete the proof of Lemma 2.1 by proving two claims which show how to use an M with unlikely behavior to break the hiding of $\langle \mathrm{C}, \mathrm{R}\rangle$. We first give an intuitive description of our method of argument. This description is slightly technical but does not get into the specifics of either Claim 2.8 or Claim 2.9.

Our adversary $\mathcal{A}$ is defined as follows:

- $\mathcal{A}$ chooses random $m_0, m_1 \in \mathbb{Z}_q$ and sends (m_0, m_1) to a challenger $\mathcal{C}$, signaling the beginning of the hiding game of $\langle \mathrm{C}, \mathrm{R}\rangle$.
- $\mathcal{A}$ instantiates M and runs two sessions of $\langle \mathrm{C}, \mathrm{R}\rangle$ until the end of the response message of both executions, forwarding the messages it receives as C to $\mathcal{C}$. In the left execution, $\mathcal{C}$ commits to m_u for secret $u \in \{0, 1\}$. More specifically:
  - $\mathcal{A}$, acting as R, sends $\tilde{\sigma}$ to M, and receives σ which it forwards to $\mathcal{C}$.
  - $\mathcal{A}$ then receives **Com** from $\mathcal{C}$ which it forwards to M, and receives $\tilde{\mathbf{Com}}$.

- $\mathcal{A}$ sends random $\tilde{\boldsymbol{\alpha}}$ such that $\tilde{\alpha}_i \in \left[2^{\tilde{t}_i}\right]$ to M, receiveing $\boldsymbol{\alpha}$ which it forwards to $\mathcal{C}$.
- $\mathcal{A}$ receives $\mathbf{a}$ from $\mathcal{C}$ which it forwards to M, obtaining $\tilde{\mathbf{a}}$.
- $\mathcal{A}$ continues forwarding messages between M and $\mathcal{C}$ during the zero-knowledge proof phase of $\langle \mathrm{C}, \mathrm{R} \rangle$, playing honestly as R in the right interaction.
- When the proofs are finished, $\mathcal{A}$ verifies both π and $\tilde{\pi}$. If either is not accepted, $\mathcal{A}$ aborts. Let $\mathbb{T} = \left(\mathbf{Com}, \tilde{\boldsymbol{\alpha}}, \mathbf{a}\right)$ be the resulting transcript.

- $\mathcal{A}$ chooses random $u' \in \{0, 1\}$ and defines polynomial vector $\mathbf{f}$ such that $\mathbf{f}(\boldsymbol{\alpha}) = \mathbf{a}$ and every coordinate of $\mathbf{f}$ has constant term $m_{u'}$.

- $\mathcal{A}$ rewinds M to the beginning of the query phase of the right execution and sends a new query $\tilde{\boldsymbol{\beta}}$, receiving left query $\boldsymbol{\beta}$. It can do this many times, resulting in a set of new right queries $\{\tilde{\boldsymbol{\beta}}, \tilde{\boldsymbol{\gamma}}, \dots\}$.

- $\mathcal{A}$ answers the left queries it obtained in the previous step with $\mathbf{f}$, and receives a right response. It collects the points it receives into the set $\left\{(\tilde{\boldsymbol{\alpha}}, \tilde{\mathbf{a}}), (\tilde{\boldsymbol{\beta}}, \tilde{\mathbf{b}}), (\tilde{\boldsymbol{\gamma}}, \tilde{\mathbf{c}}), \dots\right\}$.

- $\mathcal{A}$ tests whether the points $\left\{(\tilde{\boldsymbol{\alpha}}, \tilde{\mathbf{a}}), (\tilde{\boldsymbol{\beta}}, \tilde{\mathbf{b}}), (\tilde{\boldsymbol{\gamma}}, \tilde{\mathbf{c}}), \dots\right\}$ satisfy some condition. If so, then $\mathcal{A}$ outputs u', if not it outputs $1 - u'$.

Exactly what condition $\mathcal{A}$ tests for will change between the two proofs. In the proof of Claim 2.8, $\mathcal{A}$ checks that the points $\left\{(\tilde{\boldsymbol{\alpha}}, \tilde{\mathbf{a}}), (\tilde{\boldsymbol{\beta}}, \tilde{\mathbf{b}}), (\tilde{\boldsymbol{\gamma}}, \tilde{\mathbf{c}})\right\}$ are collinear, while in the proof of Claim 2.9, $\mathcal{A}$ checks whether $\tilde{b}_i = \tilde{a}_i$ for some preselected i. The important thing however, is that the condition be satisfied when M answers correctly, but not when M answers incorrectly. Note that if $u' = u$ then responses generated with $\mathbf{f}$ are correct and so if $\mathbb{T} \in \mathsf{USEFUL}$, then we can lower bound the probability that M answers correctly on the right using Claim 2.4. On the other hand, if $u' \neq u$ then the responses on the left are random (as long as $\beta_i \neq \alpha_i$

for all $i \in [n]$). If $\mathbb{T} \notin \mathsf{EXT}$ then we have an upper bound on the probability that M answers any right query correctly. These observations together tell us that there is a non-negligible gap between the probability is satisfied when $u' = u$ and when $u' \neq u$. This gap translates to $\mathcal{A}$ having a noticeable advantage of winning the hiding game.

One issue with the above is that we have assumed that $\mathbb{T} \in \mathsf{INT}$ when in reality we area only allowed to assume that $\mathbb{T} \in \mathsf{INT}$ with probability at least $\frac{\delta' p}{4}$. The following fact says that as long as the gap is large enough, assuming $\mathbb{T} \in \mathsf{INT}$, this actually does not matter.

Fact 2.1. *Consider a condition that the set* $\{(\tilde{\boldsymbol{\alpha}}, \tilde{\boldsymbol{a}}), (\tilde{\boldsymbol{\beta}}, \tilde{\boldsymbol{b}}), (\tilde{\boldsymbol{\gamma}}, \tilde{\boldsymbol{c}}), \dots\}$ *either satisfies or not, as described in the above paragraphs. Let* $\mathbf{E}$ *be an event such that:*

- $\Pr_{\mathbb{T} \in \mathsf{ACC}}(\mathbf{E}) \geq \xi$;
- $\Pr\big(\text{Condition satisfied} \big| u' = u \ \&\ \mathbf{E}\big) \geq \xi'$;
- $\Pr\big(\text{Condition satisfied} \big| u' \neq u \ \&\ \mathbf{E}\big) \leq \xi''$,

for non-negligible values ξ, ξ', ξ'' *satisfying* $\xi'' \leq (p\xi\xi')/8$. *Then* $\mathcal{A}$ *breaks the hiding of* $\langle \mathrm{C}, \mathrm{R} \rangle$.

Proof. Fix $\ell = 1/(2\xi'')$ and let $\mathcal{A}$ play in an $\ell-$way version of the usual hiding game of $\langle \mathrm{C}, \mathrm{R} \rangle$ as follows:

- $\mathcal{A}$ chooses random $m_1, \dots, m_\ell \in \mathbb{Z}_q$ and sends $(m_1, \dots, m_\ell)$ to $\mathcal{C}$.
- $\mathcal{A}$ instantiates M and runs two sessions of $\langle \mathrm{C}, \mathrm{R} \rangle$ until the end of the response message of both executions, forwarding the messages it receives as C to $\mathcal{C}$. In the left execution, $\mathcal{C}$ commits to $m_{j'}$ for secret $j' \in [\ell]$. More specifically:
- For each $j \in [\ell]$, $\mathcal{A}$ defines polynomial vectors $\mathbf{g}_j$ such that $\mathbf{g}_j(\boldsymbol{\alpha}) = \mathbf{a}$ and every coordinate of $\mathbf{g}_j$ has constant term m_j.

- $\mathcal{A}$ rewinds M to the beginning of the query phase of the right execution and sends a new queries $\tilde{\boldsymbol{\beta}}, \tilde{\boldsymbol{\gamma}}, \ldots$ receiving left queries $\boldsymbol{\beta}, \boldsymbol{\gamma}, \ldots$.

- For each $j \in [\ell]$, $\mathcal{A}$ answers the left queries it obtained in the previous step with $\mathbf{g}_j$, and receives a right response. It collects the set $\big\{(\tilde{\boldsymbol{\alpha}}, \tilde{\mathbf{a}}), (\tilde{\boldsymbol{\beta}}, \tilde{\mathbf{b}}_j), (\tilde{\boldsymbol{\gamma}}, \tilde{\mathbf{c}}_j), \ldots\big\}_j$.

- $\mathcal{A}$ tests whether the points $\big\{(\tilde{\boldsymbol{\alpha}}, \tilde{\mathbf{a}}), (\tilde{\boldsymbol{\beta}}, \tilde{\mathbf{b}}_j), (\tilde{\boldsymbol{\gamma}}, \tilde{\mathbf{c}}_j), \ldots\big\}$ satisfy the condition. If so, then $\mathcal{A}$ outputs $j^* = j$ and halts.

We have

$$\begin{aligned}
\Pr(\mathcal{A} \text{ wins}) &\geq \Pr_{\mathbb{T}}\big(\mathbb{T} \in \mathsf{ACC}\big) \cdot \Pr_{\mathbb{T} \in \mathsf{ACC}}(\mathbf{E}) \\
&\cdot \Pr\big(\text{Condition satisfied when } j = j' \big| \mathbf{E}\big) \\
&\cdot \Pr\big(\text{Condition not satisfied whenever } j \neq j' \big| \mathbf{E}\big) \\
&\geq (p\xi\xi') \cdot \Pr\big(\text{Not } \mathbf{E}'_j \text{ for all } j \neq j' \big| \mathbf{E}\big).
\end{aligned}$$

where $\mathbf{E}'_j$ is the event

$$\mathbf{E}'_j : \text{ “Conditions are satisfied when } \mathbf{g}_j \text{ is used to answer left queries.”}$$

We are given that $\Pr\big(\mathbf{E}'_j \big| \mathbf{E}\big) \leq \xi''$ for all $j \neq j'$, and as the $\mathbf{E}'_j$ are independent this means that the expected number of $\mathbf{E}'_j$ which occur is at most $\xi''\ell = 1/2$. It follows that

$$\Pr(\mathcal{A} \text{ wins}) \geq (p\xi\xi') \cdot \Pr\big(\text{No } \mathbf{E}'_j \text{ occur when } j \neq j' \big| \mathbf{E}\big) \geq \frac{p\xi\xi'}{2} \geq \frac{2}{\ell},$$

which means that $\mathcal{A}$'s chances of winning the hiding game are noticeably greater than $1/\ell$, which violates the hiding of Com. □

Claim 2.8. *Fix* $\sigma = \frac{(\varepsilon'\delta' p)^2}{65n^6}$. *If* $\Pr_{\mathbb{T}\in\mathsf{ACC}}\big(\mathbb{T} \in \mathsf{SUPER}{-}\mathsf{POLY}\ \&\ \mathbb{T} \in \mathsf{INT}\big) \geq \sigma$ *then there exists a PPT* $\mathcal{A}$ *who breaks the hiding of* $\langle \mathrm{CR}\rangle$.

Proof. Our $\mathcal{A}$ proceeds as follows.

- As described above, $\mathcal{A}$ chooses random $m_0, m_1 \in \mathbb{Z}_q$ and begins the hiding game, sending (m_0, m_1) to $\mathcal{C}$. Then $\mathcal{A}$ instantiates M and runs two sessions of $\langle \mathrm{C}, \mathrm{R}\rangle$ forwarding the messages it receives as C to $\mathcal{C}$. In the left interaction, $\mathcal{C}$ commits to m_u for unknown $u \in \{0,1\}$. Let $\mathbb{T} = \big(\mathbf{Com}, \tilde{\boldsymbol{\alpha}}, \mathbf{a}\big)$ be the resulting transcript. Additionally, $\mathcal{A}$ chooses random $u' \in \{0,1\}$ and defines the polynomial vector $\mathbf{f}$, to be the unique such vector so that $\mathbf{f}(\boldsymbol{\alpha}) = \mathbf{a}$ and so that every coordinate of $\mathbf{f}$ has constant term $m_{u'}$.

- $\mathcal{A}$ chooses two new random challenge vectors $\tilde{\boldsymbol{\beta}}$ and $\tilde{\boldsymbol{\gamma}}$ such that $\tilde{\beta}_i, \tilde{\gamma}_i \in \big[2^{\tilde{t}_i}\big]$. It rewinds M back to the beginning of the right execution's query phase and sends $\tilde{\boldsymbol{\beta}}$ receiving left query $\boldsymbol{\beta}$. It responds with $\mathbf{b} = \mathbf{f}(\boldsymbol{\beta})$ receiving right response $\tilde{\mathbf{b}}$. It repeats this process, sending challenge $\tilde{\boldsymbol{\gamma}}$, answering $\boldsymbol{\gamma}$ with $\mathbf{c} = \mathbf{f}(\boldsymbol{\gamma})$ and receiving $\tilde{\mathbf{c}}$.

- $\mathcal{A}$ checks whether the points $\big\{(\tilde{\boldsymbol{\alpha}}, \tilde{\mathbf{a}}), (\tilde{\boldsymbol{\beta}}, \tilde{\mathbf{b}}), (\tilde{\boldsymbol{\gamma}}, \tilde{\mathbf{c}})\big\}$ are collinear. If so, $\mathcal{A}$ outputs u', if not $\mathcal{A}$ outputs $1 - u'$.

In light of Fact 2.1, it suffices to prove that:

1. $\Pr\big(\big\{(\tilde{\boldsymbol{\alpha}}, \tilde{\mathbf{a}}), (\tilde{\boldsymbol{\beta}}, \tilde{\mathbf{b}}), (\tilde{\boldsymbol{\gamma}}, \tilde{\mathbf{c}})\big\}\ \text{collinear}\big|u' = u\ \&\ \mathbb{T} \in \mathsf{SUPER}{-}\mathsf{POLY} \cap \mathsf{INT}\big) \geq \frac{\delta^2 p^4}{18}$;
2. $\Pr\big(\big\{(\tilde{\boldsymbol{\alpha}}, \tilde{\mathbf{a}}), (\tilde{\boldsymbol{\beta}}, \tilde{\mathbf{b}}), (\tilde{\boldsymbol{\gamma}}, \tilde{\mathbf{c}})\big\}\ \text{collinear}\big|u' \neq u\ \&\ \mathbb{T} \in \mathsf{SUPER}{-}\mathsf{POLY} \cap \mathsf{INT}\big) \leq 2\varepsilon^*$,

since $2\varepsilon^* \leq (\sigma\delta^2 p^5)/144$. For the first quantity, note that $\big\{(\tilde{\boldsymbol{\alpha}}, \tilde{\mathbf{a}}), (\tilde{\boldsymbol{\beta}}, \tilde{\mathbf{b}}), (\tilde{\boldsymbol{\gamma}}, \tilde{\mathbf{c}})\big\}$ are collinear if M answers correctly. As $\mathbb{T} \in \mathsf{USEFUL}$, M answers $\tilde{\boldsymbol{\beta}}$ and $\tilde{\boldsymbol{\gamma}}$ correctly with probability at least $\big(\delta p^2/3\big)^2$ if $u' = u$ (i.e., if M receives correct responses on the left) by Claim 2.4. We see that

$$\Pr\big(\big\{(\tilde{\boldsymbol{\alpha}}, \tilde{\mathbf{a}}), (\tilde{\boldsymbol{\beta}}, \tilde{\mathbf{b}}), (\tilde{\boldsymbol{\gamma}}, \tilde{\mathbf{c}})\big\}\ \text{collinear}\big|u' = u\ \&\ \mathbb{T} \in \mathsf{SUPER}{-}\mathsf{POLY} \cap \mathsf{INT}\big) \geq \frac{\delta^2 p^4}{18}.$$

For the second quantity, define the temporary event

$$\mathbf{E}' : \text{“}\left\{(\tilde{\boldsymbol{\alpha}}, \tilde{\mathbf{a}}), (\tilde{\boldsymbol{\beta}}, \tilde{\mathbf{b}}), (\tilde{\boldsymbol{\gamma}}, \tilde{\mathbf{c}})\right\} \text{ are collinear.”}$$

We have

$$\begin{aligned}
\Pr\left(\mathbf{E}' \middle| u' \neq u \;\&\; \mathbb{T} \in \mathsf{SUPER}-\mathsf{POLY} \cap \mathsf{INT}\right) &\leq \Pr\left(\mathbf{E}' \middle| \tilde{\boldsymbol{\beta}} \text{ is incorrect}\right) \\
&+ \Pr\left(\tilde{\boldsymbol{\beta}} \text{ is correct} \middle| u' \neq u \;\&\; \mathbb{T} \notin \mathsf{EXT}\right) \\
&\leq \Pr\left(\mathbf{E}' \middle| \tilde{\boldsymbol{\beta}} \text{ is incorrect}\right) + \varepsilon^*,
\end{aligned}$$

as if $u' \neq u$ then the answer M receives to $\boldsymbol{\beta}$ is random (i.e., distributed identically to an answer from E) and $\mathbb{T} \notin \mathsf{EXT}$. Therefore, proving that $\Pr\left(\mathbf{E}' \middle| \tilde{\boldsymbol{\beta}} \text{ is incorrect}\right) = \mathbf{negl}(\lambda)$ completes the proof of Claim 2.8.

Suppose that $\tilde{\boldsymbol{\alpha}}$ and $\tilde{\boldsymbol{\alpha}}'$ are such that $\mathrm{M}(\tilde{\boldsymbol{\alpha}}) = \boldsymbol{\alpha} = \mathrm{M}(\tilde{\boldsymbol{\alpha}}')$ and such that M answers $\tilde{\boldsymbol{\alpha}}$ and $\tilde{\boldsymbol{\alpha}}'$ honestly given correct response $\mathbf{a}$ to $\boldsymbol{\alpha}$. Note that it cannot be the case that $\left\{(\tilde{\boldsymbol{\alpha}}, \tilde{\mathbf{a}}), (\tilde{\boldsymbol{\beta}}, \tilde{\mathbf{b}}), (\tilde{\boldsymbol{\gamma}}, \tilde{\mathbf{c}})\right\}$ and $\left\{(\tilde{\boldsymbol{\alpha}}', \tilde{\mathbf{a}}'), (\tilde{\boldsymbol{\beta}}, \tilde{\mathbf{b}}), (\tilde{\gamma}, \tilde{\mathbf{c}})\right\}$ are collinear as this would mean that the four points

$$\left\{(\tilde{\boldsymbol{\alpha}}, \tilde{\mathbf{a}}), (\tilde{\boldsymbol{\alpha}}', \tilde{\mathbf{a}}'), (\tilde{\boldsymbol{\beta}}, \tilde{\mathbf{b}}), (\tilde{\boldsymbol{\gamma}}, \tilde{\mathbf{c}})\right\}$$

lie on the same line, and moreover, that this is the correct line as it contains the correct points $(\tilde{\boldsymbol{\alpha}}, \tilde{\mathbf{a}})$ and $(\tilde{\boldsymbol{\alpha}}', \tilde{\mathbf{a}}')$. This contradicts the hypothesis that $\tilde{\mathbf{b}}$ is an incorrect answer. So we see that for each $\boldsymbol{\alpha}$ there exists at most one $\tilde{\boldsymbol{\alpha}}$ such that

1. $\mathrm{M}(\tilde{\boldsymbol{\alpha}}) = \boldsymbol{\alpha}$;

2. M answers $\tilde{\boldsymbol{\alpha}}$ honestly;

3. $\left\{(\tilde{\boldsymbol{\alpha}}, \tilde{\mathbf{a}}), (\tilde{\boldsymbol{\beta}}, \tilde{\mathbf{b}}), (\tilde{\boldsymbol{\gamma}}, \tilde{\mathbf{c}})\right\}$ are collinear.

As $\mathbb{T} \in \mathsf{SUPER-POLY}$, there are at least λ^{ω} values of $\tilde{\boldsymbol{\alpha}}$ such that number 1 holds, and as $\mathbb{T} \notin \mathsf{BAD}_2$, M answers a non-negligible fraction of these honestly. It follows that there are a superpolynomial number of $\tilde{\boldsymbol{\alpha}}$ satisfying 1 and 2, which means that the probability that $\mathcal{A}$ chose the unique $\tilde{\boldsymbol{\alpha}}$ such that all three hold is negligible. □

Claim 2.9. *If* $\Pr_{\mathbb{T}\in\mathsf{ACC}}\big(\mathbb{T} \in \mathsf{IND} \;\&\; \mathbb{T} \in \mathsf{INT}\big) \geq \frac{\delta' p}{4}$ *then there exists a PPT* $\mathcal{A}$ *who breaks the hiding of* $\langle \mathrm{C}, \mathrm{R} \rangle$.

Proof. Our $\mathcal{A}$ proceeds as follows.

- As described above, $\mathcal{A}$ chooses random $m_0, m_1 \in \mathbb{Z}_q$ and begins the hiding game, sending (m_0, m_1) to $\mathcal{C}$. Then $\mathcal{A}$ instantiates M and runs two sessions of $\langle \mathrm{C}, \mathrm{R} \rangle$ forwarding the messages it receives as C to $\mathcal{C}$. In the left interaction, $\mathcal{C}$ commits to m_u for unknown $u \in \{0, 1\}$. Let $\mathbb{T} = \big(\mathbf{Com}, \tilde{\boldsymbol{\alpha}}, \mathbf{a}\big)$ be the resulting transcript. Additionally, $\mathcal{A}$ chooses random $u' \in \{0, 1\}$ and defines the polynomial vector $\mathbf{f}$, to be the unique such vector so that $\mathbf{f}(\boldsymbol{\alpha}) = \mathbf{a}$ and so that every coordinate of $\mathbf{f}$ has constant term $m_{u'}$.

- $\mathcal{A}$ chooses random $i \in [n]$ and random challenge vector $\tilde{\boldsymbol{\beta}}$ such that $\tilde{\beta}_i = \tilde{\alpha}_i$. It rewinds M back to the beginning of the right execution's query phase and sends $\tilde{\boldsymbol{\beta}}$ receiving left query $\boldsymbol{\beta}$. If $\beta_{i'} = \alpha_{i'}$ for any $i' \in [n]$ then $\mathcal{A}$ aborts. If not, $\mathcal{A}$ responds with $\mathbf{b} = \mathbf{f}(\boldsymbol{\beta})$ receiving right response $\tilde{\mathbf{b}}$.

- $\mathcal{A}$ checks whether $\tilde{b}_i = \tilde{a}_i$. If so, $\mathcal{A}$ outputs u', if not $\mathcal{A}$ outputs $1 - u'$.

First note that

$$\begin{aligned}
\Pr_{\tilde{\boldsymbol{\beta}}}\big(\mathcal{A} \text{ not abort} \big| \mathbb{T} \in \mathsf{IND} \cap \mathsf{INT}\big) \;&\geq\; \Pr\big(\tilde{\alpha}_i \; \varepsilon - \text{independent} \big| \mathbb{T} \in \mathsf{IND} \cap \mathsf{INT}\big) \\
&\quad \cdot \; \Pr_{\tilde{\boldsymbol{\beta}}}\big(\beta_{i'} \neq \alpha_{i'} \forall i' \big| \tilde{\alpha}_i \; \varepsilon - \text{independent} \;\&\; \tilde{\beta}_i = \tilde{\alpha}_i\big). \\
&\geq\; \frac{1}{n} \cdot (n\varepsilon') = \varepsilon'.
\end{aligned}$$

So by Fact 2.1, it suffices to prove that:

1. $\Pr\big(\tilde{b}_i = \tilde{a}_i \big| u' = u \;\&\; u' = u\mathbb{T} \in \mathsf{IND} \cap \mathsf{INT} \;\&\; \mathcal{A} \text{ not abort}\big) \geq \frac{\delta p^2}{3n}$;

2. $\Pr\big(\tilde{b}_i = \tilde{a}_i \big| u' \neq u \;\&\; \mathbb{T} \in \mathsf{IND} \cap \mathsf{INT} \;\&\; \mathcal{A} \text{ not abort}\big) \leq \varepsilon^*$,

since $\varepsilon^* \leq (\delta\delta'\varepsilon' p^4)/96n$. For the first quantity, note that $\tilde{b}_i = \tilde{a}_i$ if M answers correctly. As $\mathbb{T} \in \mathsf{USEFUL}$, M answers $\tilde{\boldsymbol{\beta}}$ correctly with probability at least $\delta p^2/3n$ if $u' = u$ by Claim 2.4. For the second quantity, note that if $u' \neq u$ then the answers M receives on the left are distributed identically to the responses it gets from E (this is using that $\beta_{i'} \neq \alpha_{i'}$ for all $i' \in [n]$), and so $\Pr\big(\tilde{b}_i = \tilde{a}_i \big| u' \neq u \;\&\; \mathbb{T} \in \mathsf{INT} \;\&\; \mathcal{A} \text{ not abort}\big) \leq \varepsilon^*$. □

Claims 2.5 through 2.9 combine to give that if Com is computationally hiding, then

$$\begin{aligned}
\Pr_{\mathbb{T}\in\mathsf{ACC}}\big(\mathbb{T} \in \mathsf{INT}\big) \;&\leq\; \Pr_{\mathbb{T}\in\mathsf{ACC}}\big(\mathbb{T} \in \mathsf{IND} \;\&\; \mathbb{T} \in \mathsf{INT}\big) + \Pr_{\mathbb{T}\in\mathsf{ACC}}\big(\mathbb{T} \in \mathsf{UNBAL} \;\&\; \mathbb{T} \in \mathsf{INT}\big) \\
&+\; \Pr_{\mathbb{T}\in\mathsf{ACC}}\big(\mathbb{T} \in \mathsf{1{-}2} \;\&\; \mathbb{T} \in \mathsf{INT}\big) \\
&+\; \Pr_{\mathbb{T}\in\mathsf{ACC}}\big(\mathbb{T} \notin \mathsf{IND} \cup \mathsf{UNBAL} \cup \mathsf{1{-}2} \;\&\; \mathbb{T} \in \mathsf{INT}\big) \\
&\leq\; \frac{\delta' p}{4} + \frac{\delta' p}{4} + \frac{\delta' p}{4} + \frac{\delta' p}{4} = \delta' p,
\end{aligned}$$

completing the proof of Lemma 2.1, Theorem 2.2 and Theorem 2.1.

2.4.6 Many-Many Non-Malleability

Though we have only proved non-malleability in the setting where M engages in a single left and right session, our proof may be extended using known techniques to accomodate the situation where M engages in polynomially many sessions on the right and left. The proof is in two steps.

First, we show that our protocol is one-many non-malleable following the techniques of [Goy11]. The observation is that we have many sessions on the right but only one on the

left, we can polynomially many copies of our extractor, one for each right session. Then since none of the extractors needs to rewind the left session, we can prove that they all succeed whp by the union bound.

Next, we use the transformation of [LPV08], that any one-many non-malleable commitment scheme is also many-many non-malleable. The proof is a hybrid argument. The intuition is that if having many sessions on the left compromised non-malleability, then there would be (at least) one session whose existence was to blame. This violates one-many non-malleability.

2.5 Optimizing Communication

In this section we address the issue of minimizing the communication complexity of our scheme $\langle \mathrm{C}, \mathrm{R} \rangle$ from the previous section. A first glance does not offer much hope as the query/response phase appears to be the most communication heavy, requiring a total of $\omega(n^2 \log \lambda)$ bits to be sent back and forth, and the size of the challenge space being this large played an important role in the proof of non-malleability. However, viewing it from another angle and asking the question "is there any way we can use the same protocol to commit to a larger message?" offers some hope. Though this would not decrease the communication in one run of $\langle \mathrm{C}, \mathrm{R} \rangle$, it would reduce the communication *per bit of commitment message*.

Committing to Different Messages in each Coordinate. Recall that in the commitment phase of our protocol, C commits to the coefficients of n linear polynomials, each with constant term m. Our first approach is to remove the requirement that all polynomials have the same constant term. Note that this will not allow C's secret message to be $(m_1, \ldots, m_n)$, the set of all constant terms. The reason is that our proof of non-malleability requires the extractor E to be able to extract M's commitment on the right, which, if all polynomials share the same constant term, ammounts to proving that there is one coordinate where extraction is possible. In order for E to extract $(\tilde{m}_1, \ldots, \tilde{m}_n)$, we would have to prove that extraction is possible *in all coordinates*, which is too much to hope for. Certainly extraction cannot be guaranteed in coordinates where $t_i = \tilde{t}_i$. However, if we are able to amend our proof to be able to show that extraction is possible from say a constant fraction of the coordinates then we could let the constant terms be secret shares of the commitment message. Now extracting in $\Omega(n)$ coordinates will be sufficient to reconstruct the secret, which if we use a secret sharing scheme with constant rate, such as [FY92]'s packed variant of Shamir's secret sharing scheme [Sha79], can be $\Omega(n)$ different messages.

In order to prove that extraction is possible from many coordinates we must make a small modification to our protocol because of the following attack. Recall that the tags are arranged in such a way so that $\tilde{t}_i < t_{i'}$ for all $i' > i$. Suppose after receiving C's commitment message, which is a commitment to the coefficients of linear polynomials $f_1 \dots f_n$, M sends commitments to the coefficients of $\tilde{f}_1, \dots, \tilde{f}_n$ where $\tilde{f}_n$ is random and $\tilde{f}_i = f_{i+1}$ for all $i \leq n-1$. Then upon receiving challenge $(\tilde{\alpha}_1, \dots, \tilde{\alpha}_n)$ from R it sends $\boldsymbol{\alpha}$ to C where $\alpha_i = \tilde{\alpha}_{i-1}$ for all $i \geq 2$. Note that this is sure to be a legal query as $\left[2^{\tilde{t}_{i-1}}\right] \subset \left[2^{t_i}\right]$ for all $i \geq 2$. Upon receiving left response $\mathbf{a}$, M sets $\tilde{a}_n = \tilde{f}_n(\tilde{\alpha}_n)$ and $\tilde{a}_i = a_{i+1}$ for all $i \leq n-1$. Such behavior from M ensures that E can only extract from the last coordinate.

In order to fix the protocol so the above attack is not possible, we arrange the commitment message so that it consists of commitments to the coefficients of $n/2$ quadratic polynomials instead of n linear polynomials. Then during the query phase, the query vector $\boldsymbol{\alpha}$ is parsed so that (α_i, α_{n-i}) correspond to the $i-$th quadratic polynomial. This prevents M from shifting the polynomials it receives because each polynomial gets a query from a small domain and a large domain and furthermore, the smaller the domain is for the first query, the larger the domain is for the second. We let the $n/2$ constant terms be secret shares of C's commitment message $\mathbf{m} = (m_1, \dots, m_\ell)$ for some $\ell = \Omega(n)$. The new protocol is shown in Figure 2.4, and the updated proof of non-malleability is given in Section 2.5.2. We collect the necessary technical results into Section 2.5.1 to make exposition in Section 2.5.2 more coherent.

2.5.1 Mixed Dependencies

A large part of our proof of non-malleability involved examining different options for the dependencies between the left and right sessions and proving that certain "unlikely seeming" possibilities were in fact impossible. Changing the protocol so that more than one

Public Parameters: Tags $t_1, \ldots, t_n$, large prime q such that $q > 2^{t_i}$ for all i and $\ell = \Omega(n)$.

Commiter's Private Input: Message vector $\mathbf{m} \in \mathbb{F}_q^\ell$ to be committed to. Let $\big([\mathbf{m}]_1, \ldots, [\mathbf{m}]_{n/2}\big)$ be shares of $\mathbf{m}$ under an $\big(\ell - 1, n/2\big)-$Franklin-Yung packed secret sharing scheme [FY92].

