THE STRUCTURE OF SECURE MULTI-PARTY COMPUTATION

BY

MICHAEL J. ROSULEK

B.S., Iowa State University, 2003

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2009

Urbana, Illinois

Doctoral Committee:

Assistant Professor Manoj Prabhakaran, Chair
Professor Michael C. Loui, Co-chair
Assistant Professor Nikita Borisov
Professor Carl Gunter
Professor Ran Canetti, Tel Aviv University

Copyright

Abstract

Secure multi-party computation is a conceptual framework in which distrusting parties engage in a protocol to securely perform a computational task. Depending on the precise model of security, different sets of tasks admit secure protocols. We take a complexity-theoretic approach to studying the inherent difficulty of securely realizing tasks in various standard security models.

- We give the first alternate characterization of secure realizability in the framework of universally composable (UC) security. This is the first characterization in any model to consider completely arbitrary computational tasks and completely arbitrary communication channels.
- The most long-standing class of computational tasks studied are those in which two parties evaluate a deterministic function. For these tasks, we give the first complete, combinatorial characterizations of secure realizability in the passive, standalone, and universally composable security models, against computationally unbounded adversaries.
- Say that a task $\mathcal{G}$ has "as much cryptographic complexity" as another task $\mathcal{F}$ if there is a secure protocol for $\mathcal{F}$ that uses access to a secure implementation of $\mathcal{G}$. We show that there is an infinite hierarchy of tasks with *strictly increasing* cryptographic complexities, with respect to computationally unbounded security. We also show that there exist tasks whose cryptographic complexities are incomparable.
- In contrast, we show that under a standard cryptographic assumption, there exist only *two* distinct levels of cryptographic complexity with respect to polynomial-time security. Every task either has a trivial protocol using plain communication channels, or is complete (i.e., given access to a secure implementation of this task, there are secure protocols for all other tasks). This is the first result to derive a characterization of completeness for a class of arbitrary *interactive* tasks.

In light of these characterizations, the only tasks which are securely realizable in the demanding framework of universal composition are those related to secure communication. Indeed, the framework has been used to define the security of encryption schemes, which has allowed for modular design and analysis of protocols. We consider a similar approach for *homomorphic* encryption schemes. A homomorphic scheme is one in which anyone can obtain an encryption of $f(m_1, \ldots, m_n)$, given only the encryptions of unknown messages $m_1, \ldots, m_n$, for a specific set of functions f.

- We give a construction of a homomorphic encryption scheme in which the allowed homomorphic operation is as full-featured as possible — namely, one can derive a *correctly-distributed* encryption of $f(m)$ given an encryption of unknown message m, for some functions f — yet it is computationally infeasible to generate a ciphertext that is related to other ciphertexts in any other way. Our contributions involve developing new appropriate security definitions as well as new constructions.
- We show that schemes with such powerful security guarantees can be used to build conceptually simple, efficient, UC-secure protocols for verifiable computations on encrypted data. We show protocols for two tasks related to aggregating encrypted data.

Acknowledgements

I was somehow lucky enough to find an ideal pair of co-advisors, Manoj Prabhakaran and Michael Loui. It was my privilege to be Manoj Prabhakaran's first Ph.D. advisee, though I never once felt like his guinea pig. He was never protective of good research ideas, sharing them freely, and patiently tolerating the intellectual "drag factor" of a student just getting started in cryptography. His work ethic is superhuman, but he always displayed the patience of a saint when my work ethic was not. I can only hope that some hint of these traits wore off on me.

I certainly would not have stayed in grad school if I had not discovered Michael Loui. He has an astonishing dedication to all aspects of academia, and was the ideal model of a consummate professional. I especially thank him for encouraging me to become serious about teaching.

I am grateful to the members of my doctoral committee — Nikita Borisov, Ran Canetti, and Carl Gunter — for providing helpful feedback on this dissertation, and to my collaborators and co-authors — Anna Lisa Ferrara, Yuval Ishai, Hemanta Maji, Lars Olson, Amit Sahai, and Marriane Winslett. I am also in debt to all the dedicated teachers who have encouraged and motivated me throughout my 23 years in the public education system. There are too many to list, but I must single out Jack Lutz, whose courses at Iowa State University are what inspired me to pursue a career in theoretical computer science. I would like to thank my fellow CS theory grad students and officemates for providing frequent support, even more frequent distractions, and an outlet for venting frustration when I needed it.

Finally, but most importantly, the unconditional support and encouragement of my family has meant the world to me. Thanks, Laura, Mom, Dad, Dave, Audrey, and the Feltons, for always being my biggest cheerleaders.

To Laura.

Table of Contents

CHAPTER 1

Introduction

How can several mutually distrusting parties perform a computational task together on the Internet, without compromising the security of their information? This very general question is the focus of *secure multi-party computation* (MPC), introduced by Yao [96]. For example, the computational task in question may be something as simple as implementing a private communication channel in the presence of an eavesdropper, or evaluating a function on the parties' combined datasets. Or the task might be randomized and *interactive*, like an Internet poker game, to be played without a trusted dealer.

Security guarantees for MPC protocols are formulated using the *real/ideal* paradigm, introduced by Goldreich, Micali, and Wigderson [41]. In this paradigm, we consider a *real world* in which parties engage in a protocol (in the presence of an adversary) as well as an *ideal world* in which parties simply send their inputs to a trusted party who performs the task on their behalf (also in the presence of an adversary). For example, in the case of secure communication, the trusted party simply relays messages to the desired recipient; in the case of evaluating a function, the trusted party receives inputs, performs the desired computation, and gives outputs to each of the parties; in the case of a poker game, the trusted party is the dealer who shuffles and deals cards, elicits and verifies actions from the parties, and determines a winner. We say that the protocol is a *secure realization* of the task if the real world is "as secure as" the ideal world; or, more formally, if for every adversary attacking the real world interaction, there is a corresponding adversary attacking the ideal world interaction that achieves the same effect. How the details of this definition are specified determines a concrete model of MPC security.

Note that the behavior of the trusted party (formally defined as an interactive computer program) completely specifies an MPC task and all of its implicit security guarantees. For instance, implicit in the function evaluation example is that all parties' inputs are chosen independently of other parties' inputs, and that each party learns no more about another party's input than can be legitimately inferred from that party's output. Implicit in the poker example is the requirement that no party can influence how the cards are dealt, or have an unfair advantage in guessing another player's cards. Thus, multi-party computation provides a unified way of modelling many kinds of cryptographic security requirements.

1.1 Cryptographic Complexity

In this dissertation, we classify MPC tasks using an approach inspired by modern computational complexity theory. Instead of studying the inherent complexity of solving decision problems, we study the complexity of securely realizing MPC tasks. Like computational complexity, our *cryptographic complexity* approach classifies tasks according to two main themes: alternately characterizing complexity classes, and comparing tasks using reductions.