0. R $\to$ C **Initialization message:** Send the first message σ of the Naor commitment scheme.

1. C $\to$ R **Commit message:** Sample random $r_1, \ldots, r_n \in \mathbb{F}_q$.

   - Define quadratic functions $f_1, \ldots, f_{n/2}$ by $f_i(x) = r_{n-i+1}x^2 + r_i x + [\mathbf{m}]_i$.
   - Send commitments $\mathbf{Com} = \big(\mathrm{Com}(r_{n-i+1}), \mathrm{Com}(r_i), \mathrm{Com}\big([\mathbf{m}]_i\big)\big)_{i=1,\ldots,n/2}$.

2. R $\to$ C **Query:**

   - Send random challenge vector $\boldsymbol{\alpha} = (\alpha_i, \alpha_{n-i+1})_{i=1,\ldots,n/2}$ where $\alpha_i \in [2^{t_i}] \subset \mathbb{F}_q^2$.

3. C $\to$ R **Response:**

   - Send $\mathbf{a} = (a_i, a_{n-i+1})_{i=1,\ldots,n/2}$, where $(a_i, a_{n-i+1}) = \big(f_i(\alpha_i), f_i(\alpha_{n-i+1})\big)$.

4. C $\longleftrightarrow$ R **Consistency proof:** Parties engage in a zero-knowledge argument protocol where C proves to R that the commit message is well formed and that the response is correct.

Figure 2.4: : Updated NMC scheme $\langle \mathrm{C}, \mathrm{R} \rangle_{\mathsf{MANY\text{-}COORDS}}$.

query is asked of each polynomial opens up a new possibility which seems unlikely. In this section we prove that it cannot be the case that two right queries of the same polynomial each are dependent on a left query, and moreover that these left queries are of different polynomials. Towards this end, we first prove an extension of Claim 2.2 which examines the liklihood that M gives more than one correct response on the right given that it receives exactly one correct response on the left and all of its other left queries are answered randomly.

Definition 2.10 (Related Coordinates). *We say that $i_1, i_2 \in [n]$ are* related*, written $i_1 \sim i_2$, if the queries in coordinates i_1 and i_2 are answered by the same polynomial.*

Definition 2.11 (Generalized Extractable Transcripts). *Define $\mathsf{GEN}-\mathsf{EXT}^{(i_1,i_2,i')}$ to be the set of left commitment messages* **Com** *such that* M*'s likelihood of correctly answering $\tilde{\beta}_{i_1}$ and $\tilde{\beta}_{i_2}$ on the right is $\geq \varepsilon^{**} = \lambda^2\sqrt{\varepsilon^*} = \left(4\lambda^9/N\right)^{1/4}$, if its left queries are answered as follows:*

- *$\beta_{i'}$ is answered correctly;*
- *all other β_i are answered randomly.*

These sets can be seen as generalized versions of the sets EXT^i defined in Definition 2.5. Set

$$\mathsf{GEN}-\mathsf{EXT} = \left\{\mathbb{T} \in \mathsf{ACC} : \exists\ (i_1, i_2, i') \text{ st } i_1 \sim i_2 \ \&\ \mathbf{Com} \in \mathsf{GEN}-\mathsf{EXT}^{(i_1,i_2,i')}\right\}.$$

Claim 2.10. *If $\Pr_{\mathbb{T}\in\mathsf{ACC}}\left(\mathbb{T} \in \mathsf{GEN}-\mathsf{EXT} \setminus \mathsf{EXT}\right) \geq \frac{\delta' p}{6}$, then there exists an $\mathcal{A}$ that breaks the hiding of $\langle \mathrm{C}, \mathrm{R}\rangle_{\mathsf{MANY-COORDS}}$.*

Proof. Our $\mathcal{A}$ proceeds as follows.

- $\mathcal{A}$ chooses random $\mathbf{m}_0, \mathbf{m}_1 \in \mathbb{Z}_q^\ell$ and sends $(\mathbf{m}_0, \mathbf{m}_1)$ to $\mathcal{C}$ signaling the start of the hiding game of $\langle \mathrm{C}, \mathrm{R}\rangle_{\mathsf{MANY-COORDS}}$. Then $\mathcal{A}$ instantiates M and runs two sessions of $\langle \mathrm{C}, \mathrm{R}\rangle_{\mathsf{MANY-COORDS}}$, forwarding the messages it receives as C to $\mathcal{C}$. In the left

interaction, $\mathcal{C}$ commits to $\mathbf{m}_u$ for unknown $u \in \{0,1\}$. Let $\mathbb{T} = \big(\mathbf{Com}, \tilde{\boldsymbol{\alpha}}, \mathbf{a}\big)$ be the resulting transcript. Additionally, $\mathcal{A}$ chooses random $u' \in \{0,1\}$ and $i' \in [n]$ and defines the polynomial vector $\mathbf{f}$, to be such that $\mathbf{f}(\boldsymbol{\alpha}) = \mathbf{a}$ and such that the polynomial corresponding to the $i-$th coordinate has constant term $[\mathbf{m}_{u'}]_i$. Let f denote the coordinate polynomial corresponding to the $i'-$th query. Finally, $\mathcal{A}$ initializes a counter variable c to zero.

- $\mathcal{A}$ now chooses $N' = \big(N/4\lambda^5\big)^{1/4}$ random challenge vectors $\tilde{\boldsymbol{\beta}}_1, \ldots, \tilde{\boldsymbol{\beta}}_{N'}$ and rewinds M back to the beginning of the right execution's query phase N' times, each time sending $\tilde{\boldsymbol{\beta}}_j$ and receiving $\boldsymbol{\beta}_j$.

- $\mathcal{A}$ chooses random $i_1, i_2 \in [n]$ such that $i_1 \sim i_2$. For each $j \in [N']$, $\mathcal{A}$ prepares response $\mathbf{b}_j$ by choosing $\mathbf{b}_j$ randomly except that $b_{i',j} = f(\beta_{i'})$. $\mathcal{A}$ sends $\mathbf{b}_j$ to M and receives $\tilde{\mathbf{b}}_j$. $\mathcal{A}$ checks whether the points

$$\big\{(\alpha_{i_1}, a_{i_1}), (\alpha_{i_2}, a_{i_2}), (\beta_{i_1,j}, b_{i_1,j}), (\beta_{i_2,j}, b_{i_2,j})\big\}$$

  all lie on the same quadratic. If so, $\mathcal{A}$ increments c and continues.

- If after looping over all $j \in [N']$, if $c \geq \lambda/2$ then $\mathcal{A}$ outputs u'. Otherwise, $\mathcal{A}$ outputs $1 - u'$.

For the purposes of this proof, define the events

- $\mathbf{E}$: "$\mathbb{T} \in \mathsf{GEN}-\mathsf{EXT} \setminus \mathsf{EXT}$ & (i_1, i_2, i') st $\mathbf{Com} \in \mathsf{GEN}-\mathsf{EXT}^{(i_1,i_2,i')}$."

- $\mathbf{E}'_j$: "$\big\{(\tilde{\alpha}_{i_1}, \tilde{a}_{i_1}), (\tilde{\alpha}_{i_2}, \tilde{a}_{i_2}), (\tilde{\beta}_{i_1,j}, \tilde{b}_{i_1,j}), (\tilde{\beta}_{i_2,j}, \tilde{b}_{i_2,j})\big\}$ lie on the same quadratic."

Note that $\Pr_{\mathbb{T} \in \mathsf{ACC}}(\mathbf{E}) \geq \delta' p/6n^3$ by the hypotheses of Claim 2.10. By Fact 2.1, it suffices to prove that:

1. $\Pr\big(\mathbf{E}'_j \text{ for at least } \lambda/2 \text{ values of } j \in [N'] \big| u' = u \;\&\; \mathbf{E}\big) \geq \frac{1}{2}$;

2. $\Pr\big(\mathbf{E}'_j \text{ for at least } \lambda/2 \text{ values of } j \in [N'] \big| u' \neq u \ \&\ \mathbf{E}\big) \leq 2\sqrt{\varepsilon^*}$,

since $2\sqrt{\varepsilon^*} \leq \delta' p^2/96n^3$. For the first quantity, note that if $u' = u$ then the answers M receives on the left are random except for $b_{i'}$ which is a correct response to $\beta_{i'}$. As $\mathbf{Com} \in \mathsf{GEN}-\mathsf{EXT}^{(i_1,i_2,i')}$, we see that $\Pr\big(\mathbf{E}'_j \big| u' = u \ \&\ \mathbf{E}\big) \geq \varepsilon^{**}$ for all j. As the $\mathbf{E}'_j$ are independent, the expected number of $\mathbf{E}'_j$ which occur is at least $\varepsilon^{**}N' = \lambda$. Using Markov, the probability that at least $\lambda/2$ occur is at least $1/2$.

For the second quantity, note that when $u' \neq u$, the answers M receives on the left are generated according to a random quadratic except that $b_{i''}$ is generated completely randomly. Call this quadratic Q. As $\mathbb{T} \notin \mathsf{EXT}$, we know that if M were receiving completely random answers on the left then its probability of correctly answering $\tilde{\beta}_{i_1}$ on the right would be at most ε^*. Using conditional probability we can show that with probability at least $1 - \sqrt{\varepsilon^*}$ over Q, the probability that M answers $\tilde{\beta}_{i_1}$ correctly when given random answers to all its left queries except that $\beta_{i'}$ is answered according to Q is at most $\sqrt{\varepsilon^*}$. It follows that the expected number of times M gives a correct answer to $\tilde{\beta}_{i_1}$ is at most $\sqrt{\varepsilon^*}N' = 1/\lambda$, and so by the Chernoff bound, this number is less than $\lambda/2$ with high probability. Therefore, in order to bound the second quantity, it suffices to prove that $\big(\mathbf{E}'_j \big| \tilde{\beta}_{i_1} \text{ answered incorrectly}\big) = \mathbf{negl}(\lambda)$. However, this is precisely what we proved in Claim 2.1, and so Claim 2.10 follows. □

Definition 2.12 (Transcripts with Mixed Polynomial Dependencies). *We write $i_1 \sim i_2$ if i_1 and i_2 are query coordinates for the same polynomial. Define the set*

$$\mathsf{MIXED} := \left\{ \mathbb{T} \in \mathsf{ACC} : \exists\ (i_1, i_2, i'_1, i'_2)\ st \begin{array}{ll} 1) & \alpha_{i'_b} \text{ is } \varepsilon - \text{dependent on } \tilde{\alpha}_{i_b},\ b = 1,2 \\ 2) & \nexists\ k \neq i'_b \text{ st } \alpha_k \text{ is } \varepsilon' - \text{dependent on } \tilde{\alpha}_{i_b} \\ 3) & i_1 \sim i_2 \text{ but } i'_1 \not\sim i'_2 \end{array} \right\}.$$

Claim 2.11. *If*

$$\Pr_{\mathbb{T} \in \mathsf{ACC}}\big(\mathbb{T} \in (\mathsf{INT} \cap \mathsf{MIXED}) \setminus \mathsf{GEN}-\mathsf{EXT}\big) \geq \frac{\delta' p}{6}$$

then there exists a PPT $\mathcal{A}$ who breaks the hiding of Com.

Proof. Our $\mathcal{A}$ proceeds as follows.

- $\mathcal{A}$ chooses random $\mathbf{m}_0, \mathbf{m}_1 \in \mathbb{Z}_q^\ell$ and begins the hiding game, sending $(\mathbf{m}_0, \mathbf{m}_1)$ to $\mathcal{C}$. Then $\mathcal{A}$ instantiates M and runs two sessions of $\langle \mathrm{C}, \mathrm{R} \rangle$ forwarding the messages it receives as C to $\mathcal{C}$. In the left interaction, $\mathcal{C}$ commits to $\mathbf{m}_u$ for unknown $u \in \{0,1\}$. Let $\mathbb{T} = \big(\mathbf{Com}, \tilde{\boldsymbol{\alpha}}, \mathbf{a}\big)$ be the resulting transcript. Additionally, $\mathcal{A}$ chooses random $u' \in \{0,1\}$ and defines the polynomial vector $\mathbf{f}$, to be the unique such vector so that $\mathbf{f}(\boldsymbol{\alpha}) = \mathbf{a}$ and so that every coordinate of $\mathbf{f}$ has constant term $\mathbf{m}_{u'}$.

- $\mathcal{A}$ chooses random $i_1, i_2 \in [n]$ such that $i_1 \sim i_2$ and random challenge vectors $\tilde{\boldsymbol{\beta}}, \tilde{\boldsymbol{\beta}}', \tilde{\boldsymbol{\gamma}}$ such that $\tilde{\beta}_{i_1} = \tilde{\beta}'_{i_1} = \tilde{\alpha}_{i_1}$ and $\tilde{\gamma}_{i_2} = \tilde{\alpha}_{i_2}$. It rewinds M back to the beginning of the right execution's query phase three times sending $\tilde{\boldsymbol{\beta}}, \tilde{\boldsymbol{\beta}}', \tilde{\boldsymbol{\gamma}}$, receiving left challenges $\boldsymbol{\beta}, \boldsymbol{\beta}', \boldsymbol{\gamma}$. $\mathcal{A}$ aborts unless there exist $i'_1, i'_2 \in [n]$ such that $i'_1 \not\sim i'_2$ and $\beta_{i'_1} = \beta'_{i'_1} = \alpha_{i'_1}$ but $\beta_k \neq \alpha_k$, $\beta'_k \neq \alpha_k$ for all $k \neq i'_1$ and $\gamma_{i'_2} = \alpha_{i'_2}$ but $\gamma_k \neq \alpha_k$ for all $k \neq i'_2$.

- If $\mathcal{A}$ does not abort in the previous step, it proceeds by sending left responses $\mathbf{b} = \mathbf{f}(\boldsymbol{\beta})$, $\mathbf{b}' = \mathbf{f}(\boldsymbol{\beta}')$ and $\mathbf{c} = \mathbf{f}(\boldsymbol{\gamma})$, receiving right response vectors $\tilde{\mathbf{b}}, \tilde{\mathbf{b}}', \tilde{\mathbf{c}}$.

- $\mathcal{A}$ checks whether $\tilde{b}_{i_1} = \tilde{b}'_{i_1} = \tilde{a}_{i_1}$, $\tilde{c}_{i_2} = \tilde{a}_{i_2}$ and the points

$$\big\{(\tilde{\alpha}_{i_1}, \tilde{a}_{i_1}), (\tilde{\alpha}_{i_2}, \tilde{a}_{i_2}), (\tilde{\beta}_{i_2}, \tilde{b}_{i_2}), (\tilde{\beta}'_{i_2}, \tilde{b}'_{i_2}), (\tilde{\gamma}_{i_1}, \tilde{c}_{i_1})\big\}$$

  lie on the same quadratic. If so, $\mathcal{A}$ outputs u', if not $\mathcal{A}$ outputs $1 - u'$.

For the purposes of this proof, define the events

- $\mathbf{E}$: "$\mathbb{T} \in (\mathsf{INT} \cap \mathsf{MIXED}) \setminus \mathsf{GEN{-}EXT}$ & $\mathcal{A}$ does not abort."

- $\mathbf{E}'$: "$\big\{(\tilde{\alpha}_{i_1}, \tilde{a}_{i_1}), (\tilde{\alpha}_{i_2}, \tilde{a}_{i_2}), (\tilde{\beta}_{i_2}, \tilde{b}_{i_2}), (\tilde{\beta}'_{i_2}, \tilde{b}'_{i_2}), (\tilde{\gamma}_{i_1}, \tilde{c}_{i_1})\big\}$ lie on the same quadratic."

Note that in order for $\mathcal{A}$ not to abort given that $\mathbb{T} \in (\mathsf{INT} \cap \mathsf{MIXED}) \setminus \mathsf{GEN}-\mathsf{EXT}$, $\mathcal{A}$ must choose i_1, i_2 correctly and then M must fix only the $i_1'-$th (resp. $i_2'-$th) coordinates of $\boldsymbol{\beta}$ and $\boldsymbol{\beta}'$ (resp. $\boldsymbol{\gamma}$). Clearly the former happens with probably at least $1/n^2$ and if we let Y equal the latter probability then we can bound Y as follows

$$\begin{aligned} Y &\geq \Pr_{\tilde{\boldsymbol{\beta}},\tilde{\boldsymbol{\beta}}',\tilde{\boldsymbol{\gamma}}}\big(\tilde{\boldsymbol{\beta}}, \tilde{\boldsymbol{\beta}}', \tilde{\boldsymbol{\gamma}} \in \mathsf{HON}\big) \cdot \Big(\Pr_{\tilde{\boldsymbol{\beta}} \in \mathsf{HON}}\big(\beta_{i_1'} = \alpha_{i_1'} \;\&\; \nexists\, k \neq i_1' \text{ st } \beta_k = \alpha_k \big| \tilde{\beta}_{i_1} = \tilde{\alpha}_{i_1}\big)\Big)^3 \\ &\geq \left(\frac{\delta p^2}{3n}\right)^3 \cdot \Big(\Pr_{\tilde{\boldsymbol{\beta}} \in \mathsf{HON}}\big(\beta_{i_1'} = \alpha_{i_1'} \big| \tilde{\beta}_{i_1} = \tilde{\alpha}_{i_1}\big) - \Pr_{\tilde{\boldsymbol{\beta}} \in \mathsf{HON}}\big(\exists\, k \neq i_1' \text{ st } \beta_k = \alpha_k \big| \tilde{\beta}_{i_1} = \tilde{\alpha}_{i_1}\big)\Big)^3 \\ &\geq \left(\frac{\delta p^2}{3n}\right)^3 \cdot \big(\varepsilon - n\varepsilon'\big)^3 \geq \left(\frac{\delta p^2}{12n^2}\right)^3. \end{aligned}$$

Note that we have used Claim 2.4 to lower bound the probability that $\tilde{\boldsymbol{\beta}} \in \mathsf{HON}$. We get that $\Pr_{\mathbb{T} \in \mathsf{ACC}}(\mathbf{E}) \geq (\delta^3 \delta' p^7)/(6912 n^8)$. By Fact 2.1, it suffices to prove that:

1. $\Pr\big(\mathbf{E}' \;\&\; \tilde{b}_{i_1} = \tilde{b}'_{i_1} = \tilde{a}_{i_1} \;\&\; \tilde{c}_{i_2} = \tilde{a}_{i_2} \big| u' = u \;\&\; \mathbf{E}\big) \geq \frac{\delta^3 p^6}{27 n^3}$;

2. $\Pr\big(\mathbf{E}' \;\&\; \tilde{b}_{i_1} = \tilde{b}'_{i_1} = \tilde{a}_{i_1} \;\&\; \tilde{c}_{i_2} = \tilde{a}_{i_2} \big| u' \neq u \;\&\; \mathbf{E}\big) \leq 2\varepsilon^{**}$;

since $2\varepsilon^{**} \leq (\delta^6 \delta' p^{13})/(2239488 n^{11})$. For the first quantity, note that if $u' = u$ then M gets receives correct answers on the left. Therefore, the conditions will be satisfied whenever $\tilde{\boldsymbol{\beta}}, \tilde{\boldsymbol{\beta}}', \tilde{\boldsymbol{\gamma}} \in \mathsf{HON}$, which happens with probability at least $\big(\delta p^2/3n\big)^3$ by Claim 2.4.

To examine the second quantity, note that as $\mathbb{T} \notin \mathsf{GEN}-\mathsf{EXT}$, the probability that M answers $\tilde{\beta}_{i_2}$ correctly is at most ε^{**}. This is because when $u' \neq u$, the answers on the left are all incorrect except for $b_{i'}$. Note that even though the incorrect answers are not generated randomly, they are each generated by evaluating a random quadratic polynomial vector, where each coordinate quadratic intersects the correct quadratic at two random points. One can show that these distributions are identical. Similarly, we see that M answers $\tilde{\beta}'_{i_2}$ and $\tilde{\gamma}_{i_1}$

correctly each with probability at most ε^{**}. Therefore, it suffices to show that

$$\Pr\left(\mathbf{E}' \middle| \tilde{b}_{i_2}, \tilde{b}'_{i_2}, \tilde{c}_{i_1} \text{ incorrect}\right) = \mathbf{negl}(\lambda).$$

Towards this end, for query response pair $(\tilde{\boldsymbol{\gamma}}, \tilde{\mathbf{c}})$ and a single query $\tilde{\alpha}_{i_1}$ let $\mathsf{COMP}\left(\tilde{\alpha}_{i_1}, \tilde{\boldsymbol{\gamma}}, \tilde{\mathbf{c}}\right)$ be the set of queries $\tilde{\boldsymbol{\beta}}, \tilde{\boldsymbol{\beta}}'$ such that

- $\tilde{\beta}_{i_1} = \tilde{\beta}'_{i_1} = \tilde{\alpha}_{i_1}$, $\tilde{b}_{i_1} = \tilde{b}'_{i_1} = \tilde{a}_{i_1}$ is the correct response;
- $\tilde{b}_{i_2}$ and $\tilde{b}'_{i_2}$ are incorrect responses to $\tilde{\beta}_{i_2}$ and $\tilde{\beta}'_{i_2}$;
- $\left\{(\tilde{\alpha}_{i_1}, \tilde{a}_{i_1}), (\tilde{\beta}_{i_2}, \tilde{b}_{i_2}), (\tilde{\beta}'_{i_2}, \tilde{b}'_{i_2}), (\tilde{\gamma}_{i_1}, \tilde{c}_{i_1}), (\tilde{\gamma}_{i_2}, \tilde{c}_{i_2})\right\}$ lie on the same quadratic.

Intuitively, $\mathsf{COMP}\left(\tilde{\alpha}_{i_1}, \tilde{\boldsymbol{\gamma}}, \tilde{\mathbf{c}}\right)$ is the set of queries $\tilde{\boldsymbol{\beta}}, \tilde{\boldsymbol{\beta}}'$ that if M's queries $\boldsymbol{\beta}, \boldsymbol{\beta}'$ are answered using $\mathbf{f}$ then M will answer incorrectly in both of the i_2-th coordinates, but in such a way that the conditions will be satisfied. Note that if $(\tilde{\boldsymbol{\gamma}}, \tilde{\mathbf{c}})$ and $(\tilde{\boldsymbol{\gamma}}', \tilde{\mathbf{c}}')$ are such that $\tilde{\gamma}_{i_2} \neq \tilde{\gamma}'_{i_2}$ but $\tilde{c}_{i_2}$ and $\tilde{c}'_{i_2}$ are correct responses, then for all $\tilde{\alpha}_{i_1}$,

$$\mathsf{COMP}\left(\tilde{\alpha}_{i_1}, \tilde{\boldsymbol{\gamma}}, \tilde{\mathbf{c}}\right) \cap \mathsf{COMP}\left(\tilde{\alpha}_{i_1}, \tilde{\boldsymbol{\gamma}}', \tilde{\mathbf{c}}'\right) = \emptyset.$$

To see this, suppose $(\tilde{\boldsymbol{\beta}}, \tilde{\boldsymbol{\beta}}') \in \mathsf{COMP}\left(\tilde{\alpha}_{i_1}, \tilde{\boldsymbol{\gamma}}, \tilde{\mathbf{c}}\right) \cap \mathsf{COMP}\left(\tilde{\alpha}_{i_1}, \tilde{\boldsymbol{\gamma}}', \tilde{\mathbf{c}}'\right)$. Then

1. $\left\{(\tilde{\alpha}_{i_1}, \tilde{a}_{i_1}), (\tilde{\beta}_{i_2}, \tilde{b}_{i_2}), (\tilde{\beta}'_{i_2}, \tilde{b}'_{i_2}), (\tilde{\gamma}_{i_1}, \tilde{c}_{i_1}), (\tilde{\gamma}_{i_2}, \tilde{c}_{i_2})\right\}$ lie on the same quadratic; and
2. $\left\{(\tilde{\alpha}_{i_1}, \tilde{a}_{i_1}), (\tilde{\beta}_{i_2}, \tilde{b}_{i_2}), (\tilde{\beta}'_{i_2}, \tilde{b}'_{i_2}), (\tilde{\gamma}'_{i_1}, \tilde{c}'_{i_1}), (\tilde{\gamma}'_{i_2}, \tilde{c}'_{i_2})\right\}$ lie on the same quadratic.

Call these quadratics Q_1 and Q_2. Since Q_1 and Q_2 intersect on three points, we have $Q_1 = Q_2 = Q$. But then this means that Q contains the points $\left\{(\tilde{\alpha}_{i_1}, \tilde{a}_{i_1}), (\tilde{\gamma}'_{i_1}, \tilde{c}'_{i_1}), (\tilde{\gamma}'_{i_2}, \tilde{c}'_{i_2})\right\}$ which are all points of correct responses. So Q intersects the correct quadratic at three points, making it the correct quadratic. This means that $(\tilde{\beta}_{i_2}, \tilde{b}_{i_2})$ and $(\tilde{\beta}'_{i_2}, \tilde{b}'_{i_2})$ are also correct points contradicting the second criterion for membership in COMP.

It follows that for any $\tilde{\alpha}_{i_1}$ and $\tilde{\boldsymbol{\beta}}, \tilde{\boldsymbol{\beta}}'$, there is at most one value of $\tilde{\alpha}_{i_2}$ that could cause the conditions to be satisfied if $\tilde{b}_{i_2}$ and $\tilde{b}'_{i_2}$ are incorrect responses to $\tilde{\beta}_{i_2}$ and $\tilde{\beta}'_{i_2}$. The probability that this particular $\tilde{\alpha}_{i_2}$ is the one that appears in the main thread is negligible. □

2.5.2 Proof of Non-Malleability of $\langle \mathrm{C}, \mathrm{R} \rangle_{\mathsf{MANY-COORDS}}$

In this section, we complete the proof that $\langle \mathrm{C}, \mathrm{R} \rangle_{\mathsf{MANY-COORDS}}$ is non-malleable. The proof is very similar to the proof in Section 2.4, and so we will follow the same outline, and only delve into details when the two proofs differ.

Modifying the Extractor E. Recall that the proof in Section 2.4 assumed that a MIM adversary M successfully executes a mauling attack on $\langle \mathrm{C}, \mathrm{R} \rangle$ with probability at least $2p$, for non-negligible quantity $p = p(\lambda)$. It then described an extractor E (Figure 2.3), and proved that when given a completed transcript $\mathbb{T}$, E successfully outputs $\tilde{m}$, M's commitment in the right execution, with probability at least $1 - p$. This ensured that M's mauling attack, and the E's extraction would both succeed with probability at least p, which breaks the hiding of Com.

We use the same high level approach, and also the same extractor in our proof that $\langle \mathrm{C}, \mathrm{R} \rangle_{\mathsf{MANY-COORDS}}$ is non-malleable modified to account for the fact that $\langle \mathrm{C}, \mathrm{R} \rangle_{\mathsf{MANY-COORDS}}$ uses quadratic polynomials with secret shares of secret vector $\tilde{\mathbf{m}} \in \mathbb{Z}_q^\ell$ as constant terms instead of linear polynomials with the same constant term. Explicitly, fix $N = \mathrm{poly}(\lambda)$ such that $N = \omega\big(\lambda^9 n^{44}/p^{52}\big)$. For $j \in [N]$:

- E chooses two new queries $\tilde{\boldsymbol{\beta}} = \tilde{\boldsymbol{\beta}}_j, \tilde{\boldsymbol{\gamma}} = \tilde{\boldsymbol{\gamma}}_j$ and rewinds M twice to the beginning of the right execution's query phase sending $\tilde{\boldsymbol{\beta}}$ and $\tilde{\boldsymbol{\gamma}}$, receiving $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$.

- E sends random responses $\mathbf{b}$ and $\mathbf{c}$, receiving right responses $\tilde{\mathbf{b}}$ and $\tilde{\mathbf{c}}$.

- E keeps track of a set $S \subset [n]$ initialized to $[n]$. For all $i_1, i_2 \in S$ such that $i_1 \sim i_2$, E checks whether the points

$$\big\{(\tilde{\alpha}_{i_1}, \tilde{a}_{i_1}), (\tilde{\alpha}_{i_2}, \tilde{a}_{i_2}), (\tilde{\beta}_{i_1}, \tilde{b}_{i_1}), (\tilde{\gamma}_{i_1}, \tilde{c}_{i_1})\big\} \text{ OR } \big\{(\tilde{\alpha}_{i_1}, \tilde{a}_{i_1}), (\tilde{\alpha}_{i_2}, \tilde{a}_{i_2}), (\tilde{\beta}_{i_2}, \tilde{b}_{i_2}), (\tilde{\gamma}_{i_2}, \tilde{c}_{i_2})\big\}$$

  lie on the same quadratic. If so, E records $(i_1, i_2, [\tilde{\mathbf{m}}]_{(i_1,i_2)})$, where $[\tilde{\mathbf{m}}]_{(i_1,i_2)}$ is the constant term of this quadratic. E also removes i_1 and i_2 from S.

- As soon as E has recorded ℓ constant terms, it reconstructs and outputs $\tilde{\mathbf{m}}$ and halts.

  The non-malleability of $\langle \mathrm{C}, \mathrm{R} \rangle_{\mathsf{MANY-COORDS}}$ now follows from proving that

$$\Pr_{\mathbb{T} \in \mathsf{ACC}}\big(\mathrm{E}(\mathbb{T}) \neq \tilde{\mathbf{m}}\big) \leq p.$$

Modifying INT, the Set of Interesting Transcripts. Our proof in Section 2.4 then proceeded to define the set of interesting transcripts, $\mathsf{INT} = \mathsf{USEFUL} \setminus \mathsf{EXT}$. These are the transcripts on which 1) if rewound, M will not simply "always abort" and 2) extraction fails. Our definition of USEFUL from Definition 2.6 remains the same, but our definition of EXT from Definition 2.5 changes, resulting in a new set $\mathsf{INT}_{\mathsf{MANY\text{–}COORDS}}$. The reason for the change in EXT is that in order for extraction to be successful, E must extract from many coordinates rather than just one.

Explicitly, we keep the definition of EXT^i and redefine

$$\mathsf{EXT}_{\mathsf{MANY\text{–}COORDS}} = \big\{\mathbb{T} \in \mathsf{ACC} : {}^{\#}\{i : \mathbf{Com} \in \mathsf{EXT}^i\} \geq \ell\big\}.$$

The proof that extraction succeeds whp if $\mathbb{T} \in \mathsf{EXT}_{\mathsf{MANY\text{–}COORDS}}$ requires, just as in the original proof, proving that E does not output the incorrect value $\tilde{\mathbf{m}}' \neq \tilde{\mathbf{m}}$ except with negligible probability. This involves showing that M does not answer "incorrectly but co-

quadratically" except with negligible probability, which is proved just as Claim 2.1. Now, $\Pr_{\mathbb{T}\in\mathsf{ACC}}\big(\mathrm{E}(\mathbb{T}) \neq \tilde{\mathbf{m}}\big) \leq p$ follows from

$$\Pr_{\mathbb{T}\in\mathsf{ACC}}\big(\mathbb{T} \in \mathsf{INT}_{\mathsf{MANY\text{–}COORDS}}\big) \leq \delta' p$$

where $\mathsf{INT}_{\mathsf{MANY\text{–}COORDS}} = \mathsf{USEFUL} \setminus \mathsf{EXT}_{\mathsf{MANY\text{–}COORDS}}$. If we think of $\mathsf{INT}_{\mathsf{MANY\text{–}COORDS}}$ in terms of M's behavior, it is the set of completed transcripts such that if M is rewound and asked a new query:

- M answers correctly with non-negligible probability if given correct responses to its own queries;
- M answers correctly in at most $\ell - 1$ coordinates if given random responses to its own queries.

So, at least behaviorally, $\mathsf{INT}_{\mathsf{MANY\text{–}COORDS}}$ can be thought of as containing INT (even though technically they are sets of transcripts of different protocols, so no formal notion of containment is possible).