1.1.1 Understanding a Model via Complexity Classes

Whether a cryptographic task has a secure protocol depends on the precise security model being considered. For instance, every MPC security model must address the following important questions:

Tasks: Which possible tasks are considered in the model? Can tasks be *randomized* or must they be *deterministic*; general *multi-party* or only *two-party*; *interactive* or only *non-reactive* (i.e., simply evaluating a single function)?

Resource bound: Are adversaries and protocols allowed to have unbounded computational power, or must they be computationally bounded? In the latter case, probabilistic polynomial time is usually considered.

Adversarial capabilities: What are adversaries allowed to do? Can they deviate from the protocol (*active* corruption) or are they bound to follow it (*passive* corruption)? Can they choose which parties to corrupt "on the fly" (*adaptive* corruption), or must they choose once and for all at the beginning of the interaction (*static* corruption)?

Context: In what context is the protocol's security analyzed? Can the protocol be executed in the presence of arbitrary other protocols, on possibly correlated inputs (*universally composable (UC)* setting); can it be executed only in the presence of other copies of the same protocol (*concurrent self-composition*); or can the protocol be executed only in isolation of other concurrent instances (*standalone* setting)?

Each concrete security model naturally defines a "complexity class" consisting of the tasks which have secure protocols in that model (the *securely realizable* tasks), analogous to how computational complexity classes consist of the decision problems computable within a particular machine model. The expressivity and limitations of a security model are better understood by finding alternate characterizations of its associated complexity class, and by comparing its complexity class with those of other security models.

Related work. Some of the first results related to the complexity of MPC were to show that in some security models, *all* tasks have secure protocols.[1] Yao [97] and Goldreich, Micali, and Wigderson [41] constructed protocols for every multi-party task, secure against passive, computationally bounded adversaries, under standard cryptographic assumptions. Goldreich, Micali, and Wigderson also gave protocols for every task secure against active adversaries in the standalone setting. Ben-Or, Goldwasser, and Wigderson [9] and Chaum, Crépeau, and Damgård [25] constructed protocols for every MPC task, secure against unbounded adversaries that are allowed to corrupt a strict minority of parties. Since secure realizability is a trivial property of tasks in these models, research in these models has focused more on efficiency than on feasibility.

In security models where not all tasks have secure protocols, there are far fewer complete characterizations known. Kushilevitz [67] and Beaver [4] independently gave a combinatorial characterization of the symmetric-output secure function evaluation (SFE) tasks that have *perfectly* secure protocols against computationally unbounded, passive adversaries. In the same setting, Chor and Kushilevitz [28] also showed an alternative characterization for the special case of boolean-output functions.

In the UC framework, Canetti [15] demonstrated in the seminal work on the framework that not all tasks are securely realizable. Canetti, Kushilevitz, and Lindell [22] showed several broad impossibility results for many classes of two-party SFE in the framework. For several of these special classes of SFE tasks, their results provide a complete characterization of realizability. Beyond these results, there were no complete characterizations known for realizability in the UC setting.

1.1.2 Comparing Tasks via Reductions

In addition to comparing whole classes of tasks, it is often instructive to compare specific individual tasks. As an example, one might wonder whether bit-commitment is a more sophisticated task than coin-tossing (these two tasks are defined formally in Section 2.4). We can compare the cryptographic complexity of individual tasks, just as in computational complexity, by demonstrating a *reduction* between the two. The most natural reduction in the context of cryptographic complexity is whether there is a secure protocol for one task using an ideal black-box (i.e., access to a trusted party) which securely implements another task.

Reductions also provide a succinct way to understand larger complexity classes, by identifying *complete* tasks. A task is complete if all other tasks reduce to it; thus, complete tasks embody the essence of a complexity class.

As before, the precise reduction depends crucially on the security model enforced on the secure protocol. A more restrictive security model yields a reduction that can make finer complexity distinctions among tasks. In this work, we consider two reductions, both defined using the strongest natural security model available: security in the universal composition framework. We say that $\mathcal{F}$ reduces to $\mathcal{G}$ (written $\mathcal{F} \sqsubseteq \mathcal{G}$) if there is a UC-secure protocol for $\mathcal{F}$ which uses access to an ideal $\mathcal{G}$ functionality (or in standard UC parlance, a secure protocol for $\mathcal{F}$ in the "$\mathcal{G}$-hybrid" setting). We obtain two natural reductions of different strengths by considering the UC security requirement against computationally unbounded and computationally bounded (probabilistic polynomial-time) adversaries. We refer to these two concrete reductions as $\sqsubseteq^u$ and $\sqsubseteq^p$, respectively. Being based on a composable security notion, these reductions are both symmetric and transitive, and are therefore useful tools for comparing complexity.

Related work. Strictly speaking, feasibility results in a security model can be viewed as a reduction to a base task like a simple communication channel, which is usually provided "for free" in the model. We call a task *trivial* if it reduces to the model's basic communication channel. In some security models (for example, standalone or passive security against polynomial-time adversaries), all tasks are trivial (under certain cryptographic hardness assumptions), and thus are all equivalent under the kind of reduction we consider.

Even in security models with more than one level of complexity, almost all previous work has focused on the extremes of complexity: namely, identifying trivial tasks and complete tasks. The related work described in the previous section can be interpreted as classifying the trivial tasks in several security settings. Intermediate levels of complexity have rarely been considered, apart from demonstrating that such levels do exist. Furthermore, most of

[1] Technically, only *well-formed* tasks can have secure protocols, in any model. For example, the task of determining which parties are corrupted is not well-formed. To simplify the exposition in this introduction, we consider only well-formed tasks.

the results on completeness have not considered a strong reduction based on UC security. One notable exception is Canetti et al. [23], who showed that the bit commitment and coin-tossing tasks are complete under the $\sqsubseteq^p$ reduction. Kidron and Lindell [60] also showed that the impossibility results of Canetti, Kushilevitz, and Lindell [22] for secure realizability in the UC model can be extended to show impossibility of a $\sqsubseteq$-reduction to several natural key-registration tasks.

One of the first tasks observed to be complete against active adversaries (though not in the UC framework) was *oblivious transfer* (Section 2.4), as shown by Kilian [61]. Later, Ishai, Prabhakaran, and Sahai [56] showed that oblivious transfer is also complete under the more demanding $\sqsubseteq^u$ reduction. Many other variants of oblivious transfer were studied and found to be equivalent to the original [32, 14].

Kilian [62, 63] also gave a combinatorial characterization of the two-party SFE tasks that are complete against computationally unbounded, active adversaries. His result for deterministic functions was later extended to the $\sqsubseteq^u$ reduction by Kraschewski and Müller-Quade [65].

In some security settings, there are only two levels of complexity: the trivial tasks and the complete tasks. Kilian et al. [64] (building on [62, 68]) showed this to be the case for secure evaluation of *boolean-output, symmetric* functions, with respect to unbounded, passive adversaries. They also show an example of a non-boolean function which is neither complete nor trivial. In the case of two-party SFE in which only one party receives output, Kilian [62], and also Beimel, Malkin, and Micali [5], showed that every task is either trivial or complete with respect to unbounded active adversaries. However, the completeness definition in [5] is unique in not being based on black-box reductions. Harnik et al. [50] gave a characterization for completeness for the same class of functions, but against bounded, passive adversaries.

1.2 Our Results

In this dissertation, we present several new results furthering the understanding of cryptographic complexity for MPC tasks.

1.2.1 Structural Cryptographic Complexity

Universally composable (UC) security [16] is a strong notion in which a protocol can be safely used regardless of the context (i.e., other protocol instances) in which it executes. It was previously known [15, 17, 22] that this very strong security requirement precluded secure protocols for most tasks. However, no complete characterization of UC security was known for any class of tasks apart from several subclasses of SFE tasks.

In Chapter 3, we give the first alternative characterization of UC security. Intuitively, a task is securely realizable in the UC setting if and only if it is possible to "synchronize" two independent instances of the task. Most importantly, the characterization does not rely on any internal properties or representation of the task; instead, the characterization treats an MPC task as a black box, and secure realizability is determined based on the task's input/output behavior. As such, this characterization is the first in any non-trivial security model (let alone the demanding setting of UC security) to classify realizability of arbitrary *multi-party, interactive* tasks. We call this general characterization a "structural" result since it depends only on the structure of interactions among parties in the UC framework, and not on the details of the computational model. Indeed, this general characterization can be specialized to either the computationally unbounded or the probabilistic polynomial-time setting.

By applying this characterization, we can re-derive elementary proofs for all previously known impossibility results in the UC framework, as well as extend these impossibility results to more general communication channels. For the special case of 2-party tasks, we give the first characterization of UC-secure realizability for *non-reactive* tasks, as well as a completely *combinatorial* characterization for the special case of (deterministic) SFE tasks.

This result is joint work with Manoj Prabhakaran, and appeared in *CRYPTO* 2008 [84].

1.2.2 Unbounded Cryptographic Complexity

In Chapter 4, we give several new results classifying cryptographic complexity of *two-party symmetric-output secure function evaluation (SSFE)*, in several different security settings involving computationally unbounded adversaries. SSFE tasks are the most fundamental class of tasks, having been studied since the first MPC paper by Yao [96].

A SSFE task is completely specified by a function $\mathcal{F} : X \times Y \to Z$, where X, Y, and Z are finite sets. The task is for Alice, who holds input $x \in X$, and Bob, who holds input $y \in Y$, to both obtain the value $\mathcal{F}(x, y)$.

Unifying all of our results for SSFE tasks is a new technical tool for proving the impossibility of secure realization in a variety of settings. Namely, we show that $\mathcal{F}$ is securely realizable in any of the security settings we consider *if*

and only if a very simple "canonical" protocol for $\mathcal{F}$ is secure in that setting. Thus, we are able to give conceptually simple proofs of impossibility for SSFE tasks by simply showing an attack against a single canonical protocol that violates security in the desired setting. More specifically, we give new results for the following security settings:

- Against passive adversaries: Recall that a passive adversary is one that is not allowed to deviate from the prescribed protocol, but is "curious" and tries to infer as much as possible about the other party's input, based on what it sees during the interaction. Although this model has been studied for over 20 years, no complete characterization of SSFE tasks was known. However, Beaver [4] and Kushilevitz [67] independently gave a complete combinatorial characterization of the functions that have a secure protocol in a variant of this model, when the protocol is required to be *error-free*. We show that their characterization extends to the more natural case where the protocol is allowed to have negligible error.

  We also show an alternate characterization of this class in terms of the $\sqsubseteq^u$ reduction. Namely, an SSFE task $\mathcal{F}$ has a secure protocol against passive, unbounded adversaries if and only if $\mathcal{F} \sqsubseteq^u \mathcal{F}_{\text{COM}}$, where $\mathcal{F}_{\text{COM}}$ is the *bit-commitment* functionality, a natural cryptographic task.

- Against active adversaries in the standalone setting: In this model, adversaries are allowed to arbitrarily deviate from the prescribed protocol, but the protocol is assumed to execute in isolation of all other protocol instances. We give the first complete combinatorial characterization of the SSFE tasks with secure protocols in this model.

- Against active adversaries under concurrent self-composition: In this model, we analyze the protocol in a setting where each party may be participating in several concurrent instances of the *same* protocol. Lindell [70] showed that for a large class of tasks (including all SSFE tasks), security under self-composition implies UC security (security in the presence of *arbitrary* other protocols). We give a simpler and tighter proof of Lindell's result for the SSFE case. In particular, we show an optimal result: that security against *two* concurrent protocol instances is equivalent to UC security.

Finally, we also use our main technical tool to show that the $\sqsubseteq^u$ reduction (defined in terms of UC security against unbounded adversaries) can make very fine distinctions among cryptographic complexities. We prove a completely combinatorial sufficient condition for SSFE tasks $\mathcal{F}$ and $\mathcal{G}$ which implies that $\mathcal{F} \not\sqsubseteq^u \mathcal{G}$, and use it to show the following results:

- There exists an infinite hierarchy of strictly increasing cryptographic complexities. That is, there is an infinite sequence of tasks $\mathcal{G}_1, \mathcal{G}_2, \ldots$ such that $\mathcal{G}_i \sqsubseteq^u \mathcal{G}_j$ if and only if $i \leq j$.

- There exist tasks with incomparable complexities — that is, tasks $\mathcal{F}$ and $\mathcal{G}$ such that $\mathcal{F} \not\sqsubseteq^u \mathcal{G}$ and $\mathcal{G} \not\sqsubseteq^u \mathcal{F}$.

All previous results relevant to cryptographic complexity (in any security setting) focused on the extremes of complexity: the securely realizable tasks and the complete tasks. In some settings, intermediate complexities were identified but not studied in detail. Our new results are the first to make fine distinctions among these intermediate levels of complexity. Indeed, these distinctions are visible only using the extremely strong $\sqsubseteq^u$ reduction, which was never previously studied in great detail. As we show in Chapter 5, even the slightly weaker $\sqsubseteq^p$ reduction cannot make any of these fine complexity distinctions.

Interestingly, one theme underlying our results is to incorporate ideas from the *passive* security setting, and apply them to much stricter settings. We show in our main new technical tool that "canonical" protocols, which are defined very naturally in terms of passive security, also play an important role in active corruption settings. One of our two results classifying passive security shows that passive security has a natural characterization in the seemingly unrelated language of UC security. Finally, we show that if $\mathcal{G}$ has a lower round complexity than $\mathcal{F}$ against *passive* adversaries, then $\mathcal{F} \not\sqsubseteq^u \mathcal{G}$; again, a consequence in a much more demanding security setting.

These results represent joint work with Hemanta Maji and Manoj Prabhakaran, and appeared in *Theory of Computation Conference* 2009 [73]. The combinatorial characterizations for passive security and standalone security were also independently discovered by Künzler, Müller-Quade, and Raub [66], who also extended the characterization to non-symmetric SFE tasks.