Analyzing Dependencies. We now turn to the task of examining the dependencies between the left and right queries. We have already done most of the work required to complete the proof, either in Section 2.4 or Section 2.5.1.

Recall we defined IND to be the set of transcripts such that there exists a query $\tilde{\alpha}_i$ such that no $\alpha_{i'}$ is $\varepsilon-$dependent on $\tilde{\alpha}_i$ (in this case we said that $\tilde{\alpha}_i$ is $\varepsilon-$*independent*). We then proved Claim 2.9 by proving essentially that if $\tilde{\alpha}_i$ is $\varepsilon-$independent, then $\mathbf{Com} \in \mathsf{EXT}^i$. This implied that $\mathbb{T} \in \mathsf{EXT}$, which allowed us to bound $\Pr_{\mathbb{T}\in\mathsf{ACC}}\big(\mathbb{T} \in \mathsf{IND} \setminus \mathsf{EXT}\big)$. In our case, we still have that if $\tilde{\alpha}_i$ is $\varepsilon-$independent then $\mathbf{Com} \in \mathsf{EXT}^i$, but this no longer implies

that $\mathbb{T} \in \mathsf{EXT}_{\mathsf{MANY\text{–}COORDS}}$. We therefore define

$$\mathsf{IND}_{\mathsf{MANY\text{–}COORDS}} = \big\{\mathbb{T} \in \mathsf{ACC} : {}^{\#}\{i : \tilde{\alpha}_i \text{ is } \varepsilon - \text{independent}\} \geq \ell\big\}.$$

This allows us to bound $\Pr_{\mathbb{T}\in\mathsf{ACC}}\big(\mathbb{T} \in \mathsf{IND}_{\mathsf{MANY\text{–}COORDS}} \setminus \mathsf{EXT}_{\mathsf{MANY\text{–}COORDS}}\big)$, analogously to Claim 2.9.

Recall that we defined $\mathsf{SUPER}-\mathsf{POLY}$ to be the set of transcripts such that the left query vector $\boldsymbol{\alpha}$ has a superpolynomial number of preimage right query vectors $\tilde{\boldsymbol{\alpha}}$. In Claim 2.8 we proved that interesting transcripts cannot be in $\mathsf{SUPER}-\mathsf{POLY}$ except with very small probability; otherwise M can be used to break hiding. We then defined sets of transcripts

- $\mathsf{1}-\mathsf{2} := \big\{\mathbb{T} \in \mathsf{ACC} : \exists\ (i_1, i_2, i') \text{ st } \alpha_{i'} \text{ is } \varepsilon' - \text{dependent on both } \tilde{\alpha}_{i_1} \text{ and } \tilde{\alpha}_{i_2}\big\}$
- $\mathsf{UNBAL} := \big\{\mathbb{T} \in \mathsf{ACC} : \exists\ i' > i \text{ st } \alpha_{i'} \text{ is } \varepsilon' - \text{dependent on } \tilde{\alpha}_i\big\}$

and proved that in Claims 2.5 and 2.6 that if interesting transcripts are likely to be in either set, then they are also likely to be in $\mathsf{SUPER}-\mathsf{POLY}$, which breaks the hiding of $\langle \mathrm{C}, \mathrm{R}\rangle$ via Claim 2.8. It is easy to see that Claim 2.5, Claim 2.6 and Claim 2.8 all carry over identically to $\langle \mathrm{C}, \mathrm{R}\rangle_{\mathsf{MANY}-\mathsf{COORDS}}$, since these proofs only rely on counting the sizes of various sets of queries, and not on the specific differences between $\langle \mathrm{C}, \mathrm{R}\rangle$ and $\langle \mathrm{C}, \mathrm{R}\rangle_{\mathsf{MANY}-\mathsf{COORDS}}$.

Now, define the set $S(\mathbb{T})$ to be the set of i such that $\tilde{\alpha}_i$ is not $\varepsilon-$independent. By definition, for these $\tilde{\alpha}_i$ there exists an $\alpha_{i'}$ such that $\alpha_{i'}$ is $\varepsilon-$dependent on $\tilde{\alpha}_i$. By our discussion of $\mathsf{IND}_{\mathsf{MANY\text{–}COORDS}}$, we may assume that $\big|S(\mathbb{T})\big| \geq n - \ell + 1$. As no two $\tilde{\alpha}_{i_1}$ and $\tilde{\alpha}_{i_2}$ can be $\varepsilon-$dependent on the same $\alpha_{i'}$, we see that for at least $n - 2\ell + 2$ values of i, $\tilde{\alpha}_i$ is $\varepsilon-$dependent on a unique $\alpha_{i'}$. This means that there are at least $\big(n/2\big) - 2\ell + 2$ pairs (i_1, i_2) such that $i_1 \sim i_2$ and $\tilde{\alpha}_{i_1}$ and $\tilde{\alpha}_{i_2}$ are dependent on unique $\alpha_{i'_1}$ and $\alpha_{i'_2}$. By Claim 2.11, it must be that $i'_1 \sim i'_2$, which means that there are at least $\big(n/2\big) - 2\ell + 2$ related queries

on the right with a unique pair of related queries on the left that is $\varepsilon-$dependent on it. By Claim 2.6, it must be that each of these pairs are the same coordinates. In other words, it must be that there are at least $\big(n/2\big) - 2\ell + 2$ pairs (i_1, i_2) such that $i_1 \sim i_2$ and such that α_{i_1} (resp. α_{i_2}) is $\varepsilon-$dependent on $\tilde{\alpha}_{i_1}$ (resp. $\tilde{\alpha}_{i_2}$). Finally, this means that there are at least $n - 4\ell + 4$ values of i such that α_i is $\varepsilon-$dependent on $\tilde{\alpha}_i$. Just as in Claim 2.7, we can show that this means that $\mathbb{T} \in \mathsf{SUPER-POLY}$ with sufficient probability to break the hiding of $\langle \mathrm{C}, \mathrm{R} \rangle_{\mathsf{MANY-COORDS}}$.

This completes the walkthrough of the proof of non-malleability of $\langle \mathrm{C}, \mathrm{R} \rangle_{\mathsf{MANY-COORDS}}$. Explicitly, $\Pr_{\mathbb{T} \in \mathsf{ACC}}\big(\mathbb{T} \in \mathsf{INT}_{\mathsf{MANY\text{-}COORDS}}\big) \leq \delta' p$ follows from the following assertions, whose proofs are analogous to those of the claims they are next to.

- **Claim 2.9:** $\Pr_{\mathbb{T} \in \mathsf{ACC}}\big(\mathbb{T} \in \mathsf{INT}_{\mathsf{MANY\text{-}COORDS}} \cap \mathsf{IND}_{\mathsf{MANY\text{-}COORDS}}\big) \leq \frac{\delta' p}{6}$;
- **Claim 2.5 via Claim 2.8:** $\Pr_{\mathbb{T} \in \mathsf{ACC}}\big(\mathbb{T} \in \mathsf{INT}_{\mathsf{MANY\text{-}COORDS}} \cap \mathsf{1-2}\big) \leq \frac{\delta' p}{6}$;
- **Claim 2.6 via Claim 2.8:** $\Pr_{\mathbb{T} \in \mathsf{ACC}}\big(\mathbb{T} \in \mathsf{INT}_{\mathsf{MANY\text{-}COORDS}} \cap \mathsf{UNBAL}\big) \leq \frac{\delta' p}{6}$;
- **Claim 2.10:** $\Pr_{\mathbb{T} \in \mathsf{ACC}}\big(\mathbb{T} \in \mathsf{INT}_{\mathsf{MANY\text{-}COORDS}} \cap \mathsf{GEN-EXT}\big) \leq \frac{\delta' p}{6}$;
- **Claim 2.11:** $\Pr_{\mathbb{T} \in \mathsf{ACC}}\big(\mathbb{T} \in (\mathsf{INT}_{\mathsf{MANY\text{-}COORDS}} \cap \mathsf{MIXED}) \setminus \mathsf{GEN-EXT}\big) \leq \frac{\delta' p}{6}$;
- **Claim 2.7 via Claim 2.8:**

$$\Pr_{\mathbb{T} \in \mathsf{ACC}}\big(\mathbb{T} \in \mathsf{INT}_{\mathsf{MANY\text{-}COORDS}} \setminus (\mathsf{IND}_{\mathsf{MANY\text{-}COORDS}} \cup \mathsf{1-2} \cup \mathsf{UNBAL} \cup \mathsf{MIXED})\big) \leq \frac{\delta' p}{6}.$$

2.6 Non-Malleability in 4-Rounds

In this section we show how to squeeze our non-malleable protocol $\langle \mathrm{C}, \mathrm{R} \rangle_{\mathsf{MANY-COORDS}}$ into 4 rounds. In the new protocol, the zero-knowledge messages are lifted up and sent

together with the challenge-response messages. Consider the 4-round Feige-Shamir zero-knowledge protocol [FS90] which consists of having the verifier prove a hard statement using 3-rounds Witness indistinguishable Argument of Knowledge ($\mathcal{WIAOK}$) while the prover also uses 3-rounds $\mathcal{WIAOK}$ to prove another hard statement (that involves potential knowledge of verifier's trapdoor). We carefully parallelize the Feige-Shamir protocol with our 4-round basic component of $\langle \mathrm{C}, \mathrm{R} \rangle_{\mathsf{MANY-COORDS}}$ and obtain the first 4-round non-malleable commitment scheme. Our 4-round commitment scheme $\langle \mathrm{C}, \mathrm{R} \rangle_{\mathsf{OPT}}$ appears in Figure 2.5. We denote by π_1 and π_2 the $\mathcal{WIAOK}$ of the verifier and prover respectively. In π_1 we use two One-Way Functions and refer by L_1 and L_2 to their corresponding (hard on average) languages.

Public Parameters: Tags $t_1, \ldots, t_n$, large prime q such that $q > 2^{t_i}$ for all i and $\ell = \Omega(n)$.

Commiter's Private Input: Message vector $\mathbf{m} \in \mathbb{F}_q^\ell$ to be committed to.

1. $\mathrm{R} \to \mathrm{C}$: Sample random $x_1 \in L_1$ and $x_2 \in L_2$ and send the first message of π_1 proving either $x_1 \in L_1$ or $x_2 \in L_2$ along with the first message σ of the Naor commitment scheme.
2. $\mathrm{C} \to \mathrm{R}$: Send the challenge message of π_1 along with the first message of π_2 and the commitment vector $\mathbf{Com}$ as done in Step 1 of $\langle \mathrm{C}, \mathrm{R} \rangle_{\mathsf{MANY-COORDS}}$. (The statement of π_2 is determined in the last step).
3. $\mathrm{R} \to \mathrm{C}$: Send the last message of π_1 along with challenge message of π_2 and the challenge vector $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)$ as done in Step 2 of $\langle \mathrm{C}, \mathrm{R} \rangle_{\mathsf{MANY-COORDS}}$.
4. $\mathrm{C} \to \mathrm{R}$: Send the evaluations $\{f_i(\alpha_i), f_i(\alpha_{n-i})\}_{i \in [n/2]}$ as in Step 3 of $\langle \mathrm{C}, \mathrm{R} \rangle_{\mathsf{MANY-COORDS}}$ along with the last message of π_2 proving the following statement:
   - message 1 and 3 are correct; OR
   - $x_1 \in L_1$; OR
   - $x_2 \in L_2$.

Figure 2.5: : 4-round non-malleable commitment scheme $\langle \mathrm{C}, \mathrm{R} \rangle_{\mathsf{OPT}}$.

Proposition 2.2. $\langle \mathrm{C}, \mathrm{R} \rangle_{\mathsf{OPT}}$ *is a four round statistically binding, non-malleable commitment*

scheme.

Proof Sketch. Statistical binding follows immediately from the statistical binding property of Com. To prove computational hiding, we follow the proof of Proposition 2.1.

The proof that $\langle \mathrm{C}, \mathrm{R} \rangle_{\mathsf{OPT}}$ is non-malleable against a synchronizing adversary follows exactly the same argument as the proof that $\langle \mathrm{C}, \mathrm{R} \rangle_{\mathsf{OPT}}$ is non-malleable except that now when E is rewinding and answering M's queries randomly in the left interaction, it must complete the proof π_2 using one of the trapdoor statements. However, this is not problem. It just means that before E can start rewinding M back to the beginning of the right execution's query phase and trying to extract $\tilde{\mathbf{m}}$, it must first rewind M back to the left execution's commit phase and extract M's trapdoor witness.

In order to prove non-malleability against a non-synchronizing M, we simply enumerate and check all of the other possibilities for M's scheduling. As $\langle \mathrm{C}, \mathrm{R} \rangle_{\mathsf{OPT}}$ has only four rounds, this is a very manageable task. Essentially extraction is trivial from an M who uses any scheduling other than the synchronizing one. □

By slightly modifying our new commitment scheme $\langle \mathrm{C}, \mathrm{R} \rangle_{\mathsf{OPT}}$, we obtain a simple 4-round non-malleable zero knowledge argument (P, V) for any language L in $\mathcal{NP}$. Detailed description of our (P, V) protocol appears in Figure Figure 2.6.

Proposition 2.3. *The protocol (P, V) is a non-malleable zero knowledge argument of knowledge for any language $L \in \mathcal{NP}$.*

Proof Sketch. The proof of non-malleability is similar to the non-malleability of $\langle \mathrm{C}, \mathrm{R} \rangle_{\mathsf{OPT}}$. The zero-knowledge property follows by the ZK of [FS90] and the hiding of $\langle \mathrm{C}, \mathrm{R} \rangle_{\mathsf{OPT}}$. Lastly, the AOK property can be derived by constructing an extractor that behaves as a honest verifier and extracts the witness by rewinding and sending different challenges. □

Tags: Let the tag for the interaction be $t = t_1, \ldots, t_n$ in error corrected form.

Common input: $x \in L$.

Input to the prover: A witness w for $x \in L$.

1. $V \to P$: Sample random $x_1 \in L_1$ and $x_2 \in L_2$ and send the first message of π_1 protocol proving either $x_1 \in L_1$ or $x_2 \in L_2$ along with the first message σ of the Naor commitment scheme.

2. $P \to V$: Send the challenge message of π_1 along with the first message of the π_2 and the commitment vector **Com** with w being the free coefficient as done in Step 1 of $\langle \mathrm{C}, \mathrm{R} \rangle_{\mathsf{OPT}}$. (The statement of π_2 is determined in the last step).

3. $V \to P$: Send the last message of π_1 along with challenge message of π_2 and the challenge vector $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)$ as done in Step 2 of $\langle \mathrm{C}, \mathrm{R} \rangle_{\mathsf{OPT}}$.

4. $P \to V$: Send the evaluations $\{f_i(\alpha_i), f_i(\alpha_{n-i})\}_{i \in [n/2]}$ as in Step 3 of $\langle \mathrm{C}, \mathrm{R} \rangle_{\mathsf{OPT}}$ along with the last message of π_2 proving the following statement:

   - either knowledge of w together with correctness of $\{f_i(\alpha_i), f_i(\alpha_{n-i})\}_{i \in [n/2]}$.
   - or $x_1 \in L_1$.
   - or $x_2 \in L_2$.

Figure 2.6: 4-round non malleable zero-knowledge argument (P, V).

Chapter 3

Concurrent Zero-Knowledge in the Bounded Player Model

3.1 Concurrent Zero-Knowledge

The original definition of zero knowledge is relevant to the "stand-alone" setting where security is only required to hold if the protocol is run in isolation. If one were to run many executions of such a protocol concurrently, the stand-alone security guarantees would disappear. Given the interconnectivity present in today's world thanks to the internet and other network environments, it seems unreasonable to demand that only one execution of a protocol be run at once. With this in mind, Dwork, Naor and Sahai [DNS98] initiated the study of *concurrent zero-knowledge* (cZK) proofs that remain secure even if several instances of the protocol are executed concurrently under the control of an adversarial verifier. Subsequent to their work, cZK has been the subject of extensive research, with a large body of work devoted to studying its round-complexity. In the standard model, the round-complexity of cZK was improved from polynomial to slightly super-logarithmic in a sequence of works [RK99, KP01, PRS02]. In particular, the $\tilde{\mathcal{O}}(\log \lambda)$-round construction of [PRS02] nearly matches the lower bound of $\tilde{\Omega}(\log k)$ with respect to black-box simulation [CKPR01] (see

also [KPR98, Ros00]).

Despite a decade of research, the $\tilde{O}(\log k)$-round construction of [PRS02] is still the most round-efficient cZK protocol known. Moreover, the lower bound of [CKPR01] tells us that in order to improve drastically upon [PRS02] one must use non-blackbox simulation techniques, which would represent quite a breakthrough.

3.1.1 Round-efficient cZK in relaxed models.

While the round-complexity of cZK in the standard model still remains an intriguing open question, a long line of work has been dedicated towards constructing round-efficient cZK in various relaxations of the standard model. Below, we briefly discuss the state of the art on some of these models.

Bounded Concurrency Model. An interesting relaxation of the standard model is the bounded-concurrency model [Bar01], where an *a priori* bound is assumed over the number of sessions that will ever take place (in particular, this bound is known to the protocol designer). It is known how to realize constant-round cZK [Bar01] in this model (sometimes called bounded cZK), and also constant-round bounded-concurrent secure two-party and multi-party computation [Lin03a, PR03, Pas04a].

Bare Public Key and Related Preprocessing Models. The zero-knowledge preprocessing model was proposed in [KMO89] in the stand-alone setting and in [CO99] in the context of cZK. In [CO99], interaction is needed between all the involved players in a preprocessing phase. Then, after a synchronization barrier is passed, the preprocessing is over and actual proofs begin. Interactions in each phase can take place concurrently, but the two phases can not overlap in time. An improved model called the "Bare Public-Key" (BPK) model was later proposed in [CGGM00] where the preprocessing is required to be

non-interactive. The name is derived from the observation that the non-interactive messages played in the preprocessing can be considered as public announcements of public keys. In this model it is known how to obtain constant-round black-box cZK under standard assumptions [SV12].

Superpolynomial Time Simulation Model. In the SPS model [Pas03], the zero-knowledge simulator is allowed to run in super-polynomial time. This relaxation has yielded not only constant-round cZK [Pas03], but also concurrent-secure computation [LPV09, CLP10, GGJS12]. This is in contrast to the standard model, where concurrent-secure computation is known to be impossible to achieve [Lin04, Lin03b, CKL03, CF01].

Timing Model In the timing model of [DNS98], where an upper-bound is assumed on the delivery time needed of a message (and therefore the adversary is assumed to have only limited control of the communication network), it is known how to construct constant-round cZK [DNS98, Gol02, PTV10], as well as cMPC [KLP05].

Common Reference String Model. In the CRS model of [BSMP91], where all parties receive the same honestly generated random string as input, it is known how to construct constant round cZK and even UC secure MPC [SCO+01, CLOS02] which is impossible in the plain model [Can01].

The Need for a New Model. While the above models are appealing both in terms of naturality and in terms of feasibility results yielded, they provide limited insight towards resolving the round complexity of cZK in the plain model. Indeed, any model where cMPC or constant round black-box cZK is possible (such as all of the ones mentioned above) is fundamentally different from the plain model where these are known to be impossible [Lin04, CKPR01].

3.2 The Bounded Player Model

We put forth the *bounded player* (BP) *model* for secure computation. In this model we assume that there is an *a priori* polynomial upper bound on the number of players who will ever participate in the protocol. The BP model lies between the bounded concurrency model and the plain model since though there is a bound on the number of players, there is no bound on the number of executions each one may enact. Interestingly, we prove in Section **??** that the the impossibility result of [Lin04] for cMPC in the plain model extends as well to the BP model. This proves that the BP model is strictly closer to the plain model than the bounded concurrency model.

Though the BP model bears resemblance to the BPK model described above, the lack of a synchronization barrier in our model turns out to be very important and prevents techniques relevant to the BPK model from applying. Indeed, the lower bound of [CKPR01] in the plain model extends immediately to the BP model, while it is known how to construct constant round black-box cZK in the BPK model [SV12].

3.2.1 cZK in the BP Model

In Sections 3.3 and 3.4 we give two protocols for cZK in the BP model culminating with the following main theorem.

Theorem 3.1. *Assume that CRHF families exist. Then there exists a constant round cZK argument system in the BP model.*

These protocols come from [GJO$^+$13a] and [GJO$^+$13b], respectively. Our first protocol has slightly superconstant (any $\omega(1)$) round complexity, and makes slightly stronger assumptions. We describe them both however, because they represent different and interesting solutions to the same general problem one is confronted with when trying to construct cZK

in the BP model. This problem is discussed informally below.

3.2.2 Techniques

When faced with the task of constructing cZK in the BP model, one immediately gravitates toward Barak's non-blackbox techniques since [CKPR01]'s impossibility of constant round blackbox cZK in the plain model carries over to the BP model. Now, a natural approach to leverage the bound on the number of players is to associate with each verifier V_i a public key pk_i and then design an FLS-style protocol [FLS90] that allows the ZK simulator to extract, in a non-black-box manner, the secret key sk_i of the verifier and then use it as a "trapdoor" for "easy" simulation. The key intuition is that once the simulator extracts the secret key sk_i of a verifier V_i, it can perform easy simulation of *all* the sessions associated with V_i. Then, since the total number of verifiers is bounded, the simulator will need to perform non-black-box extraction only an *a priori* bounded number of times (once for each verifier), which can be handled in a manner similar to the setting of bounded concurrency [Bar01].

The above idea has some merit. Indeed, if we can ensure that $\mathcal{S}$ can extract every public key using a bounded polynomial number of non-blackbox simulations, then the *a priori* bound on the number of players will land us in the setting of bounded concurrency and we can argue using the techniques of [Bar01]. However, one quickly realizes that this idea runs into the same problem one encounters when trying to extend [Bar01]'s bounded cZK protocol to the setting of unbounded concurrency. The issue is that as soon as our simulator attempts to extract sk_i from V_i in one protocol execution, V_i can switch and start a new session. When $\mathcal{S}$ tries to extract in this new session, V_i can switch and start another one, and so on. This type of unruly behavior from V_i would cause the running time of $\mathcal{S}$ to blow up exponentially.

Our two protocols have different ways of dealing with the above problem. The protocol

of Section 3.3 solves this problem by introducing many extraction opportunities in each protocol. Now V_i has to guess in which slot, $\mathcal{S}$ is trying to extract from. If V_i guesses wrong and $\mathcal{S}$ is able to extract from a slot without V_i switching to a new execution, then $\mathcal{S}$ will have learned sk_i, the trapdoor in all protocol executions. Proof of security in this protocol reduces to a "balls and bins" style probability computation. On the down side, a superconstant number of rounds seem inherent to this proof technique.

The protocol of Section 3.4 solves the problem differently. Instead of inserting many opportunities to extract the trapdoor, it extracts the trapdoor in pieces in such a way so that if V_i switches and opens a new session mid-extraction, $\mathcal{S}$ will be able to use the partial extraction work it has already done in the next session. In this protocol, V_i will get no benefit from switching sessions, as $\mathcal{S}$ will be able to recycle the work it has already done, preventing its running time from blowing up.

3.2.3 Formal Definitions

Let $N = \mathrm{poly}(\lambda)$ be a bound on the total number of players that can engage in concurrent executions of a protocol at any time. We assume that each player P_i ($i \in [N]$) has an associated unique identity id_i, and that there is an established mechanism to enforce that party P_i uses the same identity id_i in each protocol execution that it participates in. Note, however, that such identities do not have to be established in advance. In particular, new players can join the system with their own (new) identities, as long as the number of players does not exceed N. We stress that there is not bound on the number of protocol executions that can be started by each party.

The bounded player model is formalized by means of a functionality F_{bp}^N that registers the identities of the player in the system. Specifically, a player P_i that wishes to participate in protocol executions can, at any time, register an identity id_i with the functionality F_{bp}^N. The

registration functionality does not perform any checks on the identities that are registered, except that each party P_i can register at most one identity id_i, and that the total number of identity registrations are bounded by N. In other words, F_{bp}^N refuses to register any new identities once N identities have already been registered. The functionality F_{bp}^N is formally defined in Figure 3.1.

Functionality F_{bp}^N

F_{bp}^N initializes a variable *count* to 0 and proceeds as follows.

- **Register commands:** Upon receiving a message $(\textbf{register}, sid, \mathsf{id}_i)$ from some party P_i, the functionality checks that no pair (P_i, id_i') is already recorded and that $count < N$. If this is the case, it records the pair (P_i, id_i) and sets $count = count + 1$. Other wise, it ignores the received message.
- **Retrieve commands:** Upon receiving a message $(\mathsf{retrieve}, sid, P_i)$ from some party P_j or the adversary A, the functionality checks if some pair (P_i, id_i) is recorded. If this the case, it sends $(sid, P_i, \mathsf{id}_i)$ to P_j (or A). Otherwise, it returns $(sid, P_i, \perp)$.

Figure 3.1: The Bounded Player Functionality F_{bp}^N.

In our constructions we will explicitly work in the setting where the identity of each party is a tuple (h, vk), where $h \leftarrow \mathcal{H}_\lambda$ is a hash function chosen from a family $\mathcal{H}_\lambda$ of collision resistant hash functions, and vk is a verification key for a signature scheme.

In order to define cZK in the BP model, we appropriately modify the random variables of Definition 1.2. Let $\langle P, V \rangle$ be an interactive argument system for a language $L \in \mathcal{NP}$. Define the random variables of $\mathbf{VIEW}^N_{\langle P(w), V^* \rangle(x)}$ and $\mathbf{VIEW}^N_{\mathcal{S}(V^*, x)}$ to be views of V^* when interacting with P and $\mathcal{S}$, respectively, in polynomially many concurrent executions of $\langle P, V \rangle$ where each execution is run on common input x, and when V^* interacts with P, P uses secret input w, a witness for $x \in L$. Furthermore, V^* may control at most N verifiers with distinct identities as per the functionality F_{bp}^N.

Definition 3.1 (cZK in the BP Model). *We say that* $\langle P, V \rangle$ *is* concurrent zero-knowledge in the bounded player model *if for all PPT adversarial verifiers* V^**, there exists a PPT simulator* $\mathcal{S}$ *such that*

$$\left\{\mathbf{VIEW}^N_{\langle P(w), V^* \rangle(x)}\right\} \approx_c \left\{\mathbf{VIEW}^N_{\mathcal{S}(V^*, x)}\right\}.$$

We remark that one must show that a candidate protocol $\langle P, V \rangle$ is *concurrently sound in the BP model*, as this notion is strictly stronger than standalone soundness in the BP model. This is unlike the plain model where concurrent soundness and standalone soundness are the same, and is proven using the techniques of [MR01] (who prove the same statement for the BPK model).

3.3 The $\omega(1)-$Round Protocol

3.3.1 Building Blocks

Perfectly Hiding Commitment Scheme. In our constructions, we will make use of a perfectly hiding string commitment scheme, denoted Com. For simplicity of exposition, we will make the simplifying assumption that Com is a non-interactive perfectly hiding commitment scheme (even though such a scheme cannot exist). In reality, Com would be taken to be a 2-round commitment scheme, which can be based on collections of claw-free permutations [GK96]. Unless stated otherwise, we will simply use the notation $\mathrm{Com}(x)$ to denote a commitment to a string x, and assume that the randomness (used to create the commitment) is implicit.

Perfect Witness Indistinguishable Argument of Knowledge. We will also make use of a perfect witness-indistinguishable argument of knowledge system for all of $\mathcal{NP}$ in our

construction. Such a scheme can be constructed, for example, by parallel repetition of the 3-round Blum's protocol for Graph Hamiltonicity [Blu86] instantiated with a perfectly hiding commitment scheme. We will denote such an argument system by $\langle P_{\mathsf{pWI}}, V_{\mathsf{pWI}} \rangle$.

Perfect Witness Indistinguishable Universal Argument. In our construction, we will use a perfect witness-indistinguishable universal argument system, denoted $\langle P_{\mathsf{pUA}}, V_{\mathsf{pUA}} \rangle$. Such an argument system can be constructed generically from a (computational) witness-indistinguishable universal argument pUA by using techniques of [PR05a, PR05b]. Specifically, in protocol $\langle P_{\mathsf{pUA}}, V_{\mathsf{pUA}} \rangle$, the prover P and verifier V first engage in an execution of pUA, where instead of sending its messages in the clear, P commits to each message using a perfectly hiding commitment scheme. Finally, P and V engage in an execution of a perfect zero knowledge argument of knowledge where P proves that the "decommitted" transcript of pUA is "accepting". The resulting protocol is still a "weak" argument of knowledge.

Perfect Bounded Concurrent Zero-Knowledge. Our cZK argument crucially uses as a building block, a variant of the bounded cZK argument of Barak [Bar01]. Similarly to [PR05a], we modify the protocol appropriately such that it is *perfect* bounded cZK. Specifically, instead of a statistically binding commitment scheme, we will use a perfectly hiding commitment scheme. Instead of a computationally witness-indistinguishable universal argument (UARG), we will use a perfect witness indistinguishable UARG, denoted $\langle P_{\mathsf{pUA}}, V_{\mathsf{pUA}} \rangle$. The perfect bounded cZK scheme is denote $\langle P_{\mathrm{pZK}}, V_{\mathrm{pZK}} \rangle_N$ and showin in Figure 3.3. It is defined using a length parameter $\ell(N)$, a function of N, the bound on the number of verifiers in the system.

Resettable Witness Indistinguishable Proof System. We will further use a *resettable* witness-indistinguishable proof system [CGGM00] for all of $\mathcal{NP}$. Informally speaking, a proof system is resettable witness indistinguishable if it remains witness indistinguishable

even against an adversarial verifier who can *reset* the prover and receive multiple proofs such that the prover uses the *same* random tape in each of the interactions. While the focus of this work is not on achieving security against reset attacks, such a proof system turns out to be useful when arguing concurrent soundness of our protocol (where our proof relies on a rewinding based argument). We will denote such a proof system by $\langle P_{\mathsf{rWI}}, V_{\mathsf{rWI}} \rangle$. It follows from [CGGM00] that such a proof system can be based on perfectly hiding commitments.

Dense Cryptosystems [SP92]. We will use a semantically secure public-key encryption scheme, denoted as $(\mathbf{Gen}, \mathbf{Enc}, \mathbf{Dec})$ that supports *oblivious* key generation (i.e., it should be possible to sample a public key without knowing the corresponding secret key). More precisely, there exists a deterministic algorithm **OGen** that takes as input the security parameter 1^λ and a sufficiently long random string σ and outputs a public key $pk \leftarrow \mathbf{OGen}(1^\lambda, \sigma)$, where pk is perfectly indistinguishable from a public key chosen by the normal key generation algorithm **Gen**. For simplicity of exposition, we will assume that the **OGen** algorithm simply outputs the input randomness σ as the public key. Such schemes can be based on a variety of number-theoretic assumptions such as DDH [SP92].

3.3.2 The Protocol

Relation R_{sim}. We first recall a slight variant of Barak's [Bar01] $\mathbf{NTIME}(T(\lambda))$ relation R_{sim}, as used previously in [PR05a]. Let $T : \mathbb{N} \to \mathbb{N}$ be a "nice" function that satisfies $T(\lambda) = \lambda^{\omega(1)}$. Let $\{\mathcal{H}_\lambda\}_\lambda$ be a family of collision-resistant hash functions where a function $h \in \mathcal{H}_\lambda$ maps $\{0,1\}^*$ to $\{0,1\}^\lambda$, and let Com be a perfectly hiding commitment scheme for strings of length λ, where for any $\alpha \in \{0,1\}^\lambda$, the length of $\mathrm{Com}(\alpha)$ is upper bounded by 2λ. The relation R_{sim} is described in Figure 3.2.