1.2.3 Polynomial-Time Cryptographic Complexity

In Chapter 5, we study cryptographic complexity in the setting of computationally bounded (probabilistic polynomial-time) adversaries. In particular, we explore the complexity of MPC tasks under the $\sqsubseteq^p$ reduction. As alluded to above, the $\sqsubseteq^p$ reduction cannot resolve as many fine distinctions between tasks as the stronger $\sqsubseteq^u$ reduction.

Any results demonstrating the different resolution strengths of the $\sqsubseteq^p$ and $\sqsubseteq^u$ reductions must necessarily depend on a cryptographic hardness assumption. We show that under a reasonable and natural hardness assumption, the

two reductions are astonishingly different in their power: *all* of the diverse cryptographic complexities evident under the $\sqsubseteq^u$ reduction collapse under the $\sqsubseteq^p$ reduction. More formally, there are only two distinct levels of cryptographic complexity: the trivial tasks (those that have a UC-secure protocol using simple communication channels) and those which are complete under the $\sqsubseteq^p$ reduction. We call this result the *zero-one law*.

We prove the zero-one law for deterministic, finite, two-party tasks. Importantly, our results are the first to consider completeness for *arbitrary, interactive* tasks, in any security setting. We model such arbitrary tasks as finite automata, and one important technical contribution is to develop new automata-theoretic tools. In particular, we identify cryptographically non-trivial behaviors of an arbitrary task, which leads to the first combinatorial characterization of *interactive* tasks in the UC framework. However, to show that a task is complete, we also construct new protocols which *exploit* the non-trivial behaviors of arbitrary tasks.

Interestingly, the hardness assumption that we consider is not only sufficient but also necessary for the zero-one law. That is, if the zero-one law holds, then the hardness assumption is also true.

This result is joint work with Hemanta Maji and Manoj Prabhakaran, currently in preparation [74].

1.2.4 Reconciling Non-Malleability & Homomorphic Encryption

Although most tasks do not have UC-secure protocols, tasks related to secure communication have proven a useful way to define security for encryption schemes and signature schemes [15, 81, 19]. These definitions in the UC framework are important for motivating the use of standard encryption schemes and signature schemes as modular components in a larger protocol. In Chapter 6, we explore this avenue for the case of homomorphic encryption schemes.

In a homomorphic encryption scheme, given encryptions of unknown messages $m_1, \ldots, m_n$, *anyone* can obtain a "fresh" encryption of $f(m_1, \ldots, m_n)$ for some functions f, as a feature of the scheme. Perhaps surprisingly, homomorphic encryption schemes are not frequently used as modular components in larger protocols, even though they appear well-suited to the task of manipulating encrypted data. This discrepancy stems from the fact that all existing security definitions for homomorphic encryption — and indeed, schemes themselves — address only the question of which homomorphic operations are possible as *features*; they do not consider preventing additional homomorphic operations that might be possible as *vulnerabilities*.

Intuitively, an ideal homomorphic encryption scheme should provide both guarantees: first, that certain homomorphic operations f are available as features, and second, a guarantee of *non-malleability*, that it should be infeasible for an adversary to generate ciphertexts that are related to other ciphertexts in any way other than the provided features. Previous security definitions for encryption could not capture such sharp requirements — they either disallowed all homomorphic features, or left open the possibility for arbitrary homomorphic vulnerabilities.

In this work, we develop new ways to define and achieve such robust security for homomorphic encryption.

- We give several new security definitions, of both the standard game-based variety and in terms of MPC tasks defined in the UC framework. We show the equivalence between these two kinds of definitions, and also show how our new definitions unify all previous definitions of non-malleability.

- We construct the first public-key encryption schemes satisfying our strong definitions, and show how such schemes can be used to construct simple, yet UC-secure protocols that involve computation on encrypted data. Previous comparable protocols that used homomorphic encryption were forced to employ costly zero-knowledge proofs to prevent adversaries from manipulating ciphertexts in undesired ways. However, our new security definitions on the encryption scheme already limit an adversary's capabilities in manipulating ciphertexts; thus our simple protocols can provide security against active adversaries without costly zero-knowledge proofs.

- We show that a large desirable class of homomorphic operations are impossible to achieve in any encryption scheme, under our strong security definitions.

These results represent joint work with Manoj Prabhakaran which appeared in *CRYPTO* 2007 [83], *ICALP* 2008 [85], and *ASIACRYPT* 2008 [86]; and also unpublished joint work with Anna-Lisa Ferrara and Manoj Prabhakaran.

CHAPTER 2

Preliminaries

In this chapter, we present some preliminary notions and definitions. Even readers who are already familiar with the UC framework may wish to quickly skim our summary of the framework given below — some of our conventions differ slightly from the original work of Canetti [16]. For convenience, we have provided in Section 2.2.2 a brief summary of these technical differences. We recommend reading the subsequent sections, however, which describe important terminology and notation used throughout this work.

2.1 Basic Notation

We say that a function $\nu : \mathbb{N} \to [0,1]$ is *negligible* if $\nu(k) = k^{-\omega(1)}$; that is, if ν approaches zero asymptotically faster than any inverse polynomial. We write $\nu \approx \mu$ to mean that $|\nu(k) - \mu(k)|$ is a negligible function in k. We say that a probability $p(k)$ is *overwhelming* (in k) if $1 - p(k)$ is a negligible function in k (i.e., $p \approx 1$).

When X is a finite set, we write $x \leftarrow X$ to mean that x is chosen uniformly at random from X.

When $\mathcal{D}_1$ and $\mathcal{D}_2$ are two discrete probability distributions with supports in the set S, we write $\Delta(\mathcal{D}_1, \mathcal{D}_2)$ to denote the *statistical distance* between them, defined as:

$$\Delta(\mathcal{D}_1, \mathcal{D}_2) = \frac{1}{2} \sum_{x \in S} \Big| \mathcal{D}_1(x) - \mathcal{D}_2(x) \Big| = \max_{T \subseteq S} \sum_{x \in T} \Big(\mathcal{D}_1(x) - \mathcal{D}_2(x) \Big).$$

We say that two probability ensembles $\mathcal{D} = \{\mathcal{D}_k \mid k \in \mathbb{N}\}$ and $\mathcal{D}' = \{\mathcal{D}'_k \mid k \in \mathbb{N}\}$ are *statistically indistinguishable* if $\Delta(\mathcal{D}_k, \mathcal{D}'_k)$ is negligible in k. We say that the two ensembles are *computationally indistinguishable* if for all polynomial-time algorithms M,

$$\Big| \Pr_{x \leftarrow \mathcal{D}_k}[M(x) = 1] - \Pr_{x \leftarrow \mathcal{D}'_k}[M(x) = 1] \Big| \text{ is negligible in } k.$$

2.2 Universal Composition Framework

The Universal Composition (UC) framework was first introduced by Canetti [16] as a realistic model for analyzing protocols in complex network environments like the Internet. The full details of the framework are well beyond the scope of this work. We provide a self-contained overview of the relevant technical details of the model, and refer the reader to [16] for the a complete treatment.