We remark as in [BG02, Pas04a, PR05b, PR05a] that Figure 3.2 is slightly oversimplified and will only be secure if $\{\mathcal{H}_\lambda\}_\lambda$ is collision-resistant against "slightly" super-polynomial

Instance: A triplet $\langle h, c, r \rangle \in \mathcal{H}_\lambda \times \{0,1\}^\lambda \times \{0,1\}^{\mathrm{poly}(\lambda)}$.
Witness: A program $\Pi \in \{0,1\}^*$, a string $y \in \{0,1\}^*$ and a string $s \in \{0,1\}^{\mathrm{poly}(\lambda)}$.
Relation: $R_{\mathsf{sim}}(\langle h, c, r \rangle, \langle \Pi, y, s \rangle) = 1$ if and only if:

1. $|y| \leq |r| - \lambda$.
2. $c = \mathrm{Com}(h(\Pi); s)$.
3. $\Pi(y) = r$ within $T(\lambda)$ steps.

Figure 3.2: R_{sim} - A variant of Barak's relation [PR05a]

sized circuits. For simplicity of exposition, in this manuscript, we will work with this assumption. We stress, however, that this assumption can be relaxed by using a "good" error-correcting code ECC (with constant distance and polynomial-time encoding and decoding procedures), and replacing the condition $c = \mathrm{Com}(h(\Pi); s)$ with $c = \mathrm{Com}(\mathsf{ECC}(h(\Pi)); s)$.

We are now ready to present our concurrent zero knowledge protocol, denoted $\langle P, V \rangle$. Let P and V denote the prover and verifier respectively. Let N denote the bound on the number of verifiers present in the system. Let f_{owf} denote a one-way function, and $(\mathbf{Gen}, \mathbf{Enc}, \mathbf{Dec})$ denote a dense public key encryption scheme. Let $\langle P_{\mathsf{pB}}, V_{\mathsf{pB}} \rangle_N$ denote the perfect zero-knowledge argument system as described above. Further, let $\langle P_{\mathsf{pWI}}, V_{\mathsf{pWI}} \rangle$ denote a perfect witness indistinguishable argument of knowledge, and let $\langle P_{\mathsf{rWI}}, V_{\mathsf{rWI}} \rangle$ denote a resettable witness indistinguishable proof system.

The protocol $\langle P, V \rangle$ is described in Figure 3.4. For our purposes, we set the length parameter $\ell(N) = n^3 \cdot N \cdot P(\lambda)$, where $P(\lambda)$ is a polynomial upper bound on the total length of the prover messages in the protocol plus the length of the secret key of the verifier.

The completeness property of $\langle P, V \rangle$ follows immediately from the construction.

Parameters: Security parameter λ, length parameter $\ell(N)$.
Common Input: $x \in \{0,1\}^{\text{poly}(n)}$.
Private Input to P: A witness w such that $R_L(x,w) = 1$.

Stage 1 (Preamble Phase):

$V \to P$: Send $h \xleftarrow{\text{R}} \mathcal{H}_\lambda$.

$P \to V$: Send $c = \text{Com}(0^\lambda)$.

$V \to P$: Send $r \xleftarrow{\text{R}} \{0,1\}^{\ell(N)}$.

Stage 2 (Proof Phase): A special-purpose UARG consisting of following steps:

$P \leftrightarrow V$: Encrypted UARG

$V \to P$: Send $\alpha \xleftarrow{\text{R}} \{0,1\}^\lambda$.

$P \to V$: Send $\widehat{\beta} = \text{Com}(0^\lambda)$.

$V \to P$: Send $\gamma \xleftarrow{\text{R}} \{0,1\}^\lambda$.

$P \to V$: Send $\widehat{\delta} = \text{Com}(0^\lambda)$.

$P \leftrightarrow V$: A perfect WIAOK $\langle P_{\mathsf{pWI}}, V_{\mathsf{pWI}} \rangle$ proving the OR of the following statements:

1. $\exists w \in \{0,1\}^{\text{poly}(|x|)}$ s.t. $R_L(x,w) = 1$.
2. $\exists \langle \beta, \delta, r_1, r_2 \rangle$ such that:
   - $\widehat{\beta} = \text{Com}(\beta; r_1)$.
   - $\widehat{\delta} = \text{Com}(\delta; r_2)$.
   - $(\alpha, \beta, \gamma, \delta)$ is an accepting transcript for perfect WI UARG $\langle P_{\mathsf{pUA}}, V_{\mathsf{pUA}} \rangle$ proving the statement: $\exists \langle \Pi, y, s \rangle$ s.t. $R_{\mathsf{sim}}(\langle h, c, r \rangle, \langle \Pi, y, s \rangle) = 1$.

Figure 3.3: Perfect Bounded cZK Protocol $\langle P_{\text{pZK}}, V_{\text{pZK}} \rangle_N$

3.3.3 Proof of Concurrent Soundness

In this section, we prove the soundness of our cZK protocol described in Section **??**. In fact, we will prove *concurrent soundness* of $\langle P, V \rangle$, i.e., we will show that a computationally-bounded adversarial prover who engages in multiple concurrent executions of $\langle P, V \rangle$ (where the scheduling across the sessions is controlled by the adversary) cannot prove a false statement in any of the executions, except with negligible probability. We note that similar to

Parameters: Security parameter λ, $N = N(\lambda)$, $t = \omega(1)$.
Common Input: $x \in \{0,1\}^{\mathrm{poly}(n)}$.
Private Input to P: A witness w s.t. $R_L(x,w) = 1$.
Private Input to V: A public key $pk = (y_0, y_1)$ and secret key $sk = (b, x_b)$ s.t. $b \stackrel{\mathrm{R}}{\leftarrow} \{0,1\}$, $y_b = f_{\mathsf{owf}}(x_b)$.

Stage 1 (Preamble Phase): Repeat the following steps t times.

- $V \to P$: Send $pk = (y_0, y_1)$.
- $P \to V$: Choose $\sigma_p \stackrel{\mathrm{R}}{\leftarrow} \{0,1\}^\lambda$ and send $c_p = \mathrm{Com}(\sigma_p)$.
- $V \to P$: Send $\sigma_v \stackrel{\mathrm{R}}{\leftarrow} \{0,1\}^\lambda$.
- $P \to V$: Send σ_p. Let $\sigma = \sigma_p \oplus \sigma_v$.
- $P \leftrightarrow V$: An execution of $\langle P_{\mathsf{pB}}, V_{\mathsf{pB}} \rangle_N$ to prove the following statement: $\exists s$ s.t. $c = \mathrm{Com}(\sigma_p; s)$.
- $V \to P$: Send $e_1 = \mathbf{Enc}_\sigma(x_b)$, $e_2 = \mathbf{Enc}_\sigma(x_b)$.
- $V \leftrightarrow P$: An execution of resettable WI $\langle P_{\mathsf{rWI}}, V_{\mathsf{rWI}} \rangle$ to prove the following statement: $\exists \langle i, b, x_b, s \rangle$ s.t. $e_i = \mathbf{Enc}_\sigma(x_b; s)$ and $y_b = f_{\mathsf{owf}}(x_b)$.

Stage 2 (Proof Phase):

- $P \leftrightarrow V$: An execution of perfect WIAOK $\langle P_{\mathsf{pWI}}, V_{\mathsf{pWI}} \rangle$ to prove the OR of the following statements:
  1. $\exists w \in \{0,1\}^{\mathrm{poly}(|x|)}$ s.t. $R_L(x,w) = 1$.
  2. $\exists \langle b, x_b \rangle$ s.t. $y_b = f_{\mathsf{owf}}(x_b)$.

Figure 3.4: Protocol $\langle P, V \rangle$−cZK in the BP Model

the bare-public key model [CGGM00], "stand-alone" soundness does not imply concurrent soundness in our model [MR01].

We first introduce some notation. Recall that in our protocol, in the execution of $\langle P_{\mathsf{pWI}}, V_{\mathsf{pWI}} \rangle$, the prover proves the OR of two statements. We will call a witness corresponding to the first (resp., second) part of the statement as *true* (resp., *trapdoor*) witness.

We first state a basic lemma related to the soundness of each instance of $\langle P_{\mathsf{pB}}, V_{\mathsf{pB}} \rangle_N$ across all executions of $\langle P, V \rangle$. Its proof is essentially identical to [Bar01], and so we omit

it.

Lemma 3.1. *Let $\hat{P}$ be any non-uniform probabilistic polynomial time adversarial prover that engages in any polynomial $m = m(\lambda)$ number of concurrent executions of $\langle P, V \rangle$ with N honest verifiers. Then, every instance of $\langle P_{\mathsf{pB}}, V_{\mathsf{pB}} \rangle_N$ across all executions of $\langle P, V \rangle$ is sound.*

Completing the Proof of Soundness. Suppose that there exists an i such that $\hat{P}$ succeeds in proving a false statement to the verifier in session i. Let S denote the set of all such i and let $v = |S|$.

Now, consider any $i \in S$. Note that it immediately follows from the (stand-alone) soundness of $\langle P_{\mathsf{pWI}}, V_{\mathsf{pWI}} \rangle$ that with probability at least $\frac{\epsilon}{v} - \mathbf{negl}(\lambda)$, $\hat{P}$ use a *trapdoor witness* in $\langle P_{\mathsf{pWI}}, V_{\mathsf{pWI}} \rangle$ in session i. Let $\tilde{V}$ denote the verifier in session i and let $pk = (y_0, y_1)$ denote the public key of $\tilde{V}$. Now, we run $\hat{P}$ such that in all protocol executions involving verifier $\tilde{V}$, we only use the secret key x_b corresponding to y_b, where $b \stackrel{\mathrm{R}}{\leftarrow} \{0, 1\}$. We now invoke the knowledge extractor E for $\langle P_{\mathsf{pWI}}, V_{\mathsf{pWI}} \rangle$ on $\hat{P}$ in session i. It follows from a standard argument (based on using "good" prefixes) that E successfully extracts a *trapdoor witness* with probability $p = p(\epsilon)$ where p is some polynomial. We now consider two cases:

1. With probability α, E outputs a witness $\hat{x}_{1-b}$ such that $y_{1-b} = f_{\mathsf{owf}}(\hat{x}_{1-b})$.
2. With probability $p - \alpha$, E outputs a witness $\hat{x}_b$ such that $y_b = f_{\mathsf{owf}}(\hat{x}_b)$.

If α is non-negligible, then it is immediate to see that we can build a polynomial-time inverter for one-way function f_{owf}. Specifically, the inverter I for f_{owf} works as follows. It runs the entire experiment with $\hat{P}$ in the same manner as above, except that y_{1-b} is taken from an external challenger for f_{owf}. When E outputs a value $\hat{x}_{1-b}$, I outputs it as the pre-image of y_{1-b} w.r.t. f_{owf}. Note that I succeeds with non-negligible probability α, which is a contradiction.

On the other hand, if $\alpha = \mathbf{negl}(\lambda)$, then we now focus on the second case. Let $\tilde{m}$ denote the total number of protocol sessions of $\langle P, V \rangle$ involving verifier $\tilde{V}$. Then, we have that with probability $p - \mathbf{negl}(\lambda)$, when $\tilde{V}$ (only) uses the secret key x_0 in all $\tilde{m}$ protocol sessions, the extractor E outputs a value $\hat{x}_0$, and similarly, when $\tilde{V}$ (only) uses x_1, E outputs a value $\hat{x}_1$, where $\hat{x}_b$ is such that $f_{\mathsf{owf}}(\hat{x}_b) = y_b$. Then, by a standard hybrid argument, there exists a session j (out of the $\tilde{m}$ sessions involving $\tilde{V}$) such that when $\tilde{V}$ (only) uses the secret key x_0 (resp., x_1) in session j, the extractor E outputs a value $\hat{x}_0$ (resp., $\hat{x}_1$), with probability at least $p' = \frac{p - \mathbf{negl}(n)}{\tilde{m}}$.[1] Let H_0 (resp., H_1) denote the hybrid experiment where $\tilde{V}$ uses x_0 (resp., x_1) in session j. Let $\hat{x}_b$ be the random variable that denotes the value that E extracts from $\hat{P}$ in experiment H_b.

We will now argue that $\hat{x}_0 \approx \hat{x}_1$, except with negligible probability, which is a contradiction to the above hypothesis, and thus concludes our proof. Let $\{e_1^q, e_2^q\}_{q=1}^t$ denote the $t = \omega(1)$ pairs of ciphertexts that $\tilde{V}$ sends to $\hat{P}$ in session j. Further, let $\{\langle P_{\mathsf{rWI}}, V_{\mathsf{rWI}} \rangle^q\}_{q=1}^t$ denote the t instances of $\langle P_{\mathsf{rWI}}, V_{\mathsf{rWI}} \rangle$ in session j. We consider three intermediate hybrid experiments H_{enc_1}, H_{wi} and H_{enc_2} described as follows.

HYBRID H_{enc_1}: This is the same as H_0, except that $\tilde{V}$ prepares each ciphertext $\{e_1^q\}_{q=1}^t$ to be an encryption of the secret key x_1. We now invoke the knowledge extractor E (for for $\langle P_{\mathsf{pWI}}, V_{\mathsf{pWI}} \rangle$) on $\hat{P}$ in $\langle P_{\mathsf{pWI}}, V_{\mathsf{pWI}} \rangle$ in session i. Let $\hat{x}_{\mathsf{enc}_1}$ be the random variable that denotes the value that E outputs.

We now claim that $\hat{x}_0 \approx \hat{x}_{\mathsf{enc}_1}$. Suppose that this is not the case. Then, by a standard hybrid argument, there exists $q \in [t]$ such that $\hat{x}_{0:q}$ is distinguishable from $\hat{x}_{0:q+1}$, where $\hat{x}_{0:q}$ is the random variable that denotes the value extracted by E in the intermediate hybrid experiment $H_{0:q}$ that is essentially the same as H_0, except that $e_1^1, \ldots, e_1^q$ are prepared as encryptions of x_1. (Thus, we have that $H_{0:t}$ is the same as H_{enc_1}.) In this case, we first note

[1] Here, the hybrids are such that $\tilde{V}$ uses x_0 in all session $j' < j$, and x_1 in all sessions $j' > j$.

that if the execution of $\langle P_{\mathsf{pWI}}, V_{\mathsf{pWI}} \rangle$ in session i concludes *before* $\hat{P}$ receives (e_1^{q+1}, e_2^{q+1}), then the witness used in $\langle P_{\mathsf{pWI}}, V_{\mathsf{pWI}} \rangle$ must be information-theoretically independent of the value encrypted in e_1^{q+1}, which gives us a contradiction. Therefore, we now only consider the case where the execution of $\langle P_{\mathsf{pWI}}, V_{\mathsf{pWI}} \rangle$ in session i concludes *after* $\hat{P}$ receives (e_1^{q+1}, e_2^{q+1}). In this case, we will construct a polynomial-time machine M that breaks the semantic security of the encryption scheme $(\mathbf{Gen}, \mathbf{Enc}, \mathbf{Dec})$.

M works in the same manner as hybrid $H_{0:q}$, except that it also interacts with an external challenger C (for the encryption scheme $(\mathbf{Gen}, \mathbf{Enc}, \mathbf{Dec})$) in the following manner. M receives a public key σ from C and then "forces" it to be the outcome of the $(q+1)^{th}$ coin-tossing subprotocol in session j. Specifically, after receiving the value σ_p from $\hat{P}$ in the $(q+1)^{th}$ coin-tossing subprotocol, M rewinds $\hat{P}$ and sends a value $\sigma_v = \sigma \oplus \sigma_p$. It now sends x_0, x_1 to C and receives a challenge ciphertext e^*. M continues in the same manner as $H_{0:q}$, except that it prepares $e_1^{q+1} = e^*$. Now, note that if e^* is an encryption of x_0, then this machine is identical to $H_{0:q}$, otherwise it is identical to $H_{0:q+1}$. M now invokes the knowledge extractor E on $\hat{P}$ in $\langle P_{\mathsf{pWI}}, V_{\mathsf{pWI}} \rangle$ in session i. Note that the sessions i and j may be interleaved in such a manner that when E rewinds $\hat{P}$ to send a new "challenge" in $\langle P_{\mathsf{pWI}}, V_{\mathsf{pWI}} \rangle$, either of the following two events happen:

1. $\hat{P}$ sends a new commitment string $c' = \mathrm{Com}(\sigma'_p)$ in the $(q+1)^{th}$ coin-tossing subprotocol in session j. In this case, M simply continues session j honestly until it receives σ'_p. At this point, it rewinds $\hat{P}$ again to send a value $\sigma_v = \sigma \oplus \sigma'_p$ and then continues honestly.

2. Alternatively, $\hat{P}$ may simply send a new value σ'_p and then proceed to prove its correctness in the execution of $\langle P_{\mathsf{pB}}, V_{\mathsf{pB}} \rangle_N^{j,q+1}$. If this is the case, then M simply aborts.

Now, conditioned on the event that M does not abort, we have that at some point, E stops and outputs a value, say, $\hat{x}$. Then, M finds b such that $f_{\mathsf{owf}}(\hat{x}) = y_b$ and outputs b to C. It follows easily that M succeeds with noticeable (in λ) advantage, which is a contradiction.

Thus it only remains to argue that M aborts only with negligible probability. To see this, we first note that it follows from the Soundness Lemma 3.1 that $\hat{P}$ only proves a true statement in each instance of $\langle P_{\mathsf{pB}}, V_{\mathsf{pB}} \rangle_N$, except with negligible probability. Then, by the computational binding property of the commitment scheme, we have that $\hat{P}$ cannot send decommitment c to two different values σ_p and σ'_p, except with negligible probability. Thus, we have the $\sigma'_p = \sigma_p$, except with negligible probability.

HYBRID H_{wi}: This is the same as H_{enc_1}, except that for every $q \in [t]$, $\tilde{V}$ uses the witness corresponding to e_1^q in the resettable-WI $\langle P_{\mathsf{rWI}}, V_{\mathsf{rWI}} \rangle^q$. We now invoke the knowledge extractor E on $\hat{P}$ in $\langle P_{\mathsf{pWI}}, V_{\mathsf{pWI}} \rangle$ in session i in experiment H_{wi}. Let $\hat{x}_{\mathsf{wi}}$ be the random variable that denotes the value that E outputs.

We now claim that $\hat{x}_{\mathsf{enc}_1} \approx \hat{x}_{\mathsf{wi}}$. Suppose that this is not the case. Then by a standard hybrid argument, there exists $q \in [t]$ such that $\hat{x}_{\mathsf{enc}_1:q}$ is distinguishable from $\hat{x}_{\mathsf{enc}_1:q+1}$ with noticeable probability, where $\hat{x}_{\mathsf{enc}_1:q}$ is the random variable that denotes the value extracted by E in the intermediate hybrid experiment $H_{\mathsf{enc}_1:q}$ that is essentially the same as H_{wi}, except that $\tilde{V}$ uses the witness corresponding to e_1^ℓ in $\langle P_{\mathsf{rWI}}, V_{\mathsf{rWI}} \rangle^\ell$ for every $\ell \in [1, q]$. (Thus, we have that $H_{\mathsf{enc}_1:t}$ is the same as H_{enc_1}.) In this case, we will construct a polynomial-time machine M that breaks the resettable witness indistinguishability property of $\langle P_{\mathsf{rWI}}, V_{\mathsf{rWI}} \rangle$. M works in the same manner as hybrid $H_{\mathsf{enc}_1:q}$, except that it forwards the $(q+1)^{th}$ instance of $\langle P_{\mathsf{rWI}}, V_{\mathsf{rWI}} \rangle$ in session j, i.e., $\langle P_{\mathsf{rWI}}, V_{\mathsf{rWI}} \rangle^{q+1}$, to an external prover P (for the resettable-WI protocol $\langle P_{\mathsf{rWI}}, V_{\mathsf{rWI}} \rangle$) in the following manner. M first gives w_1, w_2 to P, where w_1 is the witness corresponding to e_1^{q+1}, and similarly, w_2 is the witness corresponding to e_2^{q+1}. Now, during the execution of $\langle P_{\mathsf{rWI}}, V_{\mathsf{rWI}} \rangle^{q+1}$, M simply forwards each message msg_P from P to $\hat{P}$ and similarly forwards each response $\mathsf{msg}_{\hat{P}}$ from $\hat{P}$ to P. It then runs the knowledge extractor E on $\hat{P}$ in $\langle P_{\mathsf{pWI}}, V_{\mathsf{pWI}} \rangle$ in session i to extract a value, say $\hat{x}$. Note that if sessions i and j are scheduled such that when E rewinds $\hat{P}$ in $\langle P_{\mathsf{pWI}}, V_{\mathsf{pWI}} \rangle$ in session i, $\hat{P}$ sends

a new l^{th} round-message $\mathsf{msg}'_{\hat{P}}$ in $\langle P_{\mathsf{rWI}}, V_{\mathsf{rWI}} \rangle^{q+1}$, then M *resets* P to the point where its supposed to receive the l^{th} round message and sends $\mathsf{msg}'_{\hat{P}}$. It then continues the execution in the same manner as described above. When E finally outputs $\hat{x}$, then M finds b such that $f_{\mathsf{owf}}(\hat{x}) = y_b$ and outputs b to P. It follows easily that M succeeds with noticeable (in λ) advantage, which is a contradiction.

HYBRID H_{enc_2}: This is the same as H_{enc_2}, except that $\tilde{V}$ prepares each ciphertext e_2^q to be an encryption of x_1. We now run the extractor E on $\hat{P}$ in experiment H_{enc_2}. Let $\hat{x}_{\mathsf{enc}_2}$ be the random variable that denotes the value that E outputs. For the same reasons as argued above (for Hybrid H_{enc_1}), it follows that $\hat{x}_{\mathsf{wi}} \approx \hat{x}_{\mathsf{enc}_2}$.

This concludes the proof of soundness.

3.3.4 Proof of Concurrent Zero Knowledge

In this section, we prove that $\langle P, V \rangle$ is concurrent zero-knowledge in the BP model. Towards this end, we will construct a non-black-box (polynomial-time) simulator and then prove that the concurrent adversary's view output by the simulator is indistinguishable from the real view.

The Simulator. The simulator SIM consists of two parts, SIM_{easy} and SIM_{extract}. Loosely speaking, SIM_{extract} is only used to cheat in a "special" preamble block of a session in order to learn the secret key of a verifier, while SIM_{easy} is used for the remainder of the simulation, which includes following honest prover strategy in preamble blocks and simulating the proof phase of each session using the verifier's secret key as the trapdoor witness. Specifically, SIM_{extract} cheats in the $\langle P_{\mathsf{pB}}, V_{\mathsf{pB}} \rangle_N$ protocol by committing to an augmented verifier machine Π that contains the code of SIM_{easy}, allowing it to simulate all of the simulator messages except those generated by SIM_{extract} (in different sessions). As we show below, these messages can be bounded to a fixed value. We now describe the simulator in more

detail.

SETUP AND INPUTS. Our simulator SIM interacts with an adversary $V^* = (V_1^*, \ldots, V_N^*)$ who controls verifiers $V_1, \ldots, V_N$. V^* interacts with SIM in m sessions, and controls the scheduling of the messages. We give SIM non-black-box access to V^*. Throughout the interaction, SIM keeps track of a tuple $\vec{\beta} = (\beta_1, \ldots, \beta_N)$ representing the secret keys SIM has learned so far. At any point during the interaction either $\beta_i = \mathrm{sk}_i$ (more precisely, β_i is one of the coordinates of sk_i) or β_i is the symbol $\perp$. Initially, SIM sets each β_i to $\perp$, but it updates $\vec{\beta}$ throughout the interaction as it extracts secret keys. Additionally, SIM keeps a counter vector $\vec{a} = (a_1, \ldots, a_N)$, incrementing a_i each time it executes a preamble block using SIM_{extract} against V_i^*. We have SIM halt and output FAIL if any a_i ever surpasses λ^3. Our technical lemma shows that this happens with negligible probability. Finally, we have SIM keep track of a set of tuples

$$\Psi = \left\{ \left((i,j,k)_\gamma ; \phi_\gamma \right) : \gamma = 1, \ldots, n^3 N \right\}$$

where each $(i,j,k)_\gamma \in [N] \times [m] \times [t]$ and ϕ_γ is a string. The tuples $(i,j,k)_\gamma$ represent the preamble blocks played by SIM_{extract}; specifically, (i,j,k) corresponds to the $k-$th block of the $j-$th session against V_i^*. The string ϕ_γ is the collection of simulator messages sent in block $(i,j,k)_\gamma$. This set of tuples Ψ (along with β) will be the extra input given to the augmented machine. As we show below, the total size of Ψ will be *a priori* bounded by a polynomial in λ.

Consider the interaction of SIM with some V^* impersonating V_i. Each time V^* opens a session on behalf of V_i, SIM chooses a random $k \in \{1, \ldots, t\}$ according to a distribution D_t which we define later. This will be the only preamble block of the session played by SIM_{extract} provided that $\beta_i = \perp$ when the block begins. If SIM has already learned the secret key sk_i, it does not need to call SIM_{extract}. We now describe the parts of SIM

beginning with SIM_{easy}.

THE SUB-SIMULATOR SIM_{EASY}. Recall that SIM_{easy} is run on input β and Ψ. When SIM_{easy} is called to execute the next message of a preamble block, it checks if the message is already in Ψ. If this is the case, SIM_{easy} just plays the message. Otherwise, SIM_{easy} plays fairly, choosing a random σ_p and sending $c_p = \text{Com}(\sigma_p; s)$ for some s. Upon receiving σ_v, it returns σ_p and completes $\langle P_{\text{pB}}, V_{\text{pB}} \rangle$ using s as its witness. Its receipt of encryptions (e_1, e_2) and acceptance of $\langle P_{\text{rWI}}, V_{\text{rWI}} \rangle$ ends the preamble block. If SIM_{easy} does not accept V^*'s execution of $\langle P_{\text{rWI}}, V_{\text{rWI}} \rangle$ it aborts the interaction, as would an honest prover.

When SIM_{easy} is called to execute $\langle P_{\text{pWI}}, V_{\text{pWI}} \rangle$ then it checks if the secret key of the verifier is in β. If yes, SIM_{easy} completes $\langle P_{\text{pWI}}, V_{\text{pWI}} \rangle$ using sk_i as its witness. Otherwise, $\beta_i = \perp$ and SIM_{easy} halts outputting FAIL. Our technical lemma shows that the latter does not happen, except with negligible probability.

THE SUB-SIMULATOR SIM_{EXTRACT}. When SIM_{extract} is called to execute preamble block k of session j with verifier V_i^*, it receives Ψ, β and a as input. We assume $\beta_i = \perp$ since otherwise, SIM would not have called SIM_{extract}. Immediately upon being called, SIM_{extract} increments a_i and adds the tuple $\big((i,j,k); \phi\big)$ to Ψ. Initially, ϕ is the empty string, but each time SIM_{extract} sends a message, it appends the message to ϕ. By the end of the block, ϕ is a complete transcript of the simulator messages in preamble block (i,j,k).

The preamble block begins normally, with SIM_{extract} choosing a random string and sending c_p, a commitment to it. Upon receiving σ_v, however, SIM_{extract} runs **Gen** obtaining key pair (σ, τ) for the encryption scheme and returns $\sigma_p = \sigma \oplus \sigma_v$. Next, SIM_{extract} enters $\langle P_{\text{pB}}, V_{\text{pB}} \rangle$ which it completes using the already extracted secret key. Formally, when V^* sends h, beginning $\langle P_{\text{pB}}, V_{\text{pB}} \rangle$, SIM_{extract} chooses a random s and sends $\text{Com}\big(h(\Pi); s\big)$, where Π is the next message function of V^*, augmented with the ability to compute all the intermediate messages sent by SIM_{easy}. The machine Π takes input $y = (\Psi, \beta)$ and outputs

the next verifier message in an interaction between V^* and a machine M who plays exactly like SIM_{easy} with the following exception. For each tuple $\big((i,j,k);\phi\big) \in \Psi$, M reads its messages of block (i,j,k) from the string y. In order to simulate SIM_{easy} in the subprotocols $\langle P_{\text{pWI}}, V_{\text{pWI}}\rangle$, M also uses the tuple $\vec{\beta} = (\beta_1, \ldots, \beta_N)$ received as input, where each β_i is the secret key of the i'-th verifier (if available), and $\perp$ otherwise.

After committing to Π, and receiving r, SIM_{extract} completes $\langle P_{\text{pUA}}, V_{\text{pUA}}\rangle$ using witness $(\Pi, \Psi \| \beta, s)$ where Ψ and β might have been updated by other executions of SIM_{extract} occurring between the time SIM_{extract} sent $\text{Com}\big(h(\Pi); s\big)$ and received r. Our counter ensures that $|\Psi|$ is *a priori* bounded, while $|\beta|$ is bounded by definition. By construction, Π correctly predicts V^*'s message r, and so $(\Pi, \Psi \| \beta, s)$ is a valid witness for $\langle P_{\text{sUA}}, V_{\text{sUA}}\rangle$. Finally, SIM_{extract} receives encryptions e_1, e_2 and the proof of correctness in $\langle P_{\mathsf{rWI}}, V_{\mathsf{rWI}}\rangle$. It now decrypts the ciphertexts using τ thereby learning secret key sk_i of V_i^*. If the decrypted value is a valid secret key sk_i, then it updates β by setting $\beta_i = \text{sk}_i$. Otherwise, it outputs the abort symbol $\perp$ and stops. (It is easy to see that since the proof system $\langle P_{\mathsf{rWI}}, V_{\mathsf{rWI}}\rangle$ is sound, the probability of simulator outputting $\perp$ at this step is negligible.)

Analysis. There are two situations in which SIM outputs fail: if some counter a_i exceeds λ^3, or if SIM_{easy} enters an execution $\langle P_{\text{pWI}}, V_{\text{pWI}}\rangle$ without knowledge of sk. Note that the latter will not happen, as to enter an execution of $\langle P_{\text{pWI}}, V_{\text{pWI}}\rangle$, all preamble blocks, in particular the one played by SIM_{extract}, must be complete, ensuring that SIM_{extract} will have learned sk. In our main technical lemma, we show that no counter will surpass λ^3 by proving that after SIM has run SIM_{extract} λ^3 times against each V_i controlled by V^* it has, with overwhelming probability, learned sk. Before stating the lemma, we introduce some terminology.

Now, focusing on a given verifier, we say that V^* has *stopped* session j in block k if the $k-$th preamble block of session j has begun, but the $(k+1)-$th has not. We say that V^*

is playing *strategy* $\vec{k'} = (k'_1, \ldots, k'_m)$ if session j is stopped in block k'_j for all $j = 1, \ldots, m$. As the interaction takes polynomial time, V^* only gets to play polynomially many strategies over the course of the interaction. Let $k_j \in \{1, \ldots, t\}$ be the random number chosen by SIM at the beginning of session j as per distribution D_t. This gives us a tuple $\vec{k} = (k_1, \ldots, k_m)$ where the k_j are chosen independently according to the distribution D_t (defined below). At any time during the interaction, we say that V^* has *won* (resp. *lost*, *tied*) session j if $k'_j = k_j$ (resp. $k'_j > k_j$, $k'_j < k_j$). A win for V^* corresponds to SIM having run SIM_{extract}, but not yet having learned sk. As SIM only gets to call SIM_{extract} λ^3 times, a win for V^* means that SIM has used up one of its budget of λ^3 without any payoff. A loss for V^* corresponds to SIM running SIM_{extract} and learning sk, thereby allowing SIM to call SIM_{easy} in all remaining sessions. A tie means that SIM has not yet called SIM_{extract} in the session, and therefore has not used any of its budget, but has not learned sk.