Entities and their interaction. The *network* is a collection of interactive Turing Machines[1] (ITMs), who communicate with each other by writing on shared communication tapes. For simplicity, we assume that communication tapes are write-once, and that all messages written to communication tapes have unambiguous delimiters, and that ITMs do not take action on a message written to a communication tape until the message is complete, and that only one machine can write to the tape at a time.[2] We say that a communication tape is *linked* to two machines in an execution if both have access to it; i.e., if the two machines can use it to communicate.

We identify 4 different types of entities:

Functionalities model trusted parties in the network. The code of a functionality completely defines a MPC task (e.g., the functionality may be the dealer in a poker game, or a party that performs a particular fixed computation on given data).

An n-party functionality has a communication tape for each of the n parties, and a communication tape for the adversary. The functionality also has an input tape which contains a list of which of the parties (if any)

[1] The choice of the interactive Turing Machine model is not crucial. One may define an equivalent model using, say, I/O automata, as in [87].

[2] One may wish to model each communication tape as a *pair* of tapes, one for each direction of communication, with appropriate read/write privileges. For simplicity, we consider each communication line to be a single tape.

are corrupt. Sometimes it is convenient to consider functionalities which also receive a security parameter as input, though we most frequently consider functionalities which do not.

We call a functionality *non-reactive* if it simply waits for inputs from each of the parties, then gives output to one or more of the parties, then stops responding. Otherwise we say that the functionality is *reactive*.

Protocols prescribe a way of interacting with a functionality. A protocol ITM has a communication tape for the environment and another communication tape for a functionality. A protocol also has an input tape on which it will receive a security parameter and an index (i.e., the identity of the party).

An important protocol in the framework is the *dummy protocol*, denoted π_{dummy}. It prescribes the same behavior for each party, so we do not need to specify the number of parties. π_{dummy} simply keeps its two communication tapes synchronized; it relays every message appearing on one communication tape to the other, and vice-versa. In essence, π_{dummy} effects a channel for the environment and functionality to directly communicate.

Environments model arbitrary contexts in which a protocol might be asked to execute. An environment has a communication tape for each party, as well as one for the adversary. An environment also has an input tape and a (without loss of generality) single-bit output tape.

Adversaries model arbitrary deviation from protocols by corrupt parties. In this work, we exclusively consider a setting where corruptions are chosen *statically*. That is, an adversary is parameterized by the indices of parties that it wishes to corrupt. An adversary that corrupts m parties has $m + 1$ separate communication tapes for communicating with the functionality (i.e., on behalf of the m corrupt parties and on its own behalf), and a final communication tape for communicating with the environment. An adversary also has an input tape on which it receives a security parameter.

An important specific adversary is called the *dummy adversary*. Like the dummy protocol, a dummy adversary relays all of its communication with the functionality to/from the environment. It expects inputs from the environment to be labeled with the index of one of the adversary's many communication tapes with the functionality. Similarly, it labels its outputs to the environment with the index of the communication tape on which it received the message from the functionality. Alternatively, it is sometimes convenient to consider that the adversary has $m + 1$ communication tapes shared with the environment as well.

Execution. Let $\mathcal{Z}$ be an environment, $\mathcal{A}$ be an adversary, π be a protocol, and $\mathcal{F}$ be a functionality. We define $\text{EXEC}[\mathcal{Z}, \mathcal{A}, \pi, \mathcal{F}, k]$ to be the random variable of the environment $\mathcal{Z}$'s output when executing the collection of machines as described above, with security parameter k.

More formally, $\text{EXEC}[\mathcal{Z}, \mathcal{A}, \pi, \mathcal{F}, k]$ is defined as follows. Each ITM in the system is initialized with the security parameter on its input tape, where appropriate. The functionality $\mathcal{F}$ is initialized with a list of indices of the parties corrupted by $\mathcal{A}$. For each uncorrupted party, an instance of π is included in the system, initialized with the corresponding party's index. The shared communication tapes of the ITMs are linked as described above. Finally, the system is executed until the environment $\mathcal{Z}$ writes to its output tape. The environment's input tape is provided to allow non-uniform computation to be considered.

We do not formally define the order in which individual ITMs are activated during the execution; the details are not crucial to our results.

Resource bounds; admissible machines. We consider two different variants of the UC framework, to model computationally unbounded settings and polynomial-time settings. In the unbounded setting, the entities defined above are allowed to be arbitrary ITMs; we say that all ITMs are *admissible*.

In the probabilistic polynomial-time (PPT) setting, defining admissibility is a non-trivial task whose subtleties are beyond the scope of this work. We adopt the definition given by Hofheinz, Unruh, and Müller-Quade [53], which we summarize as follows:

- An environment is admissible if it runs in (*a priori*) probabilistic polynomial time[3] in the security parameter.
- A functionality $\mathcal{F}$, adversary $\mathcal{A}$, and protocol π are jointly admissible if, for all admissible environments $\mathcal{Z}$, the execution considered in $\text{EXEC}[\mathcal{Z}, \mathcal{A}, \pi, \mathcal{F}, k]$ halts in polynomial time (in the security parameter k) with overwhelming probability (in the security parameter).

Even though these admissibility requirements seem complicated, we mostly focus on subclasses of functionalities which perform a constant amount of work in each activation. Thus it is usually quite easy to see that a particular adversary or protocol is admissible for all such functionalities.

[3] That is, there is a fixed polynomial p such that the environment halts after $p(k)$ steps when the security parameter is k

Secure realization. Let π be a protocol and $\mathcal{F}$ and $\mathcal{G}$ be functionalities. We say that π is a *secure realization* of $\mathcal{F}$ in the $\mathcal{G}$-hybrid setting if for all admissible real-world adversaries $\mathcal{A}$, there exists another admissible adversary (called a *simulator*) $\mathcal{S}$ such that for all admissible environments $\mathcal{Z}$, we have

$$\Pr\left[\text{EXEC}[\mathcal{Z}, \mathcal{A}, \pi, \mathcal{G}, k] = 1\right] \approx \Pr\left[\text{EXEC}[\mathcal{Z}, \mathcal{S}, \pi_{\mathsf{dummy}}, \mathcal{F}, k] = 1\right] \tag{2.1}$$

We refer to the execution involving π and $\mathcal{G}$ as the *real world* (where parties run the protocol), and the execution involving π_{dummy} as $\mathcal{F}$ as the *ideal world*.

Intuitively, the environment's output defines a condition for an adversarial attack. The definition implies that no adversary can succeed in a given attack against the protocol π any more than is possible in the ideal world. Thus, the real world is "as secure as" the ideal world. Since the semantics of the ideal world are simpler and easy to interpret from the specification of $\mathcal{F}$, security guarantees of this form are satisfactory.

For simplicity, we often let the security parameter be implicit, and exclude it from our notation. Then we simply write:

$$\text{EXEC}[\mathcal{Z}, \mathcal{A}, \pi, \mathcal{G}] \approx \text{EXEC}[\mathcal{Z}, \mathcal{S}, \pi_{\mathsf{dummy}}, \mathcal{F}]$$

as a shorthand for Equation 2.1.

We note that since we consider only static security (the adversary cannot adaptively decide which parties to corrupt), the security requirement is trivially true for adversaries which corrupt all parties. The simulator can simply internally run the real-world adversary and ignore the functionality.

Dummy adversary. We note that, without loss of generality, the definition of secure realization can be restricted to consider only real-world adversaries $\mathcal{A}$ which are dummy adversaries. If $\mathcal{Z}$ is some environment and $\mathcal{A}$ is not a dummy adversary, then the effect of executing a protocol in the presence of $\mathcal{A}$ and $\mathcal{Z}$ can also be achieved by a dummy adversary and an environment that simulates both $\mathcal{A}$ and $\mathcal{Z}$ together.

Universal composition. We have the following fundamental result in the UC framework, due to Canetti:

Theorem 2.1 (Universal Composition [16])**.** *If π is a secure realization of $\mathcal{F}$ in the $\mathcal{G}$-hybrid world, and ρ is a secure realization of $\mathcal{G}$ in the $\mathcal{H}$-hybrid world, then π^ρ is a secure realization of $\mathcal{F}$ in the $\mathcal{H}$-hybrid world, where π^ρ is the protocol π in which each external interface to an instance of $\mathcal{G}$ is replaced with an instance of ρ.*

In other words, no matter how π might interact with $\mathcal{G}$, it is always safe to replace $\mathcal{G}$ with its secure implementation ρ.

2.2.1 Other Security Models as Special Cases

By appropriately restricting the UC security definition, we can achieve other important security models as special cases:

Standalone security. Call an environment a *standalone* environment if it does not interact with the adversary during the execution of the protocol. If in the definition of UC security, we quantify only over standalone environments, we achieve the standard notion of *standalone security*. In practical terms, standalone security means that the simulator is allowed to *rewind* the adversary, not just execute the adversary in a straight line.

Security under concurrent self-composition. In the concurrent self-composition setting [70], we consider the security of protocols in the presence of other concurrent instances of *the same protocol only.* A simple way to define this is to slightly modify the model to allow parties to invoke several instances of the protocol, and then consider only standalone environments (so that the adversary does not communicate with the environment while any of the protocol instances are executing). We further restrict the multiple protocol instances so that each party plays the same *role* in all instances (i.e., each time a party invokes a protocol instance, it runs the protocol's ITM with the same party index/identity as input).

Although this natural definition of concurrent self-composition requires a slight modification to the model and security definition (not just a restriction in its quantifiers), we can achieve the same effect by simply restricting the standard UC security definition to the class of environments which internally simulate a standalone environment, a concurrent-self-composition adversary, and all but one of the protocol instances in the self-composition interaction.

Passive security. Call an adversary *passive* with respect to a protocol π if during the interaction, it runs the π protocol honestly (in the role of the corrupt party) on inputs provided by the environment, and reports back the protocol outputs to the environment. Note that we allow the adversary to interact arbitrarily with the environment throughout the execution, although the environment cannot influence the adversary's honest execution of π.

If in the definition of UC security, we quantify over only passive adversaries and simulators, we achieve the standard definition of *passive security*. Note that in the ideal world, the simulator must be passive *with respect to the dummy protocol*. It is easy to see that without loss of generality, a real-world passive adversary simply reports its entire internal state (i.e., the adversary's *view*: random tape and transcript of the π-interaction) to the environment. Then the corresponding simulator must run the dummy protocol and output a simulated view of the real-world protocol π.

2.2.2 Differences from Canetti's Model

The model presented above is conceptually the same as the one originally presented by Canetti in [16]. However, for technical reasons, we have deviated slightly from that presentation in some technical respects:

- In addition to the standard definition, we also consider a variant of the UC framework in which all entities can be computationally unbounded.
- We use the definition of polynomial runtime from Hofheinz, Unruh, and Müller-Quade [53]. They point out some deficiencies with the definition of polynomial runtime from [16] and other works, and also carefully reprove a universal composition theorem for their new notion.
- We allow individual entities in the model (protocols, functionalities, environments, adversaries) to communicate via *completely ideal, synchronous* channels. That is, the adversary cannot directly control delivery of such communication between honest entities, nor is the adversary even notified of such communication. Instead, we suppose that any adversarial control of the network is specifically modeled within the functionality itself (i.e., it directly interacts with the adversary each time it intends to deliver an output or process an input); see the discussion in Section 2.2.4.
- For simplicity, we avoid the use of session- and process-IDs. We assume that the various instances of system entities can reliably communicate among each other. One exception is that it is often convenient to allow the environment to masquerade as another entity instance (provided that this masquerade does not conflict with the actual entities in the execution).

  We also do not consider the details of how individual ITMs are activated in the execution, the precise details of which are not crucial to our results.
- We exclusively consider protocols which use only one instance of a functionality. To simulate the effect of multiple instances, we consider a "wrapper" around a functionality that provides an interface to several independent sessions; see the following section.

2.2.3 Cryptographic Complexity Notation

In our model, separate instances of the protocol ITM cannot communicate directly with each other. Indeed, for maximum flexibility, we assume that all communication must be facilitated by the functionality. We have also made a restriction that there be only one functionality instance throughout an execution.

It is most natural to allow protocols to use many independent instances of a functionality, and to provide a dedicated communication channel for the protocol. We define the private channel functionality $\mathcal{F}_{\text{PVT}}$ to have the following behavior:

- On input (P, m) from party P', generate delayed output (P', m) for party P.[4]

Let $\mathcal{F}$ be a functionality. We define $\widetilde{\mathcal{F}}$ to be its "augmented" version, which simulates access to independent (asynchronous) instances of $\mathcal{F}$ as well as access to an instance of $\mathcal{F}_{\text{PVT}}$, as follows:

- Internally simulate independent instances of $\mathcal{F}$ and an instance of $\mathcal{F}_{\text{PVT}}$. Associate with each instance of $\mathcal{F}$ a unique session ID.

[4] Delayed outputs are defined later in Section 2.2.4.

- Upon receiving an input of the form $(\mathcal{F}, sid, cmd)$ from party P, begin internally simulating a fresh instance of $\mathcal{F}$ associated with session ID sid unless one already exists. Then hand input cmd to that instance of $\mathcal{F}$ on behalf of party P.
- On input $(\mathcal{F}_{\text{PVT}}, P, m)$ from P', give output $(\mathcal{F}_{\text{PVT}}, P', m)$ to party P.
- Whenever an instance sid of $\mathcal{F}$ produces an output m for party P, deliver output $(\mathcal{F}, sid, m)$ to party P.