Notice that these wins and ties are "temporary" events. Indeed, by the end of each session, V^* will have lost, as he will have completed the preamble block run by SIM_{extract}. However, we choose to use this terminology to better convey the key intuition of our analysis: for SIM to output FAIL, it must be that at some point during the interaction, for some identity, V^* has won at least λ^3 sessions and has not lost any. We will therefore focus precisely on proving that the probability that a PPT adversary V^* runs in the experiment m sessions so that the counter for one identity reaches the value λ^3 is negligible.

For a verifier strategy $\vec{k'}$ and a polynomial m, let $P_{(\vec{k'},m)}(W, L)$ be the probability that in an $m-$session interaction between V^* and SIM that V^* wins for some identity exactly W sessions and loses exactly L, given that V^* plays strategy $\vec{k'}$. The probability is over SIM's choice of $\vec{k}$ with $k_j \in \{1, \ldots, t\}$ chosen independently according to D_t (defined below) for all $j = 1, \ldots, m$.

The Distribution D_t and the Main Technical Lemma. Define D_t to be the distri-

bution on $\{1, \ldots, t\}$ such that

$$p_{k'} = \mathrm{Prob}_{k \in D_t}\big(k = k'\big) = \varepsilon n^{k'},$$

where ε is such that $\sum p_{k'} = 1$. Note that ε is negligible in λ.

Lemma 3.2 (Main Technical Lemma)**.** *Let $\vec{k'}$ be a verifier strategy and $m = m(n)$ a polynomial. Then we have*

$$P_{(\vec{k'},m)}(n^3, 0) = \mathbf{negl}(\lambda).$$

The above proves that any verifier strategy has a negligible chance of having λ^3 wins and no losses. As V^* plays polynomially many (i.e., N) strategies throughout the course of the interaction, the union bound proves that V^* has a negligible chance of ever achieving λ^3 wins and 0 losses. From this it follows that, with overwhelming probability, V^* will never have at least λ^3 wins and no losses, which implies that SIM outputs FAIL with negligible probability as desired. The main idea of the proof is similar to the random tape switching technique of [PRS02] and [MP07].

Proof. We fix a verifier strategy $\vec{k'}$ and a polynomial m and write $P(W, L)$ instead of $P_{(\vec{k'},m)}(W, L)$. Let $p_{k'}$ (resp. $q_{k'}$) be the probability that V^* wins (resp. loses) a session given that he stops the session in block k'. We chose the distribution D_t carefully to have the following two properties. First, since $p_1 = \varepsilon n$ is negligible, we may assume that V^* never stops in the first block of a session. And secondly, for $k' \geq 2$ we have,

$$q_{k'} = \sum_{i=1}^{k'-1} p_{k'} = \varepsilon \frac{n^{k'} - 1}{n - 1} \geq \frac{\varepsilon n^{k'}}{2n} = \frac{p_{k'}}{2n}.$$

It follows that no matter which block V^* stops a session in, it will hold that the probability he wins in that session is less then $2n$ times the probability that he looses that session. We

will use this upper bound on the probability of V^* winning a single session to show that $P(n^3, 0)$ is negligible.

Let A be the event, $(W, L) = (n^3, 0)$, B be the event $W + L = n^3$ and $\neg B$ the event $W + L \neq n^3$. Since, $A \subset B$, and since $P(A|\neg B) = 0$, we have that

$$P(n^3, 0) = P(A) = P(A|B)P(B) + P(A|\neg B)P(\neg B) = P(A|B)P(B) \leq P(A|B),$$

and so it suffices to prove that $P(A|B)$ is negligible. We continue the proof for the case $W + L = n^3$ (and thus $m \geq n^3$).

If $W + L = n^3$ then V^* ties all but λ^3 of the sessions. Let $\mathcal{C} = \{C \subset [m] : |C| = n^3\}$. Then $\mathcal{C}$ is the set of possible positions for the sessions which are not ties. We are looking to bound $P\big((W, L) = (n^3, 0)\big|W + L = n^3\big)$ and so we condition on the $C \in \mathcal{C}$. Once a fixed C is chosen, the position of each session which is not a tie is determined. Each such session must either be a win or a loss for V^*. Let p be the probability that some such session is a win. Since we proved already that the probability that V^* wins in a given session is less then $2n$ times the probability that V^* looses in that session, we have that $p \leq 2n(1 - p)$. Solving gives $p \leq \big(1 - \frac{1}{2n+1}\big)$. It follows that for any $C \in \mathcal{C}$, the probability that all sessions in C are wins is

$$\left(1 - \frac{1}{2n+1}\right)^{n^3} \leq \left[\left(1 - \frac{1}{2n+1}\right)^{2n+1}\right]^n \leq e^{-n}.$$

From the viewpoint of random tape switching, we have shown that for every random tape causing every session of C to be a win, there are exponentially many which cause a different outcome.

We therefore have

$$\begin{aligned} P(n^3,0) &\leq P\big((W,L)=(n^3,0)\big|W+L=n^3\big) \\ &= \sum_{C\in\mathcal{C}} P\big((W,L)=(n^3,0)\big|C\big)P(C) \\ &\leq e^{-n}\sum_{C\in\mathcal{C}} P(C) = e^{-n}, \end{aligned}$$

as desired. □

Bounding the length parameter $\ell(N)$. From the above lemma, it follows easily that the total length of the auxiliary input y to the machine Π committed by SIM_{extract} (at any time) is bounded by $\lambda^3 \cdot N \cdot P(\lambda)$, where $P(\lambda)$ is a polynomial upper bound on the total length of prover messages in one protocol session plus the length of a secret. Thus, when $\ell(N) \geq n^3 \cdot N \cdot P(\lambda)$, we have that $|y| \leq |r| - \lambda$, as required.

We now show through a series of hybrid experiments that the simulator's output is perfectly indistinguishable from the output of the adversary when interacting with honest provers.

3.4 The Constant Round Protocol

3.4.1 Building Blocks

Statistically binding commitment schemes. We will make use of a statistically binding string commitment scheme, denoted Com. For simplicity of exposition, we will make the simplifying assumption that Com is a non-interactive perfectly binding commitment scheme. In reality, Com would be taken to be a standard 2-round commitment scheme, e.g. [Nao91]. Naor's commitment scheme exists as long as one way functions exist.

Signature schemes. We will use a signature scheme $(\mathbf{KeyGen}, \mathbf{Sign}, \mathbf{VERIFY})$ that is unforgeable against chosen message attacks. Note that such signatures schemes are known based on one way functions [Rom90].

Witness indistinguishable arguments of knowledge. Like the previous protocol, we make use of a witness-indistinguishable proof of knowledge (WIPOK). Such a scheme can be constructed, for example, by parallel repetition of the 3-round Blum's protocol for Graph Hamiltonicity [Blu86]. We will denote such an argument system by $\langle P_{\mathsf{WI}}, V_{\mathsf{WI}} \rangle$.

The universal argument of [BG02]. In our construction, we will use the 4-round universal argument system (UA), denoted pUA. Such an argument system is sound also when the prover sends the statement in the very last round.

3.4.2 The Protocol

Recall the relation R_{sim} from Section 3.3, shown in Figure 3.2. Let P and V denote the prover and verifier respectively. Let N denote the bound on the number of verifiers present in the system. In our construction, the identity of a verifier V_i corresponds to a verification key vk_i of a secure signature scheme and a hash function $h_i \in \mathcal{H}_\lambda$ from a family $\mathcal{H}_\lambda$ of collision-resistant hash functions. Let $(\mathbf{KeyGen}, \mathbf{Sign}, \mathbf{VERIFY})$ be a secure signature scheme. Let $\langle P_{\mathsf{WI}}, V_{\mathsf{WI}} \rangle$ be a witness-indistinguishable argument of knowledge system. Let pUA be the universal argument (UARG) system of [BG02]; the transcript is composed of four messages $(h, \beta, \gamma, \delta)$ where h is a collision-resistant hash function.

The protocol $\langle P, V \rangle$ is described in Figure 3.5. For our purposes, we set the length parameter $\ell(N) = N \cdot P(\lambda) + \lambda$, where $P(\lambda)$ is a polynomial upper bound on the total length of the prover messages in the UARG pUA plus the output length of a hash function $h \in \mathcal{H}_\lambda$. For simplicity we omit some standard checks (e.g., the prover needs to check that vk and h

are recorded, the prover needs to check that the signatures is valid).

Parameters: Security parameter λ, number of players $N = N(\lambda)$, length parameter $\ell(N)$.
Common Input: $x \in \{0,1\}^{\text{poly}(n)}$.
Private Input to P: A witness w s.t. $R_L(x,w) = 1$.
Private Input to V: A key pair $(sk, vk) \stackrel{\text{R}}{\leftarrow} \textbf{KeyGen}(1^\lambda)$, and a hash function $h \stackrel{\text{R}}{\leftarrow} \mathcal{H}_\lambda$.

Stage 1 (Preamble Phase):

$V \to P$: Send vk, h.

$P \to V$: Send $c = \text{Com}(0^\lambda)$.

$V \to P$: Send $r \stackrel{\text{R}}{\leftarrow} \{0,1\}^{\ell(N)}$, and $\sigma = \textbf{Sign}_{sk}(c\|r)$.

$P \to V$: Send $c' = \text{Com}(0^\lambda)$.

$V \to P$: Send $\gamma \stackrel{\text{R}}{\leftarrow} \{0,1\}^\lambda$, and $\sigma' = \textbf{Sign}_{sk}(c'\|\gamma)$.

Stage 2 (Proof Phase):

$P \leftrightarrow V$: An execution of WIPOK $\langle P_{\mathsf{WI}}, V_{\mathsf{WI}} \rangle$ to prove the OR of the following statements:

1. $\exists w \in \{0,1\}^{\text{poly}(|x|)}$ s.t. $R_L(x,w) = 1$.
2. $\exists \langle c, r, \sigma \rangle$, and $\langle \beta, \gamma, \delta, c', t, \sigma' \rangle$ s.t.
   - $\textbf{VERIFY}_{vk}(c\|r;\sigma) = 1$, and
   - $c' = \text{Com}(\beta; t)$, and $\textbf{VERIFY}_{vk}(c'\|\gamma;\sigma') = 1$, and
   - $(h, \beta, \gamma, \delta)$ is an accepting transcript for a UARG pUA proving the following statement: $\exists \langle \Pi, y, s \rangle$ s.t. $R_{\mathsf{sim}}(\langle h, c, r \rangle, \langle \Pi, y, s \rangle) = 1$.

Figure 3.5: Protocol $\langle P, V \rangle -$ Constant Round cZK in the BP Model

The completeness property of $\langle P, V \rangle$ follows immediately from the construction.

3.4.3 Proof of Concurrent Soundness

Consider the interaction between a cheating P^* and an honest V. Suppose that P^* fools V into accepting a false proof in some session with non-negligible probability. We show how to reduce P^* to an adversary that breaks the security of one of the used ingredients. We will first consider P^* as a sequential malicious prover. We will discuss the issues deriving

from a concurrent attack later.

First of all, notice that by the proof of knowledge property of the second WIPOK, we have that with non-negligible probability, an efficient adversary E can simply run as a honest verifier and extract a witness from that WIPOK of session l where the false statement is proved. Since the statement is false, the witness extracted will therefore be $(c, r, \sigma, \beta, \gamma, \delta, c', t, \sigma')$ such that $\mathbf{VERIFY}_{vk}(c\|r; \sigma) = 1$, $c' = \mathrm{Com}(\beta; t)$, $\mathbf{VERIFY}_{vk}(c'\|\gamma; \sigma') = 1$, and $(h, \beta, \gamma, \delta)$ is an accepting transcript for a UARG pUA proving the statement $\exists\langle \Pi, y, s\rangle$ s.t. $R_{\mathsf{sim}}(\langle h, c, r\rangle, \langle \Pi, y, s\rangle) = 1$, and h is the hash function corresponding to the verifier run by E in session l.

By the security of the signature scheme, it must be the case that signatures σ and σ' were generated and sent by E during the experiment (the reduction is standard and omitted).

Therefore we have that with non-negligible probability there is a session i where h and γ were played honestly by E, $(h, \beta, \gamma, \delta)$ is an accepting transcript for the UARG for $R_{\mathsf{sim}}(\langle h, c, r\rangle, \langle \Pi, y, s\rangle) = 1$, and a commitment to β was given before γ was sent. Moreover, there is a session j where c and r where played as commitment and challenge. Remember that the session l is the one where the false statement is proved.

We can now complete the proof by relying almost verbatim on the same analysis of [Bar01, BG02]. Indeed, by rewinding the prover and changing the challenge r in session j, with another random string, we would have an execution identically distributed with respect to the previous one. Therefore it will happen with non-negligible probability that the prover succeeds in session l, still relying on the information obtained in sessions i and j. The analysis of [**?**, BG02] by relying on the weak proof of knowledge property of the UA, shows that this event can be reduced to finding a collision that contradicts the collision resistance of h.

We finally discuss the case of a concurrent adversarial prover. Such an attack is played by a prover aiming at obtaining from concurrent sessions some information to be used in the target session where the false theorem must be proved. In previous work in the BPK model and in the BP model this was a major problem because the verifier used to give a proof of knowledge of its secret key, and the malleability of such a proof of knowledge could be exploited by the malicious prover. Our protocol however bypasses this attack because our verifier does not give a proof of knowledge of the secret key of the signature scheme, but only gives signatures of specific messages. Indeed the only point in which the above proof of soundness needs to be upgraded is the claim that by the security of the signature scheme, it must be the case that signatures σ and σ' were generated and sent by E during the experiment. In case of sequential attack, this is true because running the extractor of the WIPOK in session l does not impact on other sessions since they were played in full either before or after session l. Instead, in case of a concurrent attack, instead, while rewinding the adversarial prover, new sessions could be started and more signatures could be needed. As a result, it could happen and that in such new sessions the prover would ask precisely the same signatures that are then extracted from the target session. We can conclude that this does not impact on the proof for the following two reasons. First, in the proof of soundness it does not matter if those signatures appear in the transcript of the attack, or just in the transcript of a rewinded execution. Second, the reduction on the security of the signature scheme works for any polynomial number of signatures asked to the oracle, therefore still holds in case of a concurrent attack. Indeed, the work of E is performed in polynomial time even when rewinding a concurrent malicious prover, therefore playing in total (i.e., summing sessions in the view of the prover and sessions played during rewinds) a polynomial number of sessions, and therefore asking a polynomial number of signatures only to the signature oracle.

Further details on the proof of soundness. Given a transcript $(h, UA1, UA2, UA3)$ for the universal argument of [BG02], notice that soundness still works when the prover sends the statement to the verifier only at the 4th round. This is because the prover commits to the Merkle Tree of the PCP in message $UA1$, and therefore the statement is already fixed in that commitment and the PCP would not work for other statements. The proof of concurrent soundness of our protocol goes through a reduction to the soundness of the universal argument of [BG02] and goes as follows.

Let P^*_{ua} be the adversarial prover that we construct against the universal argument of [BG02], by making use of the adversary P^* of our protocol. Let V_{ua} be the honest verifier of the universal argument of [BG02]. P^*_{ua} gets "h" from V_{ua} and plays it in a random session s of the experiment (it could therefore be played in a rewinding thread) with P^*. Later on, since by contradiction P^* is successful, UA messages $(UA1, UA2, UA3)$ are extracted and with noticeable probability they correspond to session s. Therefore P^*_{ua} sends $UA1$ to V_{ua} and gets back $UA2'$. Then P^*_{ua} rewinds P* to the precise point where $UA2$ was played. Now P^*_{ua} plays $UA2'$. Again, later on, since by contradiction P^* is successful, P^*_{ua} will again extract from P^* and with noticeable probability (still because the number of sessions played in the experiment is polynomial), it will get an accepting transcript $(UA1, UA2', UA3^*)$ for the same statement (this is guaranteed by the security of the signature scheme and the binding of the commitment). Then P^*_{ua} can send $UA3^*$ to V_{ua} therefore proving a false statement.

3.4.4 Proof of Concurrent Zero-Knowledge

In this section, we prove that the protocol $\langle P, V \rangle$ described in Figure 3.5 is concurrent zero-knowledge in the bounded player model. Towards this end, we will construct a non-black-box (polynomial-time) simulator and then prove that the concurrent adversary's view output by the simulator is indistinguishable from the real view. We start by giving an overview of the

proof and then proceed to give details.

Overview. Recall that unlike the bounded concurrency model, the main challenge in the bounded player model is that the total number of sessions that a concurrent verifier may schedule is not a priori bounded. Thus, one can not directly employ Barak's simulation strategy of committing to a machine that takes only a bounded-length input y (smaller than the challenge string r) and outputs the next message of the verifier. Towards this endIt follows that once the simulator is able to "solve" the identity of a specific verifier, then it does not need to perform any more "expensive" (Barak-style) non-black-box simulation for that identity. Then, the main challenge remaining is to ensure that the expensive non-black-box simulations that need to be performed *before* the simulator can solve a particular identity, can be a-priori bounded, regardless of the number of concurrent sessions opened by the verifier. Indeed, the protocol of Section 3.3 use a randomized simulation strategy (that crucially relies on a super-constant number of rounds) to achieve this effect.

In our case, we also build on the same set of observations. However, we crucially follow a different strategy to a-priori bound the number of expensive non-black-box simulations that need to performed in order to solve a given identity. In particular, unlike the protocol of Section 3.3, where the "trapdoor" for a given verifier simply corresponds to its secret key, in our case, the trapdoor consists of a signed statement and a corresponding universal argument proof transcript (where the signature is computed by the verifier using the signing key corresponding to its identity). Further, and more crucially, instead of the simulator making a "disjoint" effort in each session in order to extract the trapdoor, in our case, the simulator gradually builds the trapdoor by making "joint" effort across the sessions. In fact, our simulator only performs *one* expensive non-black-box simulation per identity; as such, the a-priori bound on the number of identities immediately yields us the desired effect. Indeed, this is why we can perform concurrent simulation in only a constant number of

rounds.

The Simulator. We now proceed to describe our simulator $\mathcal{S}$. Let N denote the a priori bound on the number of verifiers in the system. Then, the simulator $\mathcal{S}$ interacts with an adversary $V^* = (V_1^*, \ldots, V_N^*)$ who controls N verifiers. V^* interacts with $\mathcal{S}$ in m sessions, and controls the scheduling of the messages. $\mathcal{S}$ is given non-black-box access to V^*.

The simulator $\mathcal{S}$ consists of two main subroutines, namely, $\mathcal{S}_{\mathsf{easy}}$ and $\mathcal{S}_{\mathsf{heavy}}$. As the name suggests, the job of $\mathcal{S}_{\mathsf{heavy}}$ is to perform the "expensive" non-black-box simulation operations, namely, constructing the transcripts of universal arguments, which yield a trapdoor for every verifier V_i. On the other hand, $\mathcal{S}_{\mathsf{easy}}$ computes the actual (simulated) prover messages in both the preamble phase and the proof phase, by using the trapdoors. We now give more details.

SIMULATOR $\mathcal{S}$. Throughout the simulation, $\mathcal{S}$ maintains the following three data structures, each of which is initialized to $\perp$:

1. a list $\vec{\pi} = (\pi_1, \ldots, \pi_N)$, where each π_i is either $\perp$ or is computed to be $h_i(\Pi)$. Here, h_i is the hash function corresponding to V_i and Π is the augmented machine code that is used for non-black-box simulation. We defer the description of Π to below.

2. a list $\vec{CHANGEME}^{\mathsf{heavy}} = (CHANGEME_1^{\mathsf{heavy}}, \ldots, CHANGEME_N^{\mathsf{heavy}})$, where each $CHANGEME_i^{\mathsf{heavy}}$ is a tuple

   $\langle h_i, c, r, \Pi, y, s\rangle$ s.t. $R_{\mathsf{sim}}(\langle h_i, c, r\rangle, \langle \Pi, y, s\rangle) = 1$.

3. a list $\vec{CHANGEME}^{\mathsf{easy}} = (CHANGEME_1^{\mathsf{easy}}, \ldots, CHANGEME_N^{\mathsf{easy}})$, where each $CHANGEME_i^{\mathsf{easy}}$ is a tuple $\langle c, r, \sigma, \beta, \gamma, \delta, c', t, \sigma'\rangle$ s.t.

   - $\mathbf{VERIFY}_{vk_i}(c \| r; \sigma) = 1$, and
   - $c' = \mathrm{Com}(\beta; t)$, and $\mathbf{VERIFY}_{vk_i}(c' \| \gamma; \sigma') = 1$, and

- $(h_i, \beta, \gamma, \delta)$ is an accepting transcript for a UARG pUA proving the following statement: $\exists \langle \Pi, y, s \rangle$ s.t. $R_{\mathsf{sim}}(\langle h_i, c, r \rangle, \langle \Pi, y, s \rangle) = 1$.

Augmented machine Π. The augmented machine code Π simply consists of the code of the adversarial verifier V^* and the code of the subroutine $\mathcal{S}_{\mathsf{easy}}$ (with a sufficiently long random tape hardwired, to compute the prover messages in each session) , i.e., $\Pi = (V^*, \mathcal{S}_{\mathsf{easy}})$. The input y to the machine Π consists of the lists $\vec{\pi}$ and $\vec{CHANGEME}^{\mathsf{easy}}$, i.e., $y = (\vec{\pi}, \vec{CHANGEME}^{\mathsf{easy}})$. Note that it follows from the description that $|y| \leq \ell(N) - \lambda$.

We now describe the subroutines $\mathcal{S}_{\mathsf{easy}}$ and $\mathcal{S}_{\mathsf{heavy}}$, and then proceed to give a formal description of $\mathcal{S}$. For simplicity of exposition, in the discussion below, we assume that the verifier sends the first message in the WIPOK $\langle P_{\mathsf{WI}}, V_{\mathsf{WI}} \rangle$.

ALGORITHM $\mathcal{S}_{\mathsf{easy}}(i, \mathsf{msg}_j^V, \vec{\pi}, \vec{CHANGEME}^{\mathsf{easy}}; z)$. The algorithm $\mathcal{S}_{\mathsf{easy}}$ prepares the (simulated) messages of the prover P in the protocol. More specifically, when executed with input $(i, \mathsf{msg}_j^V, \vec{\pi}, \vec{CHANGEME}^{\mathsf{easy}}; z)$, $\mathcal{S}_{\mathsf{easy}}$ does the following:

1. If msg_j^V is the first verifier message of the preamble phase from V_i in a session, then $\mathcal{S}_{\mathsf{easy}}$ parses $\vec{\pi}$ as $\pi_1, \ldots, \pi_N$. It computes and outputs $c = \mathrm{Com}(\pi_i; z)$.

2. If msg_j^V is the second verifier message of the preamble phase from V_i in a session, then $\mathcal{S}_{\mathsf{easy}}$ computes and outputs $c = \mathrm{Com}(\beta; z)$, where β is the corresponding (i.e., fourth) entry in $CHANGEME_i^{\mathsf{easy}} \in \vec{CHANGEME}^{\mathsf{easy}}$.

3. If msg_j^V is a verifier message of the WIPOK from V_i in the proof phase of a session, then if $CHANGEME_i^{\mathsf{easy}} = \perp$, then $\mathcal{S}_{\mathsf{easy}}$ aborts and outputs $\perp$, otherwise $\mathcal{S}_{\mathsf{easy}}$ simply runs the code of the honest P_{WI} to compute the response using randomness z and the trapdoor witness $CHANGEME_i^{\mathsf{easy}}$.

ALGORITHM $\mathcal{S}_{\mathsf{heavy}}(i, j, \gamma, \vec{CHANGEME}^{\mathsf{heavy}})$. The algorithm $\mathcal{S}_{\mathsf{heavy}}$ simply prepares *one*

UARG transcript for every verifier V_i, which in turn is used as a trapdoor by the algorithm $\mathcal{S}_{\mathsf{easy}}$. More concretely, when executed with input $(i, j, \gamma, \vec{CHANGEME}^{\mathsf{heavy}})$, $\mathcal{S}_{\mathsf{heavy}}$ does the following:

1. If $j = 1$, then $\mathcal{S}_{\mathsf{heavy}}$ parses the i^{th} entry $CHANGEME_i^{\mathsf{heavy}}$ in $\vec{CHANGEME}^{\mathsf{heavy}}$ as (h_i, c, r, Π, y, s). It runs the honest prover algorithm P_{UA} and computes the first message β of a UARG for the statement: $\exists\langle\Pi, y, s\rangle$ s.t. $R_{\mathsf{sim}}(\langle h_i, c, r\rangle, \langle\Pi, y, s\rangle) = 1$. $\mathcal{S}_{\mathsf{heavy}}$ saves its internal state as state_i and outputs β.[2]

2. If $j = 2$, then $\mathcal{S}_{\mathsf{heavy}}$ uses state_i and γ to honestly compute the final prover message δ for the UARG with prefix (h_i, β, γ). It outputs δ.

ALGORITHM $\mathcal{S}$. Given the above subroutines, the simulator $\mathcal{S}$ works as follows. We assume that every time $\mathcal{S}$ updates the lists $\vec{\pi}$ and $\vec{CHANGEME}^{\mathsf{easy}}$, it also automatically updates the entry corresponding to y (i.e., the fifth entry) in each $CHANGEME_i^{\mathsf{heavy}} \in \vec{CHANGEME}^{\mathsf{heavy}}$. For simplicity of exposition, we do not explicitly mention this below.

Preamble phase:

1. On receiving the first message $\mathsf{msg}_1^V = (vk_i, h_i)$ from V^* on behalf of V_i in the preamble phase of a session, $\mathcal{S}$ first checks whether $\pi_i = \perp$ (where π_i is the i^{th} entry in the list $\vec{\pi}$); if the check succeeds, then $\mathcal{S}$ updates $\pi_i = h_i(\Pi)$. Next, $\mathcal{S}$ samples fresh randomness s from its random tape and runs $\mathcal{S}_{\mathsf{easy}}$ on input $(i, \mathsf{msg}_1^V, \vec{\pi}, \vec{CHANGEME}^{\mathsf{easy}}; s)$. $\mathcal{S}$ sends the output string c from $\mathcal{S}_{\mathsf{easy}}$ to V^*. Further, $\mathcal{S}$ adds $(h_i, c, \cdot, \Pi, y, s)$ to $CHANGEME_i^{\mathsf{heavy}}$ and $(c, \cdot, \cdot, \cdot, \cdot, \cdot, \cdot, \cdot, \cdot)$ to $CHANGEME_i^{\mathsf{easy}}$.

2. On receiving the second message message $\mathsf{msg}_2^V = (r, \sigma)$ from V^* on behalf of V_i in the preamble phase of a session, $\mathcal{S}$ first verifies the validity of the signature σ w.r.t. vk_i. If the check fails, $\mathcal{S}$ considers this session aborted (as the prover would do) and

[2]For simplicity of exposition, we describe $\mathcal{S}_{\mathsf{heavy}}$ as a stateful algorithm.

ignores any additional message for this session. Otherwise, $\mathcal{S}$ checks whether the entries corresponding to r and σ (i.e., 2nd and 3rd entries) in $CHANGEME_i^{\mathsf{easy}}$ are $\perp$. If the check succeeds, then:

- $\mathcal{S}$ sets r as 3rd entry of $CHANGEME_i^{\mathsf{heavy}}$ and r, σ as second and third entries of $CHANGEME_i^{\mathsf{easy}}$.
- Further, $\mathcal{S}$ runs $\mathcal{S}_{\mathsf{heavy}}$ on input[3] $(i, 1, \perp, \vec{CHANGEME}^{\mathsf{heavy}})$ to compute the message β of a UARG for the statement: $\exists \langle \Pi, y, s \rangle$ s.t. $R_{\mathsf{sim}}(\langle h_i, c, r \rangle, \langle \Pi, y, s \rangle) = 1$. Here h_i, c, r, Π, y, s are such that $CHANGEME_i^{\mathsf{heavy}} = \langle h_i, c, r, \Pi, y, s \rangle$.
- On receiving the output message β, $\mathcal{S}$ sets to β the fourth slot of $CHANGEME_i^{\mathsf{easy}}$.

Next, $\mathcal{S}$ samples fresh randomness t and runs $\mathcal{S}_{\mathsf{easy}}$ on input $(i, \mathsf{msg}_2^V, \vec{\pi}, \vec{CHANGEME}^{\mathsf{easy}}; t)$. On receiving the output string c' from $\mathcal{S}_{\mathsf{easy}}$, $\mathcal{S}$ forwards it to V^*. Further, $\mathcal{S}$ sets to (c', t) the 7th and 8th slot of $CHANGEME_i^{\mathsf{easy}}$.

3. Finally, on receiving the last message $\mathsf{msg}_{\mathsf{fin}}^V = (\gamma, \sigma')$ from V^* on behalf of V_i in the preamble phase of a session, $\mathcal{S}$ first verifies the validity of the signature σ' w.r.t. vk_i. If the check fails, $\mathcal{S}$ considers this session aborted (as the prover would do) and ignores any additional message for this session. Otherwise, $\mathcal{S}$ checks whether the entries corresponding to γ and σ' in $CHANGEME_i^{\mathsf{easy}}$ are $\perp$. If the check succeeds, then:

   - $\mathcal{S}$ sets to γ and σ' the 5th and 9th slot of $CHANGEME_i^{\mathsf{easy}}$.
   - Further, $\mathcal{S}$ runs $\mathcal{S}_{\mathsf{heavy}}$ on input $(i, 2, \gamma, \vec{CHANGEME}^{\mathsf{heavy}})$ to compute the final prover message δ of the UARG with prefix (h_i, β, γ), where (β, γ) are the corresponding entries in $CHANGEME_i^{\mathsf{easy}}$.
   - On receiving the output message δ, $\mathcal{S}$ sets to δ the 6th slot of $CHANGEME_i^{\mathsf{easy}}$.

[3]For simplicity of exposition, we assume that randomness is hardwired in $\mathcal{S}^{\mathsf{heavy}}$ and do not mention it explicitly.

Proof phase: On receiving any message msg_j^V from V^* on behalf of V_i, $\mathcal{S}$ runs $\mathcal{S}_{\mathsf{easy}}$ on input $(i, \mathsf{msg}_j^V, \vec{\pi}, \vec{CHANGEME}^{\mathsf{easy}})$ and fresh randomness. $\mathcal{S}$ forwards the output message of $\mathcal{S}_{\mathsf{easy}}$ to V^*.

This completes the description of $\mathcal{S}$ and the subroutines $\mathcal{S}_{\mathsf{easy}}$, $\mathcal{S}_{\mathsf{heavy}}$. It follows immediately from the above description that $\mathcal{S}$ runs in polynomial time and outputs $\perp$ with probability negligibly close to an honest prover.

We now show through a series of hybrid experiments the simulator's output is computationally indistinguishable from the output of the adversary when interacting with honest provers. Our hybrid experiments will be H_i for $i = 0, \ldots, 3$. We write $H_i \approx H_j$ if no V^* can distinguish (except with negligible probability) between its interaction with H_i and H_j.

Hybrid H_0. Experiment H_0 corresponds to the honest prover. That is, in every session $j \in [m]$, H_0 sends c and c' as commitments to the all zeros string in the preamble phase. We provide H_0 with a witness that $x \in L$ which it uses to complete the both executions of the WIPOK $\langle P_{\mathsf{WI}}, V_{\mathsf{WI}} \rangle$ played in each session.

Hybrid H_1. Experiment H_1 is similar to H_0, except the following. For every $i \in [N]$, for every session corresponding to verifier V_i, the commitment c in the preamble phase is prepared as a commitment to $\pi_i = h_i(\Pi)$, where h_i is the hash function in the identity of V_i and Π is the augmented machine code as described above.