Definition 2.2. *When $\mathcal{F}$ is securely realizable in the $\mathcal{G}$-hybrid setting, we write $\mathcal{F} \sqsubseteq_{strict} \mathcal{G}$. When $\mathcal{F}$ is securely realizable in the $\widetilde{\mathcal{G}}$-hybrid setting, we write $\mathcal{F} \sqsubseteq \mathcal{G}$.*

Thus $\mathcal{F} \sqsubseteq_{strict} \mathcal{G}$ means that $\mathcal{F}$ can be securely realized via a protocol that uses access to a single instance of $\mathcal{G}$, and without any additional communication channels. $\mathcal{F} \sqsubseteq \mathcal{G}$ means that $\mathcal{F}$ can be securely realized via a protocol that uses access to any number of instances of $\mathcal{G}$, and also access to $\mathcal{F}_{\text{PVT}}$. The $\sqsubseteq$ relation is transitive, by the universal composition theorem.

We superscript the $\sqsubseteq$ and $\sqsubseteq_{strict}$ relations with either u or p (i.e., $\sqsubseteq^u$, $\sqsubseteq^p$) when referring specifically to the unbounded or PPT settings, respectively. We use the bare notations $\sqsubseteq$ and $\sqsubseteq_{strict}$ in results that apply to both settings (including when constructing secure protocols).

2.2.4 Asynchronous Output Delivery & Non-Trivial Protocols

In our model of UC security, the communication between an honest party and the functionality is completely ideal: private (from the adversary) and instantaneous. In this way, the UC framework can be used to model important requirements like *fairness* (namely, that it is not possible for one party to learn its output without the other party also learning its own output).

However, it is most common to consider communication that is asynchronous, and under the control of the adversary. Following [16], we use a standard idiom for defining asynchronous delivery in the description of a functionality. When we say that a functionality generates a *delayed output m for party P*, we mean the following:

1. The functionality internally maintains a queue of outputs for every party P. The message m is entered into the queue at this time.
2. The functionality sends (OUTPUT, P) to the adversary at this time.
3. Later, whenever receiving an input of the form $(\text{DELIVER}, P')$ from the adversary, the functionality removes the first message in the queue for P', and if such a message exists, immediately writes it to the communication tape of P'.

We assume that the functionality's other communication with the adversary does not conflict with messages of the form (OUTPUT, P) and $(\text{DELIVER}, P)$. Delayed outputs model a network in which the adversary has total control over timings, but does not get to see the contents of the communications.

A functionality whose outputs are all delayed provides no guarantee that honest parties will ever receive any output. Consequently, the protocol which does nothing is a secure realization of *any* such functionality. To avoid this undesirable breakdown of the security definition, Canetti et al. [23] defined the following refinement of secure realization:

Definition 2.3 (Non-trivial protocols)**.** *Suppose π is a secure realization of $\mathcal{F}$ in the $\mathcal{G}$-hybrid world. We say that π is a* non-trivial *realization, and write $\mathcal{F} \sqsubseteq_{nt} \mathcal{G}$, if for every real-world adversary that corrupts no one and eventually delivers all delayed outputs of $\mathcal{G}$, the corresponding simulator also eventually delivers all delayed outputs in $\mathcal{F}$.*

In a non-trivial protocol, it is still possible that a party does not receive output. However, non-triviality implies that a party receives no output only when the ideal functionality does not specify an output, or when the adversary interferes (either by corrupting a party or by refusing to deliver some messages of the protocol). Note that a corresponding universal composition theorem immediately holds for non-trivial protocols. Again, we superscript the relation as $\sqsubseteq^u_{nt}$ and $\sqsubseteq^p_{nt}$ when referring specifically to the unbounded or PPT setting, respectively.

2.3 Two-Party Functionalities

We highlight two special classes of 2-party functionalities that are relevant for this work. In the two-party setting, we refer to the parties by their traditional names Alice and Bob.

2.3.1 Secure Function Evaluation

Most works studying MPC have been restricted to the special class of functionalities called *secure function evaluation* functionalities, also sometimes called *non-reactive* functionalities.

Definition 2.4 (Secure Function Evaluation)**.** *A 2-party functionality $\mathcal{F}$ is a* (deterministic) secure function evaluation (SFE) *functionality if it behaves as follows, for some pair of functions $f_A, f_B : X \times Y \to Z$, where X, Y, and Z are finite sets:*

1. *Wait for input $x \in X$ from Alice and input $y \in Y$ from Bob.*
2. *When both x and y have been received, send delayed output $f_A(x, y)$ to Alice and delayed output $f_B(x, y)$ to Bob.*

We often abuse notation slightly and identify $\mathcal{F}$ itself with the pair of functions (f_A, f_B). We note that $\mathcal{F}$ is deterministic and provides only one evaluation of its function. Furthermore, $\mathcal{F}$ provides no fairness guarantee – because of the delayed outputs, an adversary who corrupts a party may learn its own output and never deliver the other party's output, so our definition does not provide a guarantee of fairness.

We restrict our attention to *finite* functions. One might also consider functions whose complexity (including input and output domains) depends on the global security parameter. When this is the case, we state it explicitly.

When $f_A = f_B$, we say that the corresponding SFE functionality has *symmetric output* (SSFE). The case where one of $\{f_A, f_B\}$ is a constant function is (somewhat unfortunately) often called *asymmetric* SFE throughout the literature. We instead refer to this class of SFE functionalities as *one-sided output* SFE.

We can specify a symmetric-output SFE functionality by its function table. The table is a matrix with each row associated with an input for Alice and each column associated with an input for Bob. Each entry in the table specifies the function's output on the corresponding pair of inputs. The labels of the inputs themselves are not important, so the functionality is completely specified by its table. For example, the boolean XOR functionality can be written as $\left[\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right]$, and the boolean AND functionality can be written as $\left[\begin{smallmatrix} 0 & 0 \\ 0 & 1 \end{smallmatrix}\right]$.

Definition 2.5 (Redundant Inputs)**.** *Let $\mathcal{F} = (f_A, f_B)$ be a 2-party SFE functionality. We say that x is a* redundant input for Alice *if there exists $x' \neq x$ such that:*

$$f_A(x, y) \neq f_A(x, y') \Rightarrow f_A(x', y) \neq f_A(x', y') \quad \textit{and} \quad f_B(x, \cdot) \equiv f_B(x', \cdot).$$

That is, by changing her input from x to x', Alice learns no less about Bob's input, but Bob's output is unaffected. We define redundancy for Bob's inputs symmetrically.

It is easy to see that if x is a redundant input, then any ideal-world adversary can be made to never supply input x to $\mathcal{F}$, without loss of generality.

Definition 2.6 (Function Isomorphism)**.** *Let $\mathcal{F} = (f_A, f_B)$ and $\mathcal{G} = (g_A, g_B)$ be 2-party SFE functionalities. We say that $\mathcal{F}$ and $\mathcal{G}$ are* isomorphic *if $\mathcal{G}$ can be obtained from $\mathcal{F}$ via repeated applications of the following operations:*

- *Adding or removing a redundant input (for either party),*
- *Permuting a party's input domain,*
- *Consistently re-labeling the outputs of $f_A(x, \cdot)$ for any x, or of $f_B(\cdot, y)$ for any y,*
- *Reversing the roles of Alice and Bob.*

In the above definition, we say that f is a consistent re-labeling of g if $f(z) \neq f(z') \Leftrightarrow g(z) \neq g(z')$; that is, if the outputs have simply been renamed. For example, the symmetric-output XOR functionality $\left[\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right]$ is a consistent re-labeling of the functionality $\left[\begin{smallmatrix} 1 & 2 \\ 3 & 4 \end{smallmatrix}\right]$; thus the two are isomorphic.