The computational hiding property of Com ensures that $H_1 \approx H_0$.

Hybrid H_2. Experiment H_2 is similar to H_1, except the following. For every $i \in [N]$, for every session corresponding to verifier V_i, the commitment c' in the preamble phase is prepared as a commitment to the string β with randomness t, where β is the first prover message of a UARG computed by $\mathcal{S}_{\mathsf{heavy}}$, in the manner as described above.

The computational hiding property of Com ensures that $H_2 \approx H_1$.

Hybrid H_3. Experiment H_3 is similar to H_2, except the following. For every $i \in [N]$, for every session corresponding to verifier V_i, the WIPOK $\langle P_{\mathsf{WI}}, V_{\mathsf{WI}} \rangle$ in the proof phase is executed using the trapdoor witness $CHANGEME_i^{\mathsf{easy}}$, in the manner as described above. Note that this is our simulator $\mathcal{S}$.

The witness indistinguishability property of $\langle P_{\mathsf{WI}}, V_{\mathsf{WI}} \rangle$ ensures that $H_3 \approx H_2$.

3.5 Concurrent Self-Composition in the BP Model

In this section, we present the definition for concurrent (self-composition) secure multi-party computation in the bounded player model. The definition we give below is an adaptation of the definition of concurrent secure computation with adaptive inputs [Lin04, Pas04a], to the setting of bounded player model. Parts of the definition below have been taken almost verbatim from [Lin04, Pas04a].

We first setup notation. We denote computational indistinguishability by COMP, and the security parameter by λ. For notational simplicity, we let the lengths of the parties' inputs be λ. An n-ary functionality is denoted as $f : (\{0,1\}^*)^n \rightarrow (\{0,1\}^*)^n$, where $f = f_1, \ldots, f_n$. Let $P_1, \ldots, P_n$ denote the set of n-player that wish to jointly compute f. The output of P_i with input x_i is defined to be $f_i(\vec{x})$, where $\vec{x} = x_1, \ldots, x_n$. In the context of concurrent composition, each party uses many inputs (one per execution) and these may be chosen *adaptively* based on previous outputs. The fact that bounded player model is considered relates to the fact that the total number of parties that may engage in concurrent protocol executions is *a priori* bounded.

In this work, we consider a malicious, static adversary. The scheduling of the messages across the concurrent executions is controlled by the adversary. We do not focus on fairness, hence we do not guarantee output delivery. The security of a protocol is analyzed by com-

paring what an adversary can do in the protocol to what it can do in an ideal scenario, where a trusted party computes the function output on the inputs of the parties. Unlike in the case of stand-alone computation, in the setting of concurrent executions, the trusted party computes the functionality many times, each time upon different inputs. We now proceed to describe the ideal and real models of computation.

IDEAL MODEL. In the ideal model, there is a trusted party that computes the functionality f based on the inputs handed to it by the player. Let there be N parties $P_1, \ldots, P_N$ where arbitrary (possibly intersecting) subsets of n parties may engage in an arbitrary (polynomial) number of concurrent sessions. Let $\mathcal{I} \subset N$ denote the subset of corrupted parties controlled by the adversary. An execution in the ideal model with an adversary with auxiliary input z corrupting parties $\mathcal{I}$ proceeds as follows:

Inputs: The inputs of the parties $P_1, \ldots, P_N$ are respectively determined by probabilistic polynomial time Turing machines $M_1, \ldots, M_N$ and the initial inputs $x_1, \ldots, x_N$ to these machines. As will be described below, these Turing machine determine the input values to be used by the different parties in the protocol executions. These input values are computed from the initial input, the current session number and outputs that were obtained from executions that have already concluded. Note that the number of previous outputs ranges from zero (when no previous outputs have been obtained) to some polynomial in λ that depends on the number of sessions initiated by the adversary.

Session initiation: The adversary initiates a new session by sending a $(\mathsf{start\text{-}session}, P_i)$ to the trusted party. If $P_i \notin \mathcal{I}$, then the trusted party sends $(\mathsf{start\text{-}session}, s)$ to P_i, where s is the index of the session.

Honest parties send inputs to trusted party: Upon receiving $(\mathsf{start\text{-}session}, s)$ from the trusted party, honest party P_i applies its input-selecting machine M_i to its initial input

x_i, the session number s and its previous outputs, and obtains a new input $x_{i,j}$.[4] P_i then sends $(s, x_{i,j})$ to the trusted party.

Corrupted parties send inputs to trusted party: Whenever the adversary wishes, it may send a message $(s, x'_{i,j})$ to the trusted party for any $x'_{i,s} \in \{0,1\}^\lambda$ of its choice, on behalf of a corrupted party P_i. It can send the pairs $(s, x'_{i,s})$ in any order it wishes and can also send them adaptively. The only limitation is that for any s, at most one pair indexed by s can be sent to the trusted party on behalf of P_i.

Trusted party answers corrupted parties: When the trusted party has received messages $(s, x'_{i,j})$ from a set of n parties $P_{\ell_1}, \ldots, P_{\ell_n}$ (where $\ell_1, \ldots, \ell_n \in [N]$), it sets $\vec{x}'_s = (x'_{\ell_1,s}, \ldots, x'_{\ell_n,s})$. It then computes $f(\vec{x}'_s)$ and sends $(s, f_{\ell_i}(\vec{x}'_s))$ to party P_{ℓ_i} for every $\ell_i \in \mathcal{I}_s$, where $\mathcal{I}_s \subseteq \mathcal{I}$ denotes the set of corrupted parties in session s. Note that $\mathcal{I}_s$ must be such that $|\mathcal{I}_s| < n$.

Adversary instructs the trusted party to answer honest parties: When the adversary sends a message of the type $(\mathsf{send\text{-}output}, s, \ell_i)$ to the trusted party, the trusted party sends $(s, f_{\ell_i}(\vec{x}'_s))$ to party P_{ℓ_i}.

Outputs: Each honest party P_i always outputs the values $f_i(\vec{x}'_s)$ that it obtained from the trusted party. The adversary may output an arbitrary (probabilistic polynomial-time computable) function of its initial-input and the messages obtained from the trusted party.

Let $\mathcal{S}$ be a non-uniform probabilistic polynomial-time machine (representing the ideal-model adversary). Then, the ideal execution of f with security parameter λ, input selecting machines $M = M_1, \ldots, M_N$, initial inputs $\vec{x} = (x_1, \ldots, x_N)$ and auxiliary input z to $\mathcal{S}$,

[4] Specifically, in the first session, $x_{i,1} = M_i(x_i, 1)$. In the later sessions s, $x_{i,s} = M_i(x_i, s, y_{i,1}, \ldots, y_{i,w})$, where w sessions have concluded and the outputs of P_i were $y_{i,1}, \ldots, y_{i,w}$.

denoted $\text{IDEAL}^N_{f,\mathcal{I},\mathcal{S},M}(\lambda, \vec{x}, z)$, is defined as the output vector of the honest parties and $\mathcal{S}$ from the above ideal execution.

REAL MODEL. We next consider the real model in which a real two-party protocol is executed (and there exists no trusted third party). Let $f, \mathcal{I}, N$ be as above and let Π be a multi-party protocol for computing f. Let A denote the adversary. Then, the real concurrent execution of Π with security parameter λ, input selecting machines $M = M_1, \ldots, M_N$, initial inputs $\vec{x} = (x_1, \ldots, x_N)$ and auxiliary input z to A, denoted $\text{REAL}^N_{\Pi,\mathcal{I},A,M}(\lambda, \vec{x}, z)$, is defined as the output vector of the honest parties and A, resulting from the following real-world process. The real world execution proceeds as follows. Each honest party P_i first chooses an identity id_i and registers it with F^N_{bp}. A corrupted party may choose to register its identity at any time it wishes, even after the computation begins. An honest party initiates a new session whenever it receives a start-session message from A. It then applies its input selecting machine to its initial input, the session number and its previously received outputs, and obtains the input for this session. Note that arbitrary (possibly intersecting) sets of n (out of N) player may be participating in concurrent executions of Π. The scheduling of all messages throughout the executions is controlled by the adversary. That is, the execution proceeds as follows. The adversary sends a message of the form $(s, \mathsf{msg}, P_i, P_j)$ to an honest party P_i on behalf of a corrupted party P_j. If that honest party is participating in session s, and this is the first message it has received from P_j, then it first retrieves the identity id_j of P_j from F^N_{bp}. It then adds (msg, P_i, P_j) to its view of session s and replies according to the instructions of Π and this view.

SECURITY DEFINITION. Having defined the ideal and real models of computation, we are now ready to give our formal security definition.

Definition 3.2 (Concurrent Self-Composition in Bounded Player Model). *Let $N = N(\lambda)$ be a polynomial and let f and Π be as above. Protocol Π is said to* securely compute

f under concurrent composition in the N-bounded player model if for every real model non-uniform probabilistic polynomial-time adversary A, there exists an ideal-model non-uniform probabilistic expected polynomial-time adversary $\mathcal{S}$, such that for all input-selecting machines $M = M_1, \ldots, M_N$, every $z \in \{0,1\}^*$, every $\vec{x} = (x_1, \ldots, x_N)$, and every $\mathcal{I} \subset [N]$,

$$\left\{\text{IDEAL}^{N}_{f,\mathcal{I},\mathcal{S},M}(\lambda, \vec{x}, z)\right\}_{\lambda\in\mathbb{N}} \mathsf{COMP} \left\{\text{REAL}^{F^N_{bp}}_{\Pi,\mathcal{I},A,M}(\lambda, \vec{x}, z)\right\}_{\lambda\in\mathbb{N}}$$

3.6 Impossibility Results in Bounded Player Model

In [Lin04], Lindell gave broad impossibility results for unbounded concurrent self-composition in the standard model. We observe that the impossibility result of [Lin04] carries over in a straightforward manner to bounded player model considered in the present work. Below, in what is largely an informal discussion, we elaborate on this observation. [Lin04, Lin03b, CKL03, KL11]

Lindell's impossibility result [Lin04] for unbounded concurrent self-composition in the standard model is obtained by combining three different results. Below, we will recall all of these results and discuss how each of them carry over to the bounded player model. First, we recall some basic definitions from [Lin04]. A large part of text below is taken verbatim from [Lin04].

Security under concurrent general composition. Informally speaking, concurrent general composition considers the case that a protocol ρ for securely computing some functionality f, is run concurrently (many times) with arbitrary other protocols π. In other words, the secure protocol ρ is run many times in a network in which arbitrary activity takes place. (Note that in contrast, in concurrent self-composition, we only consider security for concurrent executions of the same protocol ρ.) The formalize security in this setting, we

model the arbitrary network activity π as a "calling protocol" with respect to the functionality ff. That is, π is a protocol that contains, among other things, "ideal calls" to a trusted party that computes a functionality f. This means that in addition to standard messages sent between the parties, protocol π's specification contains instructions of the type "send the value x to the trusted party and receive back output y". Then, the real-world scenario is obtained by replacing the ideal calls to f in protocol π with real executions of protocol ρ. The composed protocol is denoted π^ρ and it takes place without any trusted help. Security is defined by requiring that for every protocol π that contains ideal calls to f, an adversary interacting with the composed protocol π^ρ (where there is no trusted help) can do no more harm than in an execution of π where a trusted party computes all the calls to f. This therefore means that ρ behaves just like an ideal call to f, even when it is run concurrently with any arbitrary protocol π. We refer the reader to [Lin04] for a formal security definition.

Concurrent general composition in the bounded player model. We note that security under concurrent general composition can be naturally defined in the bounded player model by considering an *a priori* bound on the total number of player in the system, in the same manner as in Definition Definition 3.2. More specifically, we will consider an *a priori* bound N on the total number of player in the system. Then, arbitrary (possibly intersecting) subsets of parties may be involved in unbounded concurrent executions of ρ, in the presence of arbitrary other protocols π. (Note that π can be at-most an N-party protocol.) Security is defined in the same manner as above.

Functionalities that enable bit transmission. Informally speaking, a functionality enables bit transmission if it can be used by the parties to send bits to each other. We now recall the formal definition from [Lin04].

Definition 3.3 (Bit-transmitting functionality)**.** *A deterministic functionality $f = (f_1, f_2)$ enables bit transmission from P_1 to P_2 if there exists an input y for P_2 and a pair of inputs x,*

x' for P_1 such that $f_2(x;y) \neq f_2(x';y)$. Likewise, f enables bit transmission from P_2 to P_1 if there exists an input x for P_1 and a pair of inputs y, y' for P_2 such that $f_1(x;y) \neq f_1(x;y')$. A functionality enables bit transmission if it enables bit transmission from P_1 to P_2 and from P_2 to P_1.

The above definition can be easily generalized to probabilistic functionalities, as well as to multi-party functionalities in a straightforward way. We refer the reader to [Lin04] for more details.

Extending Lindell's impossibility result to BP model. We now consider the three steps involved in the impossibility result in [Lin04], and briefly discuss why they carry over to the bounded player model.

Step 1: First, it is shown in [Lin04] that for every functionality f that enables bit transmission, security under unbounded concurrent self-composition is equivalent to security under concurrent *general* composition. That is, if f enables bit transmission, then f can be securely computed unbounded concurrent self-composition if and only if it can be securely computed under concurrent general composition.

We note that [Lin04] proves this (unconditional) result for two-party setting where only *one set* of parties run all of the protocol executions. As such, the result already works in the bounded player model.

Step 2: Next, we use the result of [Lin03b], where it is shown that security under concurrent general composition implies security in the universal composability framework [Can01]. This result is also unconditional, and in fact, also works in a setup model (such as a common reference string, etc).

Once again, we note that [Lin04] obtains this result even for the restrictive case where only *one set* of parties engage in two-party protocol executions (the adversary is as-

sumed to be static). As such, this result is also applicable to the bounded player model.

Step 3: Finally, one can use the result of Canetti et al. [CKL03] that shows a large class of functionalities for which UC security cannot be achieved. With respect to the bounded player model, we note that very recently, Kidron and Lindell [KL11] show that the results of [CKL03] can be extend to the bulletin-board certificate authority model, which is formalized in essentially the same manner as our bounded player model, in that the parties register their unique identities to a functionality. We note that the result in [KL11] already works when the number of parties are *a priori* bounded, as such it is applicable to our setting.

Combining these three steps, we can obtain broad impossibility results for concurrent self-composition in the bounded player model. In order to obtain the formal statement, let us first recall the class of functionalities Ψ for which concurrent general composition is shown to be impossible [Lin03b]. The following is taken verbatim from [Lin04, Lin03b].

1. Let $f : \{0,1\}^* \to \{0,1\}^*$ be a deterministic polynomial-time function that is (weakly) one-way. Then, the functionality $(x, \lambda) \to (\lambda, f(x))$ cannot be securely computed under concurrent general composition by any non-trivial protocol.

2. Let $f : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ be a deterministic polynomial-time functionality. If f depends on both parties' inputs, then the functionality $(x, y) \to (f(x, y), f(x, y))$ cannot be securely computed under concurrent general composition 2. Let $f : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^* \times \{0,1\}^*$ be a deterministic polynomial-time functionality and let $f = (f_1, f_2)$. If f is not completely revealing[5] then the functionality $(x, y) \to (f_1(x, y), f_2(x, y))$

[5]Informally, a functionality is completely revealing if one party can choose an input so that the output of the functionality will reveal the other party's input. See [Lin03b, Lin04] for details.

cannot be securely computed under concurrent general composition by any non-trivial protocol.

Further, let Φ be the set of all two-party functionalities that enable bit transmission. Then, we obtain the following result:

Corollary 1. *Let f be a functionality in $\Phi \cap \Psi$. Then f cannot be securely computed under unbounded concurrent self composition by any non-trivial protocol.*

Remark. We note that the above discussion is relevant to the "fixed-roles" setting where the parties play the same roles in each session in the concurrent self-composition setting. If we allow interchangeable roles, then as shown in [Lin04], essentially all functionalities are impossible to realize. We refer the reader to [Lin04] for more details.

Chapter 4

Topology-Hiding Multi-Party Computation

4.1 Introduction

Secure multi party computation (MPC) has occupied a central role in cryptography since its inception in the '80s. The unifying question can be stated simply:

> *Can mutually distrusting parties compute a function of their inputs, while keeping their inputs private?*

Classical feasibility results [Yao82b, GMW87, BGW88] paved the way for a plenitude of research which has over time simplified, optimized and generalized the original foundational constructions. Some particularly rich lines of work include improving the complexity (round/communication/computation) of MPC protocols (e.g., [FY92, DN07, DIK10] and many more) and striving to achieve security in the more difficult (but realistic) setting where the adversary may execute many instantiations of the protocol along with other protocols (e.g., [PR03, Pas04b, CLOS02, Can01] and many more).

Common to nearly all prior work, however, is the assumption that the parties are all capable of exchanging messages with one another. That is to say, most work in the MPC literature assumes that the underlying network topology is that of a complete graph. This

is unrealistic as incomplete or even sparse networks are much more common in practice. Moreover, the comparably small body of MPC work that deals with incomplete networks concerns itself with the classical goal of hiding the parties' inputs. In light of the growing impact of networking on today's world, this traditional security goal is insufficient. Consider, for example, the graph representing a social network: nodes representing people, edges representing relationships. Most computation on social networks today is performed in a centralized way—Facebook, for example, performs computations on the social network graph to provide popular services (e.g., recommendations that depend on what "similar" people liked). However, in order to provide these services Facebook must "know" the entire graph.

One could imagine wanting to perform such a computation in a *distributed* manner, where each user communicates only with their own friends, without revealing any additional information to third parties (there is clearly wide interest in this type of service—Diaspora*, a project that was expressly started to provide "Facebook-like" functionality in a more privacy-preserving manner, raised over $200,000 in 40 days via Kickstarter).

Another motivating example is the recent push by US auto safety regulators towards vehicle-to-vehicle communication, which envisions dynamic networks of communicating vehicles; many "global" computations seem to be interesting in this setting (such as analysis of traffic patterns), but leaking information about the structure of this network could have severe privacy implications.

The rise of the "internet of things", connected by mesh networks (networks of nodes that communicate locally with each other) is yet another case in which the topology of the communication network could reveal private information that users would prefer to hide.

It is with such applications in mind that we initiate the study of *topology-hiding MPC* in the computational setting. We consider the fundamental question:

Can an MPC protocol computationally hide the underlying network topology?

4.1.1 Our Contributions

Formally Defining Topology-Hiding MPC: In keeping with tradition we give both an indistinguishability game-based definition and a simulation-based one. Very briefly, in the game-based definition the adversary corrupts $A \subset V$ and sends two network topologies G_0, G_1 on vertices V. These graphs must be so that the neighborhoods of A are the same. The challenger then picks G_b at random and returns the collective view of the parties in A resulting from the execution of the protocol on G_b. The adversary outputs b' and wins if $b' = b$. We say a protocol is *secure against chosen topology attack* (IND-CTA−secure) if no PPT adversary can win the above game with probability negligibly greater than if it simply guesses b'.

We then give a simulation-based definition of security using the UC framework. We define an ideal functionality $\mathcal{F}_{\text{graph}}$ and say that a protocol is "topology hiding" if it is UC secure in the $\mathcal{F}_{\text{graph}}$−hybrid model. The functionality $\mathcal{F}_{\text{graph}}$ models a network with private point-to-point links (private in the sense that the adversary does not know the network topology). It receives G as input, and outputs to each party a description of its neighborhood. It then acts as an "ideal communication channel" allowing neighbors to send messages to each other. For more details on $\mathcal{F}_{\text{graph}}$ and the motivations behind our definition see the discussion below. Finally, we relate the two new notions by proving that simulation-based security implies game-based security.

Feasibility of topology-hiding MPC against semi-honest adversary:

Theorem 1. *Assume trapdoor permutations exist. Let G be the underlying network graph and d a bound on the degree of every vertex in G. Then every multiparty functionality may be realized by a topology hiding MPC protocol which is secure against a semi-honest adversary who does not corrupt all parties in any $k-$neighborhood of the underlying network graph where k is such that $d^k = \text{poly}(\lambda)$.*

We point out that many naturally occurring graphs satisfy $d^D = poly(\lambda)$ where D is the diameter. Examples of such graphs include binary trees, hypercubes, expanders, and generally graphs with relatively high connectivity and low degree such as those occurring from social networks. For such graphs Theorem 1 is a feasibility result against a general semi-honest adversary.

Impossibility in fail-stop model:

Theorem 2. *There exists a functionality $\mathbb{F}$ and a network graph G such that realizing $\mathbb{F}$ while hiding G is impossible.*

Our proof uses the ability of the adversary to disconnect G with his aborts; we then prove this is inherent.

Sufficient conditions in fail-stop model:

Theorem 3. *Assume TDP exist. Every multiparty functionality may be realized by a topology hiding MPC protocol which is secure against a fail-stop adversary who does not corrupt all parties in any neighborhood of the underlying network graph and who's aborts do not disconnect the graph.*

4.1.2 Related Work

MPC on Incomplete Network Topologies One line of work which is in exception to the above began with Dolev's paper [Dol82] proving impossibility of Byzantine agreement

on incomplete network topologies with too low connectivity. Dwork et. al. [DPPU88] coined the term "almost everywhere Byzantine agreement" to be a relaxed variant of Byzantine agreement where agreement is reached by *almost* all of the honest parties. Garay and Ostrovsky [GO08] used this to achieve almost everywhere (AE) MPC. Recently [CGO12] gave an improved construction of AE Byzantine agreement translating to an improved feasibility result for AE MPC. These works are all in the information theoretic setting. We refer the curious reader to [CGO12] and the references therein for more details.

Another recent line of work is that of Goldwasser et. al. [BGT13] who consider MPC while minimizing the communication *locality*, the number of parties each player must exchange messages with. Their work is in the cryptographic setting and they give a meaningful upper bound on the locality and overall communication complexity. Their work does not address the notion of hiding the graph. Moreover they employ techniques such as leader election which seem inherently *not* to hide the graph.

Finally, we mention the two classical techniques of *mix-net* and *onion routing*. The mix-net technique introduced by Chaum [Cha81] uses public key encryption to implement a "message transmit" scheme allowing a sender and receiver to using in a message transmit using an additional shuffling mechanism. The onion routing technique [RR99, RSG98] and its extensions is a useful technique for anonymous communication over the Internet. Its basic idea is establishing paths of entities called proxies that know the topology in order to transmit massages.

Topology-Hiding MPC: While most of the cryptographic MPC literature disregards the interplay between multiparty computation and networking, the above works give a relatively satisfactory view of the landscape. Hiding the topology of the network in secure computation, on the other hand, is somewhat of a novel goal. The only work in the MPC literature of which

we are aware that has considered this question is that of Hinkelmann and Jakoby [HJ07] who focused on the information theoretic setting. Their main observation can be summarized:

> *If vertices v and w are not adjacent in G then P_v cannot send a message to P_w without some intermediate P_z learning that it sits between P_v and P_w.*

They use this observation to prove that any MPC protocol in the information theoretic setting must inherently leak information about G to an adversary. They do, however, prove a nice positive result: given some minimal amount of network information to leak (formalized as a routing table of the network), one can construct an MPC protocol which leaks no further information.

Their work left open the interesting possibility that, using cryptographic techniques, one could construct an MPC protocol which (computationally) hides the network topology. In this work we explore this possibility and emerge with a positive result.

4.2 Topology-Hiding Security

4.2.1 Graph Related Notions

We consider a network modeled by a directed graph $G = (V, E)$ that is not fully connected and where $V = \{P_1, \ldots, P_m\}$ is identified with a set of $m = \text{poly}(\lambda)$ parties. As usual in MPC, some of these parties might be corrupt, in which case they are controlled by a PPT adversary $\mathcal{A}$. For $v \in V$ we define the *neighborhood of* v by

$$N(v) = \{w \in V : (v, w) \in E\}.$$

Similarly, the *closed neighborhood of* v, is $\mathrm{N}[v] = N(v) \cup \{v\}$. We sometimes refer to $\mathrm{N}[v]$ as the closed $1-$neighborhood of v, and for $k \geq 1$ we define the $k-$neighborhood of v as

$$N^{k+1}[v] = \bigcup_{w \in N^k(v)} \mathrm{N}[w].$$

4.2.2 Topology Hiding Security – The Game-Based Version

Our first definition of topology-hiding security is formalized as a game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. The basic structure of the game fits several types of adversarial behaviors, e.g., semi-honest, fail-stop, and malicious, thus, we do not emphasize the exact behavior of the adversary during the execution of the protocol.

- Setup: Let $\mathcal{G}$ be a set of graphs. For simplicity, we assume that all $G \in \mathcal{G}$ have the same vertex set $V = \{P_1, \ldots, P_m\}$. Let Π be an $m-$party protocol which can be run on any $G \in \mathcal{G}$.

- $\mathcal{A}$ chooses a corrupt subset $S \subset V$, and gives inputs x_i to the parties $P_i \in S$. Next, $\mathcal{A}$ picks graphs $G_0, G_1 \in \mathcal{G}$ such that the local neighborhoods of S in G_0 and G_1 are identical. That is, $\mathrm{N}_{G_0}[S] = \mathrm{N}_{G_1}[S]$ (equality of graphs). It sends $\big(S; G_0, G_1; \{x_i\}_{i \in S}\big)$ to $\mathcal{C}$.

- Now C chooses a random $b \in \{0, 1\}$ and runs Π on communication graph G_b, where each honest P_i gets a random x_i as input and each dishonest party gets the input prescribed by $\mathcal{A}$. During the execution, $\mathcal{A}$ controls the parties in S, and afterwards it collects their views into the random variable $\mathbf{VIEW}_{S,\{x_i\},G_b}$.

- Finally $\mathcal{A}$ outputs $b' \in \{0, 1\}$. $\mathcal{A}$ wins if $b' = b$, otherwise $\mathcal{A}$ loses.

Definition 4.1. *We say that an MPC protocol* Π *is* Indistinguishable under Chosen Topology Attack *(*IND-CTA secure*) over* $\mathcal{G}$ *if for any PPT adversary* $\mathcal{A}$*,*

$$\left| Pr\big(\mathcal{A} \text{ wins}\big) - \frac{1}{2} \right| = \mathbf{negl}(\lambda).$$

4.2.3 UC Security

Universal composability (UC) [Can01] refers to a framework for proving protocol security. It is a strong, simulation based notion of security. Intuitively, a protocol is UC secure if any attack (concurrent, man-in-the-middle, etc.) on the protocol can be launched with equal success against an "idealized" version of the protocol where a trusted party receives parties' inputs, computes the functionality, and returns the outputs. This is formalized using the real/ideal paradigm, as described below. The strength of the UC model provides a desirable "ease of mind" in protocol design. The UC composition theorem [Can01] proves that UC secure protocols are secure under composition (with themselves and other protocols). This allows one, during security proofs, to "abstract away" subprotocols which are UC secure and focus only on the new ingredients. This "protocol stacking" technique can result in the security of protocols for highly complex functionalities following from the security of very simple ones.

In this chapter, we strive to construct an MPC scheme which hides the topology of the underlying communication network. We encapsulate this security property nicely using the UC framework. We will say that a protocol is *topology hiding* if it is UC secure in some $\mathcal{F}_{\text{graph}}$−hybrid model, where $\mathcal{F}_{\text{graph}}$ is a low level functionality, which gives parties a list of neighbors and provides message transmission between neighbors. We find this definition appealing for two reasons. First, we feel it thoroughly captures the level of security we are looking for as the $\mathcal{F}_{\text{graph}}$ functionality does not reveal any global information about the graph

to anyone. Secondly, we feel that it is reasonable to expect such a functionality be offered by a network. Indeed, it seems likely that such a functionality could be realized, say using special hardware. We now discuss the notions involved in defining UC security.

The Environment. UC security requires that an outside observer not be able to distinguish a real protocol execution from an ideal one, even if it is able to corrupt a subset of the parties. This outside observer is modeled as *the environment*, $\mathcal{Z}$. So as not to underestimate the control that $\mathcal{Z}$ has on the system, we assume that $\mathcal{Z}$ can set all parties' inputs, and read all outputs. Furthermore, $\mathcal{Z}$ has the ability to corrupt sets of parties via an adversary $\mathcal{A}$ that it controls. Once a party is corrupted, $\mathcal{Z}$ sees its entire view and gains the ability to control its behavior. What $\mathcal{Z}$ does not see, however, is the communication between the uncorrupted parties.

The Real World. The real world contains m parties who execute a protocol Π, the environment $\mathcal{Z}$, and an adversary $\mathcal{A}$, who is controlled by $\mathcal{Z}$. As mentioned above, $\mathbb{Z}$ sets the inputs and receives the outputs of all parties, and communicates with $\mathcal{A}$ who corrupts a subset of the parties. The uncorrupted parties follow the protocol, while corrupted parties are completely controlled by $\mathcal{A}$, and all of the communication to and from these parties is seen by $\mathcal{Z}$.

The Ideal World. The ideal world contains m "dummy" parties, an ideal functionality $\mathcal{F}$, the environment $\mathbb{Z}$, and an "ideal adversary" or simulator, §. The parties in this world are dummy parties because they simply pass their input to, and receive output from $\mathcal{F}$. The idea is that $\mathcal{F}$ offers the same functionality as the protocol Π in the real world, and yet is computed by a trusted third party and so offers little risk of a security breach. Just as in the real world, $\mathcal{Z}$ sets the inputs and will get the outputs of all players, and just as in the real world, $\mathcal{Z}$ can corrupt a subset of the parties, at which point, complete control over these

parties is given to $\mathcal{S}$. The goal of $\mathcal{S}$ is to simulate the view that the corrupt parties would have in the real world by executing Π. The challenge is that in the ideal world, the honest parties do not interact with the corrupt parties, and so $\mathcal{S}$ must generate this interaction on its own. Moreover, $\mathcal{S}$'s method must work for all choices of parties' inputs by $\mathcal{Z}$. $\mathcal{S}$ is allowed to communicate with $\mathcal{F}$ and may run it on inputs of its choosing.

Hybrid Worlds. As mentioned above, when proving UC security of a protocol, the UC composition theorem allows one to abstract away subprotocols using the notion of a hybrid world. More specifically, suppose that a large protocol Π uses a simpler Π' as a subprotocol, and it is known that Π' is UC secure. Then in order to prove the UC security of Π, it suffices to prove that Π is UC secure in the $\mathcal{F}'-$hybrid model. This essentially ammounts to proving that a modified version of Π, where every execution of Π' is replaced by an ideal evaluation of the functionality $\mathcal{F}'$ by a trusted third party, is UC secure. $\mathcal{F}'$ appears in the real and ideal worlds as a functionality with whom parties can interact. $\mathbb{Z}$ sees only the communication of the corrupt parties with $\mathcal{F}'$, and in the ideal world, $\mathcal{S}$ may communicate with $\mathcal{F}'$ but the honest dummy parties still only communicate with $\mathcal{F}$, the idealized version of Π. This means that $\mathcal{S}$ must simulate the interaction that corrupt parties would have with $\mathcal{F}'$ in the real $\mathcal{F}'-$hybrid world.

Definition of UC Security Let $\text{REAL}_{\Pi,\mathcal{A},\mathcal{Z}}(\lambda, z)$ denote the output of environment $\mathcal{Z}$ with input z after an execution of the protocol Π with corrupt parties controlled with adversary $\mathcal{A}$. Let We denote by $\text{IDEAL}_{\mathcal{F},\mathcal{S}^{\mathcal{A}},\mathcal{Z}}(\lambda, z)$ be the output of $\mathcal{Z}$ winth the ideal adversary (simulator) $\mathcal{S}$ and functionality $\mathcal{F}$, with security parameter λ. We denote the simulator by $\mathcal{S}^{\mathcal{A}}$, works with the adversary $\mathcal{A}$ attacking the real protocol. Similarly, we let $\text{REAL}^{\mathcal{F}'}_{\Pi,\mathcal{A},\mathcal{Z}}(\lambda, z)$ and $\text{IDEAL}^{\mathcal{F}'}_{\mathcal{F},\mathcal{S}^{\mathcal{A}},\mathcal{Z}}(\lambda, z)$ denote the outputs of $\mathcal{Z}$ after and execution in the $\mathcal{F}'-$hybrid real and ideal worlds, respectively.

Definition 4.2 (UC Security). *We say that* Π UC securely realizes a functionality $\mathcal{F}$ *if there exists a PPT* $\mathcal{S}$ *such that for all PPT* $\mathcal{Z}$ *and* $\mathcal{A}$ *and inputs* z *to* $\mathcal{Z}$*,*

$$\left\{ \mathrm{IDEAL}_{\mathcal{F},\mathcal{S}^{\mathcal{A}},\mathcal{Z}}(\lambda, z) \right\}_{\lambda} \approx_c \left\{ \mathrm{REAL}_{\Pi,\mathcal{A},\mathcal{Z}}(\lambda, z) \right\}_{\lambda}.$$

Similarly, for an ideal functionality $\mathcal{F}'$*, we say that* Π UC securely realizes $\mathcal{F}$ in the $\mathcal{F}'-$hybrid model *if for all* z*,*

$$\left\{ \mathrm{IDEAL}^{\mathcal{F}'}_{\mathcal{F},\mathcal{S}^{\mathcal{A}},\mathcal{Z}}(\lambda, z) \right\}_{\lambda} \approx_c \left\{ \mathrm{REAL}^{\mathcal{F}'}_{\Pi,\mathcal{A},\mathcal{Z}}(\lambda, z) \right\}_{\lambda}.$$

4.2.4 Topology Hiding Security – The Simulation-Based Version

As we have already mentioned, our simulation-based security notion is the usual notion of UC security in a special hybrid model we call the $\mathcal{F}_{\mathrm{graph}}-$hybrid model. The $\mathcal{F}_{\mathrm{graph}}$ functionality (shown in Figure 4.1) takes as input the network graph from a special "graph party" P_{graph} and returns to each other party a description of their neighborhood. It then handles communication between parties, acting as an "ideal channel" functionality allowing neighbors to communicate with each other. This prevents the communication from passing through the environment (implicitly revealing the network topology).

We think of $\mathcal{F}_{\mathrm{graph}}$ as modeling the actual communication network. That is, whenever a protocol specifies that a party should send a message to one of its neighbors using $\mathcal{F}_{\mathrm{graph}}$, this corresponds to a real world party directly sending the message over the network. Since $\mathcal{F}_{\mathrm{graph}}$ provides local information about the graph to all corrupted parties, *any* ideal-world adversary must have access to this information as well (regardless of the functionality we are attempting to implement). To capture this, we define the functionality $\mathcal{F}_{\mathrm{graphInfo}}$, that is identical to $\mathcal{F}_{\mathrm{graph}}$ but contains only the initialization phase. For any functionality $\mathcal{F}$, we define a "composed" functionality $(\mathcal{F}_{\mathrm{graphInfo}}||\mathcal{F})$ that adds the initialization phase of $\mathcal{F}_{\mathrm{graph}}$

Participants/Notation:

This functionality involves all the parties $P_1, \ldots, P_m$ and a special graph party P_{graph}.

Initialization Phase:

Inputs: $\mathcal{F}_{\text{graph}}$ waits to receive the graph $G = (V, E)$ from P_{graph}.

Outputs: $\mathcal{F}_{\text{graph}}$ outputs $\text{N}_G[P_i]$ to each player $P_i \in V$.

Communication Phase:

Inputs: $\mathcal{F}_{\text{graph}}$ waits to receive from each P_i a message m_i consisting of a set of destination/data pairs to be sent to its neighbors $m_i = \{(\mathsf{pid}_j, m_{i,j})\}_{P_j \in \text{N}_G(P_i)}$.

Output: Once $\mathcal{F}_{\text{graph}}$ has received input from all parties, it computes the set

$$M_i = \{P_j \in \text{N}_G(v) : \mathcal{F}_{\text{graph}} \text{ received input pair } (\mathsf{pid}_i, m_{j,i}) \text{ from } P_j\}$$

and gives output $\{(\mathsf{pid}_j, m_{j,i})\}_{P_j \in M_i}$ to P_v. If $M_i = \emptyset$ for some $P_i \in V$ then P_i gets $\perp$ as output.

Figure 4.1: The functionality $\mathcal{F}_{\text{graph}}$.

to $\mathcal{F}$. We can now define topology-hiding MPC in the UC framework:

Definition 4.3. *We say that a protocol* Π *realizes a functionality* $\mathcal{F}$ *with* topology hiding security *if it UC-realizes* $(\mathcal{F}_{graphInfo}||\mathcal{F})$ *in the* $\mathcal{F}_{graph}$*–hybrid model.*

Note that this definition allows protocols and functionalities to depend on the graph (e.g., find a shortest path between two nodes with the same input, or count the number of triangles in the graph).

4.2.5 Topology Hiding Security Implies IND-CTA Security

We prove below that our simulation-based notion is at least as strong as our game-based definition. This means that our game-based impossibility result of Section 4.4.1 implies impossibility in the simulation based definition as well.

Claim 1. *For every functionality* $\mathcal{F}$ *that does not depend on the communication graph struc-*

ture, if Π *securely realizes* $\mathcal{F}$ *with topology-hiding security (under Definition 4.3) then* Π *is IND-CTA secure.*

Proof. Let Π be a topology-hiding secure-computation protocol with respect to Definition 4.3 and let G_0 and G_1 be two graphs. We consider two specific executions of Π on network topologies G_0 and G_1 with corrupt parties given the same inputs. We define random variables $(\mathrm{REAL}_{G_0}, \mathrm{IDEAL}_{G_0})$ and $(\mathrm{REAL}_{G_1}, \mathrm{IDEAL}_{G_1})$ as usual (we use REAL even though technically we are in the $\mathcal{F}_{\mathrm{graph}}-$hybrid model). We observe that IDEAL_{G_0} are IDEAL_{G_1} are identically distributed, as the output of the protocol is the same whether the protocol is executed on G_0 or G_1. It follows that if Π realizes $\mathcal{F}$ with topology hiding security then we have

$$\mathrm{REAL}_{G_0} \approx_c \mathrm{IDEAL}_{G_0} = \mathrm{IDEAL}_{G_1} \approx_c \mathrm{REAL}_{G_1} .$$

It follows that $\mathcal{A}$ cannot win the IND-CTA game with probability that is noticeably better than $1/2$ and so Π is IND-CTA secure. □

Note that the assumption that $\mathcal{F}$ not depend on the network topology is stronger than we need. Instead if $\mathcal{G}$ is such that the output of $\mathcal{F}$ is the same on all graphs in $\mathcal{G}$, then Π will be IND-CTA secure over $\mathcal{G}$.

4.3 Topology Hiding MPC Against Semi-Honest Adv

In this section we describe a protocol for topology-hiding MPC against a semi-honest adversary for a large class of graphs. Our main result is the following:

Theorem 4. *Let* d *be a bound on the degree of any vertex in* G*. Then for every* k *satisfying* $d^k = \mathrm{poly}(\lambda)$*, and any* $m-$*party functionality* $\mathcal{F}$*, there exists a protocol* Π *that topology hiding securely realizes* $\mathcal{F}$ *against a semi-honest adversary* $\mathcal{A}$ *that does not corrupt all parties in any closed* $k-$*neighborhood of* G*.*

Note that this gives us security against a general semi-honest adversary when the graph has constant degree and a logarithmic bound on the diameter (by setting k to be anything larger than the graph diameter). We point out that many natural families of graphs are of this sort, including binary trees, hypercubes, expanders and more. We also point out that it suffices to securely realize the broadcast functionality since UC secure MPC protocols can be easily compiled from UC broadcast protocols.

4.3.1 High-Level Protocol Overview of Our Basic Protocol

Below we give a top-down description of our basic broadcast protocol: one that is secure against adversaries that do not corrupt any complete 1-neighborhood in the graph (i.e., in every star there is at least one honest node). This basic protocol can then be used to construct a broadcast protocol that tolerates larger corrupt neighborhoods (more details of this transformation appear in Section **??**).

A Naïve Broadcast Protocol. To understand the motivation for the construction, first consider a naïve broadcast protocol for a single bit:

1. In the first round, the broadcaster sends the broadcast bit b to all of its neighbors. Every other party sends 0 to all of their neighbors.
2. In each successive round, every party computes the OR of all bits received from their neighbors in the previous round, and sends this bit to all neighbors.

After j rounds, every party at distance j or less from the broadcaster will be sending the bit b (this can be easily shown by induction); after $diam(G)$ rounds all parties will agree on the bit b.

This protocol realizes the broadcast functionality, but it is not topology-hiding for two main reasons: a party can tell how far it is from the broadcaster by counting the number of

rounds until it receives a non-zero bit (assume $b = 1$ for this attack), and it can tell in which direction the broadcaster lies by noting which neighbor first sent a non-zero bit.

Using Local MPC to Hide Topology. Our construction hides the sensitive information by secret-sharing it among the nodes in a local neighborhood. Essentially, our basic protocol replaces each node in the naïve protocol above with a secure computation between the node and all of its direct neighbors in the graph.

In order to communicate a bit between one local neighborhood and another, without revealing the bit to the vertex connecting the two neighborhoods, each local neighborhood generates a public/private key pair, for which the private key is secret-shared between the parties in the neighborhood. The input to each local MPC includes the private key shares. The output to each party is encrypted under the public key of the neighborhood represented by that party (i.e., of which the party is the center node).

Since we assume that no local neighborhood is entirely corrupted, the adversary does not learn any of the plaintext bits. In the final round (at which point the broadcast bit has "percolated" to all the neighborhoods in the graph). a secure computation is used to decrypt the bits and output the plaintext to all the parties.

This part of the protocol is formally specified as two separate functionalities, each instantiated using a local secure computation: the KeyGen functionality ($\mathcal{L}_{\text{KeyGen}}$), handles the generation and distribution of the public/private key-pair shares in each local neighborhood, and the "broadcast-helper" functionality ($\mathcal{L}_{\text{bc-helper}}$), handles the encryption/decryption and ORing of the bits. The details of the construction are in Section **??**.

Implementing Local MPC. To implement $\mathcal{L}_{\text{KeyGen}}$ and $\mathcal{L}_{\text{bc-helper}}$, the basic protocol uses a general MPC functionality, $\mathcal{L}_{\text{MPC}}$, that allows the local neighborhoods to perform secure

computation protocols (i.e., among parties connected in a star graph). Realizing $\mathcal{L}_{\text{MPC}}$ ammounts to constructing a compiler which transforms a standard MPC protocol which runs on a complete graph into one which runs on a star graph. We achieve this by having players in the star who are not connected pass messages to each other through the center. The messages are encrypted to ensure privacy. One subtle point is that the protocol must not leak how many players are in the local neighborhood, as parties are not supposed to learn the degrees of their neighbors. We sidestep this issue by having the center node "invent" fake nodes so that parties learn only that the degree is at most d, some public upper bound on the degree of any node in G. The functionality $\mathcal{L}_{\text{MPC}}$ is shown in Figure 4.2, and explicit protocol is shown in Section 4.3.2.

4.3.2 Topology Hiding Securely Realizing $\mathcal{L}_{\text{MPC}}$

The local MPC functionality $\mathcal{L}_{\text{MPC}}$ is shown in Figure 4.2. As we have already mentioned, it is sufficient to securely realize message passing between all parties in P_i's local neighborhood in the $\mathcal{F}_{\text{graph}}-$hybrid model. This is because, once all parties can send messages to each other, they can simply run their favorite UC secure MPC protocol as if the network topology is that of a complete graph. Note that as we are in the semi-honest model here, UC secure MPC does not require setup. We will use the constant round, protocol of [Pas04b], as it is UC secure against a semi-honest adversary (against general adversaries it is bounded concurrent secure).

For simplicity we describe only the protocol allowing P_j to securely send a message to $P_{j'}$ for $P_j, P_{j'} \in \text{N}[P_i]$. This protocol is very simple:

1. $P_{j'}$ generates a key pair and sends the public key to P_j through P_i;

2. P_j encrypts its message and sends the ciphertext back to $P_{j'}$ through P_i.

Graph Entry Phase:

Input: $\mathcal{L}_{\text{MPC}}$ receives the graph G from P_{graph}.

Output: $\mathcal{L}_{\text{MPC}}$ outputs $\text{N}[P_i]$ to each P_i.

MPC Phase (for all $P_i \in V$):

Input: $\mathcal{L}_{\text{MPC}}$ receives from P_i a $d-$party protocol Π_i and $\{x_{j,i}\}_{P_j \in \text{N}[P_i]}$, P_i's inputs to the protocols Π_j of its neighbors.

Computation: $\mathcal{L}_{\text{MPC}}$ simulates each Π_i with inputs $\{x_{i,j}\}_j$ obtaining outputs $\{y_{i,j}\}_j$.

Output: $\mathcal{L}_{\text{MPC}}$ gives $\{y_{i,j}\}_j$ to each $P_j \in \text{N}[P_i]$ as output.

Figure 4.2: The functionality $\mathcal{L}_{\text{MPC}}$.

Such a protocol naturally extends to allow all parties in $\text{N}[P_i]$ to exchange messages with each other (as long as P_i invents enough nodes to ensure that his neighbors do not learn his degree, but just the preselected bound d). As we mention above, this is sufficient for securely realizing $\mathcal{L}_{\text{MPC}}$ in the $\mathcal{F}_{\text{graph}}-$hybrid model.

Participants/Notation: This protocol involves players $P_i, P_j, P_{j'}$, $P_j, P_{j'} \in \text{N}[P_i]$, and allows P_j to send the message msg to $P_{j'}$. Let $(\mathbf{Gen}, \mathbf{Enc}, \mathbf{Dec})$ be a public key encryption scheme.

Input: P_i and $P_{j'}$ give no input, P_j gives input msg.

Protocol for Message Passing:

- P_j chooses a key pair $(\text{pk}, \text{sk}) \leftarrow \mathbf{Gen}(1^\lambda)$ and sends pk to P_i.
- P_i forwards pk to $P_{j'}$.
- $P_{j'}$ computes encryption $y = \mathbf{Enc}_{\text{pk}}(\mathsf{msg})$ and sends y to P_i.
- P_i forwards y to P_j.
- P_j decrypts $\mathsf{msg} = \mathbf{Dec}_{\text{sk}}(y)$.

Figure 4.3: The protocol $\Pi^{(i,j,j')}_{\text{msg-pass}}$.

Security of $\mathcal{L}_{\text{MPC}}$. The proof is very simple so we suffice it briefly describe $\mathcal{S}$, and leave checking that it accurately emulates $\mathcal{A}$'s view in the real world to the reader. Since Π is a UC secure MPC protocol on a complete graph, there exists a simulator $\mathcal{S}'$ who can replicate any adversary $\mathcal{A}$'s real-world view in the ideal world. The only difference between the view $\mathcal{S}'$ outputs and the view we need to output is that we must take into account that our messages are encrypted and passed through P_i. Therefore, $\mathcal{S}$ generates key pairs $\{(\text{pk}_{j,j'}, \text{sk}_{j,j'})\}_{P_j,P_{j'}\in \text{N}[P_i]}$, where P_j will use $\text{pk}_{j,j'}$ to send messages to $P_{j'}$ and computes encryptions of the messages in the view output by $\mathcal{S}$, and distributes them accordingly to the players. Security follows from the security of the encryption scheme.

4.3.3 The Functionalities $\mathcal{L}_{\text{KeyGen}}$ and $\mathcal{L}_{\text{bc-helper}}$

The functionality $\mathcal{L}_{\text{MPC}}$ of the previous section is a general functionality that compiles an MPC protocol Π on a complete graph into an analogous one which can be executed by the parties in $\text{N}[P_i]$, without compromising the security of Π, and also without leaking any information about the topology. We will be interested in two specific local functionalities, $\mathcal{L}_{\text{KeyGen}}$ and $\mathcal{L}_{\text{bc-helper}}$. These can be securely realized in the $\mathcal{F}_{\text{graph}}-$hybrid model by simply instantiating $\mathcal{L}_{\text{MPC}}$ with two specific MPC protocols.

Recall that our underlying idea is to replace the role of P_i in a usual broadcast protocol with an MPC to be performed by the parties in P_i's neighborhood. This will hide each player's distance from the broadcaster because even though the bit might have been received by P_i's neighborhood, it will not be known to any individual player. Our first functionality, $\mathcal{L}_{\text{KeyGen}}$ is useful towards this end. Intuitively, it generates a key pair (pk, sk) for the neighborhood $\text{N}[P_i]$ and gives pk to P_i and distributes secret shares of the secret key among P_i's neighbors. Our second functionality, $\mathcal{L}_{\text{bc-helper}}$ will allow the broadcast bit to spread from neighborhood to neighborhood once the neighborhoods have keys distributed according to

$\mathcal{L}_{\text{KeyGen}}$. The functionalities $\mathcal{L}_{\text{KeyGen}}$ and $\mathcal{L}_{\text{bc-helper}}$ are shown in Figure 4.4 and Figure 4.5, respectively. Both functionalities are local, meaning that they act in the same way on all of the closed neighborhoods in G. For simplicity, we only describe the functionalities' input/output behavior in one such closed neighborhood, $\text{N}[P_i]$, even though many copies of the same behavior are occurring at once: one for each closed neighborhood of G. Finally let $\mathcal{L}_{\text{KeyGen}}(i)$ and $\mathcal{L}_{\text{bc-helper}}(i)$ denote the copies of $\mathcal{L}_{\text{KeyGen}}$ and $\mathcal{L}_{\text{bc-helper}}$, respectively, which take place in $\text{N}[P_i]$

Participants/Notation: For simplicity, we describe only the impact on the parties in $\text{N}[P_i]$, even though the same is occurring for all closed neighborhoods in G. Let $(\mathbf{Gen}, \mathbf{Enc}, \mathbf{Dec})$ be a public key encryption scheme.

Graph Entry Phase: same as in $\mathcal{L}_{\text{MPC}}$

KeyGen Phase:

- $\mathcal{L}_{\text{KeyGen}}$ generates a key pair $(\text{pk}, \text{sk}) \leftarrow \mathbf{Gen}(1^\lambda)$.
- $\mathcal{L}_{\text{KeyGen}}$ computes random shares $\{\text{sk}_j\}_{P_j \in \text{N}[P_i]}$ such that $\bigoplus_j \text{sk}_j = \text{sk}$.

Output: $\mathcal{L}_{\text{KeyGen}}$ gives outputs $\big(\text{pk}, \{\text{sk}_i\}\big)$ to P_i, and sk_j to each $P_j \in \text{N}[P_i]$.

Figure 4.4: The functionality $\mathcal{L}_{\text{KeyGen}}$.

4.3.4 Realizing $\mathcal{F}_{\text{broadcast}}$ in $\mathcal{L}_{\text{MPC}}-$hybrid model

Our $\mathcal{L}_{\text{MPC}}-$hybrid protocol for broadcast, $\Pi_{\text{broadcast}}$ uses the ideal functionalities $\mathcal{L}_{\text{KeyGen}}$ and $\mathcal{L}_{\text{bc-helper}}$ described above. As mentioned in the previous section, these functionalities are obtained from $\mathcal{L}_{\text{MPC}}$ by instantiating $\mathcal{L}_{\text{MPC}}$ with specific MPC protocols. A description of $\Pi_{\text{broadcast}}$ is given in Figure 4.6. Note that $\Pi^r_{\text{broadcast}}$ is correct as long $r > \text{diam}(G)$, the diameter of the network graph G. Our statement and proof of security is below.

Claim 4.1. *The protocol $\Pi^r_{broadcast}$ UC securely realizes $\mathcal{F}_{broadcast}$ in the $\mathcal{F}_{graph}-$hybrid model as long as the network topology graph G is such that*

Participants/Notation: For simplicity, we describe only the impact on the parties in $\mathrm{N}[P_i]$, even though the same is occurring for all closed neighborhoods in G. For $P_j \in \mathrm{N}[P_i]$, let pk^j be the public key output to P_j by $\mathcal{L}_{\mathrm{KeyGen}}(j)$. Let sk^i_j denote P_j's share of sk^i (the secret key corresponding to pk^i), given as output by $\mathcal{L}_{\mathrm{KeyGen}}(i)$.

Graph Entry Phase: same as in $\mathcal{L}_{\mathrm{MPC}}$

Main Phase:

Input: $\mathcal{L}_{\text{bc-helper}}$ receives inputs:

- $\alpha_j \in \{\text{“cipher”}, \text{“plain”}\}$ from each $P_j \in \mathrm{N}[P_i]$;
- $\left(\mathrm{pk}^j, \mathrm{sk}^i_j\right)$ from each $P_j \in \mathrm{N}[P_i]$;
- encryptions $\{x_j\}_{P_j \in \mathrm{N}[P_i]}$ from P_i, where $x_j = \mathbf{Enc}_{\mathrm{pk}^i}(b_j)$ for a bit $b_j \in \{0,1\}$.

The first input α_j is a tag which determines whether $\mathcal{L}_{\text{bc-helper}}$ outputs ciphertexts or plaintests. If all parties do not agree on α_j, $\mathcal{L}_{\text{bc-helper}}$ halts giving no output.

Computation:

- $\mathcal{L}_{\text{bc-helper}}$ reconstructs the secret key $\mathrm{sk}^i = \bigoplus_{P_j \in \mathrm{N}[P_i]} \mathrm{sk}^i_j$;
- $\mathcal{L}_{\text{bc-helper}}$ decrypts the bits $b_j = \mathbf{Dec}_{\mathrm{sk}^i}(x_j)$;
- $\mathcal{L}_{\text{bc-helper}}$ computes $b = \bigvee_{P_j \in \mathrm{N}[P_i]} b_j$.

Output:

- If $\alpha_j =$ “cipher” for all $P_j \in \mathrm{N}[P_i]$ then $\mathcal{L}_{\text{bc-helper}}$ outputs $y_j = \mathbf{Enc}_{\mathrm{pk}^j}(b)$ to each P_j.
- If $\alpha_j =$ “plain” for all $P_j \in \mathrm{N}[P_i]$ then $\mathcal{L}_{\text{bc-helper}}$ outputs b to each P_j.

Figure 4.5: The functionality $\mathcal{L}_{\text{bc-helper}}$.

1. *Diameter*$(G) < r$;

2. $\mathcal{A}$ *does not corrupt any entire closed neighborhood of* G.

Input: P_{graph} inputs the graph G, each P_i inputs a bit $b_i \in \{0,1\}$.

KeyGen: Parties call $\mathcal{L}_{\text{KeyGen}}$ and each P_i receives $\mathrm{N}[P_i]$ and $\left(\mathrm{pk}^i, \{\mathrm{sk}_i^j\}_{P_j \in \mathrm{N}[P_i]}\right)$.

Main Computation:

- Each P_i sets $x_{i,j}^0 = \mathbf{Enc}_{\mathrm{pk}^i}(b_i)$ for each $P_j \in \mathrm{N}[P_i]$.
- For $c = 1, \ldots, r-1$, parties call $\mathcal{L}_{\text{bc-helper}}$:
  - P_i gives input $\left\{\text{“cipher”};\ \left(\mathrm{pk}^i, \mathrm{sk}_i^i\right);\ \{x_{i,j}^{c-1}\}_{P_j \in \mathrm{N}[P_i]}\right\}$ to $\mathcal{L}_{\text{bc-helper}}(i)$;
  - For each $P_j \in \mathrm{N}[P_i]$, P_i gives input $\left\{\text{“cipher”};\ \left(\mathrm{pk}^i, \mathrm{sk}_i^j\right)\right\}$ to $\mathcal{L}_{\text{bc-helper}}(j)$;
  - P_i receives output $x_{i,j}^c$ from $\mathcal{L}_{\text{bc-helper}}(j)$ for all $P_j \in \mathrm{N}[P_i]$.
- Finally, parties call $\mathcal{L}_{\text{bc-helper}}$:
  - P_i gives input $\left\{\text{“plain”};\ \left(\mathrm{pk}^i, \mathrm{sk}_i^i\right);\ \{x_{i,j}^{r-1}\}_{P_j \in \mathrm{N}[P_i]}\right\}$ to $\mathcal{L}_{\text{bc-helper}}(i)$;
  - For each $P_j \in \mathrm{N}[P_i]$, P_i gives input $\left\{\text{“plain”};\ \left(\mathrm{pk}^i, \mathrm{sk}_i^j\right)\right\}$ to $\mathcal{L}_{\text{bc-helper}}(j)$;
  - P_i receives the bit $b_{i,j}^*$ as output from $\mathcal{L}_{\text{bc-helper}}(j)$.

Output: P_i outputs $b_i^* = \bigvee_j b_{i,j}^*$.

Figure 4.6: The $(\mathcal{L}_{\text{KeyGen}} || \mathcal{L}_{\text{bc-helper}})$-hybrid protocol $\Pi_{\text{broadcast}}^r$.

Simulator. Consider a corrupt party P_i. $\mathcal{S}$ simulates P_i's view as follows:

1. **KeyGen:** $\mathcal{S}$ generates $(\mathrm{pk}^i, \mathrm{sk}^i) \leftarrow \mathbf{Gen}(1^\lambda)$. When the parties call $\mathcal{L}_{\text{KeyGen}}$, $\mathcal{S}$ returns pk^i to P_i and random strings r_i^j for each $P_j \in \mathrm{N}[P_i]$, instead of shares of P_j's secret key.

2. **Main Computation:** As output to each of the first $r-1$ calls to $\mathcal{L}_{\text{bc-helper}}$, $\mathcal{S}$ gives output $\{x_{i,j}^c\}_{j,c}$ to P_i, where $x_{i,j}^c = \mathbf{Enc}_{\mathrm{pk}^i}(0^\lambda)$ to P_v. To compute the output of the last call of $\mathcal{L}_{\text{bc-helper}}$, $\mathcal{S}$ inputs b_i and all other corrupt parties' input bits to $\mathcal{F}_{\text{broadcast}}$ receiving b^* which it returns to P_i.

Hybrid Argument.

H_0 – This is the real execution of $\Pi^r_{\text{broadcast}}$. Namely, each environment first runs $\mathcal{L}_{\text{KeyGen}}$, after which each P_i has key data $\left(\text{pk}^i, \{\text{sk}_i^j\}_{P_j \in \text{N}[P_i]}\right)$. Then parties enter the loop, running $\mathcal{L}_{\text{bc-helper}}$ r times. Initially, parties enter their secret bit and the key data received from $\mathcal{L}_{\text{KeyGen}}$. In each subsequent call to $\mathcal{L}_{\text{bc-helper}}$, the output from the previous call is also given as input. Finally, P_i receives many copies of the same bit b_i^* as output from the last call to $\mathcal{L}_{\text{bc-helper}}$, and P_i outputs this bit. The view of P_i therefore consists of the following:

1. input $b_i \in \{0,1\}$, output $b_i^* \in \{0,1\}$;
2. key data $\left(\text{pk}^i, \{(\text{sk}_i^j)\}_{P_j \in \text{N}[P_i]}\right)$;
3. encryptions $\left\{x_{i,j}^c\right\}_{P_j \in \text{N}[P_i]}^{c=0,\ldots,r-1}$.

Let $B \subset V$ be the set of bad parties corrupted by $\mathcal{A}$. The view of the adversary is

$$\left\{\left(b_i, b_i^*; \text{pk}^i, \left\{\text{sk}_i^j\right\}_{P_j \in \text{N}[P_i]}; \left\{x_{i,j}^c\right\}_{j,c}\right)\right\}_{P_i \in B}.$$

H_1 – This is the same as the above experiment except the secret key shares are replaced by random strings. The resulting view is

$$\left\{\left(b_i, b_i^*; \text{pk}^i, \{r_i^j\}_j; \{x_{i,j}^c\}_{j,c}\right)\right\}_{P_i \in B}.$$

As the secret key sk^i is secret shared among $\text{N}[P_i]$ using a $\left|\text{N}[P_i]\right|-\text{out}-\text{of}-\left|\text{N}[P_i]\right|$ secret sharing scheme, and $\mathcal{A}$ does not corrupt all of $\text{N}[P_i]$, we have that $H_1 \approx H_0$.

H_2 – This is identical to H_1 except that all of the encryptions $x_{i,j}^c$ are changed to encryptions of 0. The resulting view is exactly the view of the ideal world adversary, and is indistinguishable from the view in H_1 by semantic security of the encryption scheme.

4.3.5 Allowing for Corruption of Whole Neighborhoods

Our protocol $\Pi^{r}_{\text{broadcast}}$ from the previous section successfully realizes the broadcast functionality while hiding the topology of the graph so long as $\mathcal{A}$ does not corrupt any entire neighborhood of G. If $\mathcal{A}$ were to corrupt $\text{N}[P_i]$ for some i, our protocol immediately becomes insecure, as $\mathcal{A}$ would possess all of the shares of sk^i and so could simply decrypt all of the encrypted bits P_i receives and learn when the broadcast bit reaches P_i. In this section, we show how, given a protocol Π that is secure as long as $\mathcal{A}$ does not corrupt all parties in a $k-$neighborhood, one can construct another protocol Π' for the same functionality as Π, but is secure as long as $\mathcal{A}$ does not corrupt an entire $(k+1)-$neighborhood. The round complexity of Π' will be a constant times the round complexity of Π and so one can only repeat this process logarithmically many times.

The main ideas of this section are essentially the same as those in the previous section, and show that the technique for using local MPC to hide information as it spreads to all parties in the graph is actually quite general. Like $\Pi_{\text{broadcast}}$, our protocol Π' will be given in the $\mathcal{L}_{\text{MPC}}-$hybrid model, where we will use the ideal functionality $\mathcal{L}_{\text{KeyGen}}$. However, instead of using $\mathcal{L}_{\text{bc-helper}}$, we will use a similar but different local functionality, $\mathcal{L}_{\Pi\text{-next}}$, shown in Figure 4.7. Essentially, $\mathcal{L}_{\Pi\text{-next}}$ allows the role of P_i in Π to be computed using a local MPC by all of the parties in $\text{N}[P_i]$. Then the protocol Π' uses $\mathcal{L}_{\Pi\text{-next}}$ to execute Π except that each party's role in Π is computed using local MPC by its local neighborhood in Π'. This ensures that if Π is such that any adversary wishing to attack Π must corrupt an entire $k-$neighborhood, then any adversary wishing to attack Π' must corrupt an entire $(k+1)-$neighborhood.

Our hybrid protocol Π' is described in Figure 4.8. Our statement and construction of simulator are below. We leave out the hybrid argument as it is very similar to the one in Section 4.3.4

Participants/Notation: For simplicity, we describe only the impact on the parties in $\mathrm{N}[P_i]$, even though the same is occurring for all closed neighborhoods in G. For $P_j \in \mathrm{N}[P_i]$, let pk^j be the public key output to P_j by $\mathcal{L}_{\mathrm{KeyGen}}(j)$. Let sk^i_j denote P_j's share of sk^i (the secret key corresponding to pk^i), given as output by $\mathcal{L}_{\mathrm{KeyGen}}(i)$.

Graph Entry Phase: same as in $\mathcal{L}_{\mathrm{MPC}}$

Main Phase:

Input: $\mathcal{L}_{\Pi\text{-next}}$ receives inputs:

- a round number $c_j \in \{1, \ldots, r\}$ from each $P_j \in \mathrm{N}[P_i]$;
- $\left(\mathrm{pk}^j, \mathrm{sk}^i_j\right)$ from each $P_j \in \mathrm{N}[P_i]$;
- an encrypted transcript so far $\hat{T}_i^{c-1} = \left(x_i, \sigma_i; \{\hat{y}^\ell_{i,j}\}_j^{\ell \le c-1}\right)$ from P_i, where x_i and σ_i are P_i's input and randomness and $\hat{y}^\ell_{i,j} = \mathbf{Enc}_{\mathrm{pk}^i}(y^\ell_{i,j})$ is an encryption of the message P_j sent to P_i in the $\ell-$th round of Π.

If all parties don't agree on the round number, $\mathcal{L}_{\Pi\text{-next}}$ halts giving no output.

Computation:

- $\mathcal{L}_{\Pi\text{-next}}$ reconstructs the secret key $\mathrm{sk}^i = \bigoplus_{P_j \in \mathrm{N}[P_i]} \mathrm{sk}^i_j$;
- $\mathcal{L}_{\Pi\text{-next}}$ decrypts $y^\ell_{i,j} = \mathbf{Dec}_{\mathrm{sk}^i}(\hat{y}^\ell_{i,j})$ for all $P_j \in \mathrm{N}[P_i]$ and $\ell \le c-1$;
- $\mathcal{L}_{\Pi\text{-next}}$ computes the next message function of Π,

$$F^{\Pi}_{i,c}\Big(x_i, \sigma_i, \{y^\ell_{i,j}\}_{j,\ell}\Big) = \begin{cases} \{y^c_{j,i}\}_j, & c \le r-1 \\ z_i, & c = r \end{cases}$$

Output:

- If $c \le r-1$ then each $P_j \in \mathrm{N}[P_i]$ receives $\hat{y}^c_{j,i} = \mathbf{Enc}_{\mathrm{pk}^j}(y^c_{j,i})$ from $\mathcal{L}_{\Pi\text{-next}}$.
- If $c = r$ then $\mathcal{L}_{\Pi\text{-next}}$ outputs y_i to P_i.

Figure 4.7: The functionality $\mathcal{L}_{\Pi\text{-next}}$.

Input: P_{graph} inputs the graph G, each P_i inputs x_i, their input to Π.

KeyGen: Parties call $\mathcal{L}_{\text{KeyGen}}$ and each P_i receives $\text{N}[P_i]$ and $\left(\text{pk}^i, \{\text{sk}_i^j\}_{P_j \in \text{N}[P_i]}\right)$.

Main Computation:

- P_i initializes $\hat{T}_i^0$ to $(x_i, \sigma_i; \emptyset)$.
- For $c = 1, \ldots, r$, parties call $\mathcal{L}_{\Pi\text{-next}}$:
  - P_i gives input $\left\{c;\ \left(\text{pk}^i, \text{sk}_i^i\right);\ \hat{T}_i^{c-1}\right\}$ to $\mathcal{L}_{\text{bc-helper}}(i)$;
  - For each $P_j \in \text{N}[P_i]$, P_i gives input $\left\{c;\ \left(\text{pk}^i, \text{sk}_i^j\right)\right\}$ to $\mathcal{L}_{\text{bc-helper}}(j)$;
  - P_i receives output $\hat{y}_{i,j}^c$ from $\mathcal{L}_{\text{bc-helper}}(j)$ for all $P_j \in \text{N}[P_i]$.
  - If $c \leq r-1$, P_i updates $\hat{T}_i^c$ to include the messages $\{\hat{y}_{i,j}^c\}_j$ he just received.

Output: When $c = r$, P_i receives z_i from $\mathcal{L}_{\Pi\text{-next}}(i)$, which it outputs.

Figure 4.8: The $(\mathcal{L}_{\text{KeyGen}} || \mathcal{L}_{\Pi\text{-next}})$-hybrid protocol Π'.

Claim 4.2. *The protocol Π' realizes the same functionality as Π. Moreover if Π realizes the functionality UC securely in the $\mathcal{F}_{graph}-$hybrid model as long as $\mathcal{A}$ does not corrupt an entire $k-$neighborhood of G, then Π' is UC secure in the $\mathcal{F}_{graph}-$hybrid model as long as $\mathcal{A}$ does not corrupt an entire $(k+1)-$neighborhood of G.*

Simulator. We construct a simulator $\mathcal{S}'$ which will make use of the simulator $\mathcal{S}$ for Π. Consider a corrupt party P_i. $\mathcal{S}'$ simulates P_i's view as follows:

1. **KeyGen:** $\mathcal{S}'$ generates $(\text{pk}^i, \text{sk}^i) \leftarrow \textbf{Gen}(1^\lambda)$ and random shares $\{\text{sk}_j^i\}_{P_j \in \text{N}[P_i]}$ such that $\bigoplus_j \text{sk}_j^i = \text{sk}^i$. When the parties call $\mathcal{L}_{\text{KeyGen}}$, $\mathcal{S}'$ returns $(\text{pk}^i, \text{sk}_i^i)$ to P_i and sk_j^i to P_j.

2. **Main Computation:** In order to simulate P_i's view we consider two cases:

   Case 1$-(P_i$ has at least one honest neighbor)**:** In this case $\mathcal{S}'$ simulates P_i's view by replacing all the messages P_i would receive with encryptions of 0.

   Case 2$-$(all of $\text{N}[P_i]$ is corrupt)**:** In this case $\mathcal{A}$ can reconstruct sk^i and so will be able to distinguish if $\mathcal{S}'$ sends encryptions of zero. However, $\mathcal{A}$ does not corrupt an

entire $(k+1)-$neighborhood of G which means the set $\{P_i \in V : \mathrm{N}[P_i]$ is corrupt$\}$ does not contain any $k-$neighborhood. Moreover, since each neighborhood in Π' plays the role of a player in Π, we can simulate the view of such P_i using the simulator $\mathcal{S}$ for Π. Specifically, $\mathcal{S}'$ internally runs $\mathcal{S}$ in order to simulate P_i's view in Π, and encrypts with pk^i to obtain P_i's view in Π'.

4.4 Topology Hiding MPC Against Fail-Stop Adv

In this section we turn to consider the case when corrupt parties must follow the protocol except that they may abort whenever the adversary instructs them to. Such an adversary is called fail-stop. We have two main results in this section. In Section 4.4.1 we give a general impossibility result, showing that any protocol that implements even a weak version of the broadcast functionality is not IND-CTA secure. Our proof crucially relies on the ability of the adversary to disconnect the communication graph by aborting with well placed corrupt parties. In Section 4.4.2 we show that this is inherent by transforming our broadcast protocol from the previous section into one which is secure against a fail-stop adversary who does not disconnect the graph with his aborts, and who does not corrupt (even semi-honestly) any $k-$neighborhood. We give a high level overview of our techniques of this section before proceeding to the details.

In Section 4.4.1 we consider a protocol Π realizing the broadcast functionality being executed on a line. The proof of the impossibility result is based on two simple observations. First, if some party aborts early in the protocol then honest parties' outputs cannot depend on b. Clearly, if P^* aborts before the information about b has reached him, then no information about b will reach the honest parties on the other side of P^*. This means that the outputs of *all* honest parties must be independent of b, otherwise an adversary would be able to corrupt another party P_{det} to act as a detective. Namely, $\mathcal{A}$ will instruct P_{det} to

play honestly and based on P_{det}'s output, $\mathcal{A}$ will be able to guess which side of P^* P_{det} is on. Second, if P^* aborts near the end of the protocol then all parties (other than P^*'s neighbors) must ignore this abort and output what they would have output had nobody aborted. Indeed, if P^* aborts with only k rounds remaining in the protocol, then there simply isn't time for honest parties of distance greater than k from P^* to learn of this abort. Therefore, all honest parties' outputs must be independent of the fact that P^* aborted, lest an $\mathcal{A}$ would be able to employ P_{det} to detect whether is within distance k of P^* or not. This difference in honest parties' outputs when P^* aborts early versus late means there is a round i^* such that the output distribution of P_{det} when P^* aborts in round i^* is distinguishable from P_{det}'s output distribution when P^* aborts in round $i^* + 1$. We take advantage of this by having two aborters P_1^* and P_2^* who abort in rounds i^* and $i^* + 1$. We prove that $\mathcal{A}$ will be able to distinguish the cases from when P_{det} is to the left of P_1^* with the case when he is to the right of P_2^* allowing $\mathcal{A}$ to win the IND-CTA game with non-negligible advantage.

In Section 4.4.2 we modify our broadcast protocol of Section 4.3 to be secure against a fail-stop adversary who does not disconnect the graph with his aborts. The idea is to run the semi-honest protocol $2m - 1$ times. Since the adversary can corrupt and abort with at most $m - 1$ parties we are guaranteed that the majority of the executions have no aborts. We ensure that $\mathcal{A}$ learns nothing from the outputs of the executions with aborts by holding off on giving any output until all $2m - 1$ executions have occurred. Then we use a final local MPC protocol to compute all outputs, select the majority and output this to all parties.

4.4.1 Impossibility Result

Definition 4.4. *We say that a protocol* Π weakly realizes the broadcast functionality *if* Π *is such that when all parties execute the protocol honestly, all parties output* $\bigvee x_i$ *where* x_i *is* P_i*'s input.*

Note that in weak broadcast, there are no guarantees on the behavior of honest parties if any of the parties deviates from the honest protocol.

Theorem 4.1. *There does not exist an IND-CTA secure protocol* Π *that weakly realizes the broadcast functionality in the fail-stop model.*

Let G be a line with m vertices. Namely, $G = (V, E)$ with $V = \{P_1, \ldots, P_m\}$ and $E = \{(P_i, P_{i+1})\}_{i=1,\ldots,m-1}$. Let Π be a protocol executed on G that weakly realizes the broadcast functionality where P_1 (the left most node) is the broadcaster (P_1 has input b, and the inputs to all other nodes is 0). Suppose Π has r rounds. We will show that Π cannot be IND-CTA secure.

Claim 2. *Let* $H_{v,b}$ *be the event that* P_v*'s output after executing* Π *matches the broadcast bit* b*. Let* E_i *be the event that the first abort occurs in round* i*. Then either* Π *is not IND-CTA secure, or there exists a bit* $b \in \{0,1\}$ *such that*

$$\left| Pr\big(H_{v,b} \big| E_{r-1}\big) - Pr\big(H_{v,b} \big| E_1\big) \right| \geq \frac{1}{2} - negl(\lambda)$$

for all honest P_v *whose neighbors do not abort.*

Proof. If some P^* aborts during the first round of Π then he disconnects the graph, making it impossible for the parties separated from P_1 to learn about b. These parties' outputs therefore must be independent of b, which implies that there exists a $b \in \{0,1\}$ such that $\Pr\big(H_b \big| E_1\big) \leq \frac{1}{2}$. If Π is to be IND-CTA secure then it must be that this inequality holds (with possibly a negligible error) for all honest parties. Otherwise an adversary could use the correlation between b and a party's output to deduce that this party is in the same connected component as P_1.

Formally, consider a fail-stop adversary $\mathcal{A}$ who corrupts three parties: the broadcaster P_1, aborter $P^* = P_{\lfloor \frac{m}{2} \rfloor}$ and detective P_{det}. $\mathcal{A}$ then submits (G_0, B_0), (G_1, B_1) to the challenger

where $G_0 = G_1 = G$ and $P_{\text{det}} = P_4$ in B_0 while $P_{\text{det}} = P_{m-1}$ in B_1. Note that $\mathcal{A}$'s neighborhoods are the same in G_0 and G_1. $\mathcal{A}$ instructs P^* to abort during the first round and observes P_{det}'s output. Since P_{m-1}'s output must be independent of b, if P_4's output depends in a non-negligible way on b, this will translate into an advantage for $\mathcal{A}$ in the CTA game.

Finally, note that $\Pr\big(H_{v,b}\big|E_{r-1}\big) = \Pr\big(H_{v,b}\big|\text{ no aborts}\big) = 1$ for all P_v which are not neighbors of P^*. The claim follows. □

Proof of Theorem 4.1. It follows from Claim 2 that there exists a pair $(i^*, b) \in \{1, \ldots, r\} \times \{0, 1\}$ such that

$$\Big|\Pr\big(H_{v,b}\big|E_{i^*}\big) - \Pr\big(H_{v,b}\big|E_{i^*+1}\big)\Big| \geq \frac{1}{2r} - \text{negl}(\lambda). \tag{4.1}$$

for all honest P_v who do not have an aborting neighbor. Furthermore, assume without loss of generality that $\Pr\big(H_{v,b}\big|E_{i^*}\big) > \Pr\big(H_{v,b}\big|E_{i^*+1}\big)$. We construct a fail stop adversary $\mathcal{A}$ who can leverage this fact to win the CTA game with non-negligible advantage.

Our adversary $\mathcal{A}$ corrupts four parties: the broadcaster P_1, two aborters $\big(P^*_{\text{L}}, P^*_{\text{R}}\big) = \big(P_{\lfloor \frac{m}{2} \rfloor - 1}, P_{\lfloor \frac{m}{2} \rfloor + 1}\big)$, and the detective P_{det}. $\mathcal{A}$ then submits (G_0, B_0) and (G_1, B_1) to the challenger where $G_0 = G_1 = G$ and B_0 has $P_{\text{det}} = P_4$ and B_1 has $P_{\text{det}} = P_{m-1}$. These graphs are shown in figure Figure 4.9. Note that these adversary structures have identical neighborhoods.

Now $\mathcal{A}$ guesses $(i^*, b) \in \{1, \ldots, r\} \times \{0, 1\}$. With non-negligible probability, (i^*, b) is such that inequality Equation 4.1 is satisfied. $\mathcal{A}$ gives b as input to P_1 and instructs P^*_{L} to abort on round i^*, P^*_{R} to abort on round i^*+1. Notice that since the two aborting parties are a distance 2 from each other, the information about P^*_{L}'s abort does not reach P^*_{R} by the time he aborts one round later. Therefore, the information about P^*_{L}'s abort does not reach any of the parties to the right of P^*_{R} at any point during the protocol. This means that if (G_0, B_0)

was chosen by the challenger, P_{det}'s output will be consistent with E_{i^*} whereas if (G_1, B_1) was chosen, P_{det}'s output will be consistent with E_{i^*+1}. $\mathcal{A}$ concludes by comparing P_{det}'s output bit to the broadcast bit b. If they are equal, $\mathcal{A}$ sends 0 to the challenger, otherwise he sends 1. The noticeable difference in output distributions ensured by i^* translates to a noticeable advantage for $\mathcal{A}$.

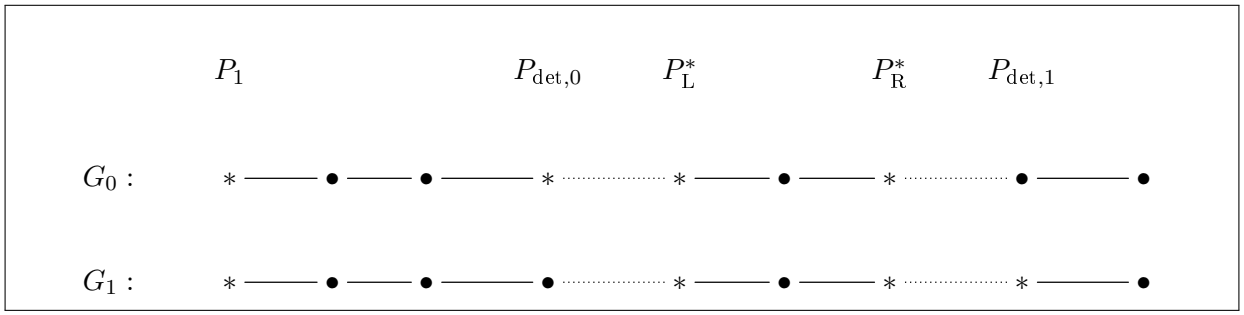

Figure 4.9: Graphs used by $\mathcal{A}$ in proof of Theorem 4.1.

□

4.4.2 Feasibility Result

In this section we informally comment that our semihonest protocol for broadcast from Section 4.3 can be compiled into a protocol which is secure against a fail-stop adversary when $\mathcal{A}$ cannot disconnect the graph with his aborts. The protocol is very simple and so we do not prove security or even formally define protocols.

The idea is to run the semihonest protocol many times. This ensures that a majority of the executions contain no aborts, and so the semi-honest correctness ensures that a majority of the executions give correct outputs. However, we change our protocol so that the outputs of the individual executions are given to each P_i in encrypted form, and only after all of them have been completed, N$[P_i]$ runs a local MPC to compute the majority of the outputs it has received. This ensures that all parties will receive the correct output.

If players simply ignore their neighbors' aborts and play on as if the aborts had not happened (this might involve making fake inputs for the aborted party so as not to alert its other neighbors of the abort), then the adversary gets no advantage from having parties abort other than ruining the current protocol execution.

Bibliography

[Bar01] Boaz Barak. How to go beyond the black-box simulation barrier. In *FOCS*, pages 106–115, 2001.

[Bar02] Boaz Barak. Constant-Round Coin-Tossing with a Man in the Middle or Realizing the Shared Random String Model. In *Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science*, FOCS '02, pages 345–355, 2002.

[BG02] Boaz Barak and Oded Goldreich. Universal arguments and their applications. In *IEEE Conference on Computational Complexity*, pages 194–203, 2002.

[BGT13] E. Boyle, S. Goldwasser, and S. Tessaro. Communication locality in secure multiparty computation - how to run sublinear algorithms in a distributed setting. In *TCC*, pages 356–376, 2013.

[BGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation (Extended Abstract). In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, STOC '88, pages 1–10, 1988.

[Blu86] Manual Blum. How to prove a theorem so no one else can claim it. In *International Congress of Mathematicians*, pages 1444–1451, 1986.

[BSMP91] Manuel Blum, Alfredo De Santis, Silvio Micali, and Giuseppe Persiano. Noninteractive zero-knowledge. *SIAM J. Comput.*, 20(6):1084–1118, 1991.

[Can01] Ran Canetti. Universally Composable Security: A New Paradigm for Cryptographic Protocols. In *Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science*, FOCS '01, pages 136–145, 2001.

[CF01] Ran Canetti and Marc Fischlin. Universally composable commitments. In *CRYPTO*, Lecture Notes in Computer Science, pages 19–40. Springer, 2001.

[CGGM00] Ran Canetti, Oded Goldreich, Shafi Goldwasser, and Silvio Micali. Resettable zero-knowledge (extended abstract). In *STOC*, pages 235–244, 2000.

[CGMO09] Nishanth Chandran, Vipul Goyal, Ryan Moriarty, and Rafail Ostrovsky. Position based cryptography. In Shai Halevi, editor, *CRYPTO*, volume 5677 of *Lecture Notes in Computer Science*, pages 391–407. Springer, 2009.

[CGO12] N. Chandran, J. A. Garay, and R. Ostrovsky. Edge fault tolerance on sparse networks. In Artur Czumaj, Kurt Mehlhorn, Andrew M. Pitts, and Roger Wattenhofer, editors, *ICALP (2)*, volume 7392 of *Lecture Notes in Computer Science*, pages 452–463. Springer, 2012.

[Cha81] D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24(2):84–88, 1981.

[CKL03] Ran Canetti, Eyal Kushilevitz, and Yehuda Lindell. On the limitations of universally composable two-party computation without set-up assumptions. In *EUROCRYPT*, pages 68–86, 2003.

[CKPR01] Ran Canetti, Joe Kilian, Erez Petrank, and Alon Rosen. Black-box concurrent zero-knowledge requires $\widetilde{\Omega}(\log n)$ rounds. In *STOC*, pages 570–579, 2001.

[CLOS02] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing*, STOC '02, pages 494–503, 2002.

[CLP10] Ran Canetti, Huijia Lin, and Rafael Pass. Adaptive hardness and composable security in the plain model from standard assumptions. In *Proceedings of the 51th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '10, pages 541–550, 2010.

[CO99] Giovanni Di Crescenzo and Rafail Ostrovsky. On concurrent zero-knowledge with pre-processing. In *CRYPTO*, pages 485–502, 1999.

[DBL90] *Proceedings of the Twenty Second Annual ACM Symposium on Theory of Computing, 14-16 May 1990, Baltimore, Maryland, USA*. ACM, 1990.

[DDN91] Danny Dolev, Cynthia Dwork, and Moni Naor. Non-Malleable Cryptography (Extended Abstract). In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing*, STOC '91, pages 542–552, 1991.

[DIK10] I. Damgård, Y. Ishai, and M. Krøigaard. Perfectly secure multiparty computation and the computational overhead of cryptography. In Henri Gilbert, editor, *EUROCRYPT*, volume 6110 of *Lecture Notes in Computer Science*, pages 445–465. Springer, 2010.

[DN07] I. Damgård and J. B. Nielsen. Scalable and unconditionally secure multiparty computation. In Alfred Menezes, editor, *CRYPTO*, volume 4622 of *Lecture Notes in Computer Science*, pages 572–590. Springer, 2007.

[DNS98] Cynthia Dwork, Moni Naor, and Amit Sahai. Concurrent zero-knowledge. In *STOC*, pages 409–418, 1998.

[Dol82] D. Dolev. The byzantine generals strike again. *J. Algorithms*, 3(1):14–30, 1982.

[DPPU88] C. Dwork, D. Peleg, N. Pippenger, and E. Upfal. Fault tolerance in networks of bounded degree. *SIAM J. Comput.*, 17(5):975–988, 1988.

[Elg85] Taher Elgamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *IEEE Transactions on Information Theory 31(4)*, pages 469–472, 1985.

[FLS90] Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple non-interactive zero knowledge proofs based on a single random string (extended abstract). In *FOCS*, pages 308–317, 1990.

[FS90] Uriel Feige and Adi Shamir. Witness indistinguishable and witness hiding protocols. In *STOC* [DBL90], pages 416–426.

[FY92] Matthew Franklin and Moti Yung. Communication complexity of secure computation. In *STOC*, pages 699–710, 1992.

[GGJS12] Sanjam Garg, Vipul Goyal, Abhishek Jain, and Amit Sahai. Concurrently secure computation in constant rounds. In *EUROCRYPT*, 2012.

[GJO+13a] Vipul Goyal, Abhishek Jain, Rafail Ostrovsky, Silas Richelson, and Ivan Visconti. Concurrent zero-knowledge in the bounded player model. Asiacrypt(1), pages 20–41, 2013.

[GJO+13b] Vipul Goyal, Abhishek Jain, Rafail Ostrovsky, Silas Richelson, and Ivan Visconti. Constant round concurrent zero-knowledge in the bounded player model. TCC, pages 60–79, 2013.

[GK96] Oded Goldreich and Ariel Kahan. How to construct constant-round zero-knowledge proof systems for np. *J. Cryptology*, 9(3):167–190, 1996.

[GLOV12] Vipul Goyal, Chen-Kuei Lee, Rafail Ostrovsky, and Ivan Visconti. Constructing non-malleable commitments: A black-box approach. In *FOCS*, pages 51–60. IEEE Computer Society, 2012.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. In *SIAM Journal of Computing 18:(1)*, ACM, pages 186–208, 1989.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game. In *STOC '87: Proceedings of the 19th annual ACM conference on Theory of computing*, pages 218–229, New York, NY, USA, 1987. ACM Press.

[GMW91] Oded Goldreich, Silvio Micali, and Avi Widgerson. Proofs that yield nothing but their validity. In *SIAM Journal of Computing 38:(3)*, ACM, pages 691–729, 1991.

[GO08] J. Garay and R. Ostrovsky. Almost-everywhere secure computation. In Nigel P. Smart, editor, *EUROCRYPT*, volume 4965 of *Lecture Notes in Computer Science*, pages 307–323. Springer, 2008.

[Gol02] Oded Goldreich. Concurrent zero-knowledge with timing, revisited. In *STOC*, pages 332–340, 2002.

[Goy11] Vipul Goyal. Constant Round Non-malleable Protocols Using One-way Functions. In *Proceedings of the 43rd Annual ACM Symposium on Theory of Computing*, STOC '11, pages 695–704. ACM, 2011.

[HILL99] Johan Haståd, Russell Impagliazzo, Leonid Levin, and Michael Luby. A pseudorandom generator from any one-way function. In *SIAM Journal of Computing 28:(4)*, ACM, pages 1364–1396, 1999.

[HJ07] M. Hinkelmann and A. Jakoby. Communications in unknown networks: Preserving the secret of topology. *Theor. Comput. Sci.*, 384(2-3):184–200, 2007.

[KL11] Dafna Kidron and Yehuda Lindell. Impossibility results for universal composability in public-key models and with fixed inputs. *J. Cryptology*, 24(3):517–544, 2011.

[KLP05] Yael Tauman Kalai, Yehuda Lindell, and Manoj Prabhakaran. Concurrent general composition of secure protocols in the timing model. In *STOC*, pages 644–653, 2005.

[KMO89] Joe Kilian, Silvio Micali, and Rafail Ostrovsky. Minimum resource zero-knowledge proofs (extended abstract). In *FOCS*, pages 474–479, 1989.

[KOS03] Jonathan Katz, Rafail Ostrovsky, and Adam Smith. Round Efficiency of Multi-party Computation with a Dishonest Majority. In *Advances in Cryptology — EUROCRYPT '03*, volume 2656 of *Lecture Notes in Computer Science*, pages 578–595. Springer, 2003.

[KP01] Joe Kilian and Erez Petrank. Concurrent and resettable zero-knowledge in poly-loalgorithm rounds. In *STOC*, pages 560–569, 2001.

[KPR98] Joe Kilian, Erez Petrank, and Charles Rackoff. Lower bounds for zero knowledge on the internet. In *FOCS*, pages 484–492, 1998.

[Lin03a] Yehuda Lindell. Bounded-concurrent secure two-party computation without setup assumptions. In *STOC*, pages 683–692, 2003.

[Lin03b] Yehuda Lindell. General composition and universal composability in secure multi-party computation. In *FOCS*, pages 394–403, 2003.

[Lin04] Yehuda Lindell. Lower bounds for concurrent self composition. In *TCC*, volume 2951 of *Lecture Notes in Computer Science*, pages 203–222. Springer, 2004.

[LP11] Huijia Lin and Rafael Pass. Constant-round Non-malleable Commitments from Any One-way Function. In *Proceedings of the 43rd Annual ACM Symposium on Theory of Computing*, STOC '11, pages 705–714, 2011.

[LPV08] Huijia Lin, Rafael Pass, and Muthuramakrishnan Venkitasubramaniam. Concurrent non-malleable commitments from any one-way function. In *Theory of Cryptography, 5th Theory of Cryptography Conference, TCC 2008*, pages 571–588, 2008.

[LPV09] Huijia Lin, Rafael Pass, and Muthuramakrishnan Venkitasubramaniam. A unified framework for concurrent security: Universal composability from standalone non-malleability. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing*, STOC '09, pages 179–188, 2009.

[MP07] Silvio Micali and Rafael Pass. Precise zero knowledge, 2007.

[MR01] Silvio Micali and Leonid Reyzin. Soundness in the public-key model. In *CRYPTO*, pages 542–565, 2001.

[Nao91] Moni Naor. Bit Commitment Using Pseudorandomness. *J. Cryptology*, 4(2):151–158, 1991.

[NSS06] Moni Naor, Gil Segev, and Adam Smith. Tight bounds for unconditional authentication protocols in the manual channel and shared key models. In Cynthia Dwork, editor, *CRYPTO*, volume 4117 of *Lecture Notes in Computer Science*, pages 214–231. Springer, 2006.

[OW93] Rafail Ostrovsky and Avi Widgerson. One-way functions are essential for non-trivial zero-knowledge. In *ISTCS*, pages 3–17, 1993.

[Pas03] Rafael Pass. Simulation in quasi-polynomial time, and its application to protocol composition. In *EUROCRYPT*, pages 160–176, 2003.

[Pas04a] R. Pass. Bounded-concurrent secure multi-party computation with a dishonest majority. In László Babai, editor, *STOC*, pages 232–241. ACM, 2004.

[Pas04b] Rafael Pass. Bounded-Concurrent Secure Multi-Party Computation with a Dishonest Majority. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing*, STOC '04, pages 232–241, 2004.

[PR03] R. Pass and A. Rosen. Bounded-concurrent secure two-party computation in a constant number of rounds. In *FOCS*, pages 404–413. IEEE Computer Society, 2003.

[PR05a] Rafael Pass and Alon Rosen. Concurrent non-malleable commitments. In *FOCS*, pages 563–572, 2005.

[PR05b] Rafael Pass and Alon Rosen. New and improved constructions of non-malleable cryptographic protocols. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing*, STOC '05, pages 533–542, 2005.

[PRS02] Manoj Prabhakaran, Alon Rosen, and Amit Sahai. Concurrent Zero Knowledge with Logarithmic Round-Complexity. In *Proceedings of the 43th Annual IEEE Symposium on Foundations of ComputerScience*, FOCS '02, pages 366–375, 2002.

[PTV10] Rafael Pass, Wei-Lung Dustin Tseng, and Muthuramakrishnan Venkitasubramaniam. Eye for an eye: Efficient concurrent zero-knowledge in the timing model. In *TCC*, pages 518–534, 2010.

[PW10] Rafael Pass and Hoeteck Wee. Constant-Round Non-malleable Commitments from Sub-exponential One-Way Functions. In *Advances in Cryptology — EUROCRYPT '10*, pages 638–655, 2010.

[RK99] Ransom Richardson and Joe Kilian. On the concurrent composition of zero-knowledge proofs. In *EUROCRYPT*, pages 415–431, 1999.

[Rom90] John Rompel. One-way functions are necessary and sufficient for secure signatures. In *STOC* [DBL90], pages 387–394.

[Ros00] Alon Rosen. A note on the round-complexity of concurrent zero-knowledge. In *CRYPTO*, pages 451–468, 2000.

[RR99] M. K. Reiter and A. D. Rubin. Anonymous web transactions with crowds. *Commun. ACM*, 42(2):32–38, 1999.

[RS60] Irving Reed and Gustave Solomon. Polynomial codes over certain fields. In *J. Soc. Ind. Appl. Math*, pages 300–304, 1960.

[RSG98] M. G. Reed, P. F. Syverson, and D. M. Goldschlag. Anonymous connections and onion routing. *IEEE Journal on Selected Areas in Communications*, 16(4):482–494, 1998.

[SCO+01] Alfredo De Santis, Giovanni Di Crescenzo, Rafail Ostrovsky, Giuseppe Persiano, and Amit Sahai. Robust non-interactive zero knowledge. In *CRYPTO*, pages 566–598, 2001.

[Sha79] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

[SP92] Alfredo De Santis and Giuseppe Persiano. Zero-knowledge proofs of knowledge without interaction. In *FOCS*, pages 427–436. IEEE Computer Society, 1992.

[SV12] Alessandra Scafuro and Ivan Visconti. On round-optimal zero knowledge in the bare public-key model. In *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques*, volume 7237 of *Lecture Notes in Computer Science*, pages 153–171. Springer, 2012.

[Wee10] Hoeteck Wee. Black-Box, Round-Efficient Secure Computation via Non-malleability Amplification. In *Proceedings of the 51th Annual IEEE Symposium on Foundations of Computer Science*, pages 531–540, 2010.

[Yao82a] Andrew Yao. Theory and applications of trapdoor functions. In *Proceedings of the 23rd Annual ACM Symposium on Foundations of Computer Science*, FOCS '82, pages 80–91, 1982.

[Yao82b] Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *FOCS*, pages 160–164. IEEE, 1982.