It is easy to see that if $\mathcal{F}$ and $\mathcal{G}$ are isomorphic, then $\mathcal{F}$ and $\mathcal{G}$ are equivalent under $\sqsubseteq^p$ and $\sqsubseteq^u$ reductions. Thus without loss of generality, we usually consider only functions that are free of redundant inputs.

2.3.2 Deterministic Finite-Memory Functionalities

In addition to SFE functionalities, we also deal extensively with reactive functionalities. The following special class of reactive functionalities plays an important role in our results:

Definition 2.7. *A* deterministic finite functionality (DFF) *is a tuple $\mathcal{F} = (Q, X, Y, \delta, f_A, f_B, q_0)$, where*

- *Q is a finite set of states,*
- *X and Y are finite input sets,*
- *$\delta : Q \times X \times Y \to Q$ is the (partial) state transition function,*
- *$f_A, f_B : Q \times X \times Y \to \{0,1\}^*$ are two output functions, and*
- *$q_0 \in Q$ is the start state.*

The behavior of $\mathcal{F}$ as an ideal functionality is as follows:

1. *Set variable q to be the initial state q_0. Then repeatedly do:*
2. *Wait for input $x \in X$ from Alice and input $y \in Y$ from Bob. If $\delta(q, x, y)$ is undefined, then simply stop responding.*
3. *Send delayed output $f_A(q, x, y)$ to Alice and delayed output $f_B(q, x, y)$ to Bob.*
4. *Update variable $q \leftarrow \delta(q, x, y)$ and repeat from step (2).*

For simplicity, we often use the above standard variable names (Q, X, Y, δ, f_A, f_B, q_0) when the context of $\mathcal{F}$ is clear.

It is easy to see that SFE functionalities are special class of DFFs. We point out that in the most general terms, a functionality need not execute in a sequence of rounds that consist of inputs from both parties and outputs to both parties. The UC framework allows for more arbitrary functionalities that take inputs and give outputs without restriction. However, to the best of our knowledge, no finite-memory reactive functionalities considered in the literature require more generality than our DFF definition. Also, the constructions in [41, 23] of protocols for arbitrary reactive functionalities explicitly model functionalities as having "rounds" of interaction as in a DFF.

We also note that even though a DFF induces a kind of synchronous delivery of inputs from the parties and outputs to the parties, we still consider DFFs within the standard (asynchronous) communication model. That is, the inputs and outputs are delivered asynchronously between the functionality and parties; the functionality simply chooses to wait until receiving input from both parties, and to give outputs to both parties in the same activation.

2.4 Important Functionalities

For reference, we include here formal definitions of some commonly used ideal functionalities in the UC framework.

Private channels, $\mathcal{F}_{\text{PVT}}$. We have already mentioned the private channel functionality above, but repeat it here (in its delayed-output formulation), in Figure 2.1.

On input (P, m) from party P', give delayed output (P', m) to party P.

Figure 2.1: Private channel functionality $\mathcal{F}_{\text{PVT}}$.

Oblivious transfer, $\mathcal{F}_{\text{OT}}$. Oblivious transfer was first introduced Rabin [88] as a generalization of a noisy communication channel. It is known to be a complete functionality in many settings [61, 56], and there are many equivalent variants [14, 32]. Throughout this work we will use the standard 1-out-of-2 oblivious bit transfer, defined in Figure 2.2.

On input $(x_0, x_1) \in \{0,1\}^2$ from Alice, and input $b \in \{0,1\}$ from Bob, give delayed output x_b to Bob, and delayed output $\perp$ to Alice.

Figure 2.2: Oblivious transfer functionality $\mathcal{F}_{\text{OT}}$.

Commitment, $\mathcal{F}_{\text{COM}}$. The commitment functionality is the cryptographic equivalent of a locked box, and is defined in Figure 2.3. The sender gives an input to the functionality, which is stored and not revealed to the receiver until the sender instructs. Once the input is given to the functionality, the sender cannot change it.

We have defined $\mathcal{F}_{\text{COM}}$ to be parameterized by a domain of m-bit strings. However, it is easy to see that m copies of 1-bit $\mathcal{F}_{\text{COM}}$ can be used in parallel to securely realize a single m-bit $\mathcal{F}_{\text{COM}}$. Thus when showing a protocol for $\mathcal{F}_{\text{COM}}$, it suffices to show a protocol for the single-bit case.

Coin-tossing, $\mathcal{F}_{\text{COIN}}$. The coin-tossing functionality samples a fair coin and gives it to both parties (Figure 2.4).

The functionality is parameterized by an integer $m > 0$.

1. On input (COMMIT, x) from Alice, where $x \in \{0,1\}^m$, and no value x has yet been recorded, internally record x and send delayed output $(\text{COMMITTED}, m)$ to Bob.
2. On input REVEAL from Alice, if a value x has been internally recorded, send delayed output (REVEAL, x) to Bob.

Figure 2.3: Commitment functionality $\mathcal{F}_{\text{COM}}$.

After receiving input OK from both Alice and Bob, randomly sample a fair coin $r \leftarrow \{0,1\}$ and send delayed output r to both Alice and Bob.

Figure 2.4: Coin-tossing functionality $\mathcal{F}_{\text{COIN}}$.

CHAPTER 3

Structural Cryptographic Complexity

3.1 Overview

The Universal Composition (UC) framework [16] is the most realistic framework for modelling security of protocols in the presence of malicious adversaries and complex networked contexts like the Internet. UC security for a protocol implies that the protocol may be safely used in any context.

It is known that the strong requirement of UC security implies that most functionalities do not have UC-secure protocols, if only simple communication channels are provided for the protocol (i.e., no additional setup assumption). Canetti and Fischlin [17] showed that the commitment functionality $\mathcal{F}_{\text{COM}}$ is not securely realizable. Canetti, Kushilevitz, and Lindell [22] proved that most 2-party SFE functionalities are also not securely realizable.

The common theme in both of these impossibility results is that *the simulator has no advantage* when the protocol uses only a simple communication channel. More formally, any information about Alice's input that a simulator can compute by interacting with a passively corrupt Alice could also be computed by a corrupt Bob interacting in the protocol with an honest Alice. Thus, if the definition of a functionality requires the simulator to extract more information from Alice than is legitimate for Bob to learn in the ideal interaction, then that functionality is not securely realizable.

3.1.1 Our Results

In this chapter, we develop and apply a new technical framework called *splittability* which greatly generalizes these previous impossibility results in the UC framework. First, splittability provides a complete characterization of secure realizability, instead of just being useful as a necessary condition as in the previous results. Next, our new techniques apply to *completely arbitrary* functionalities, even multi-party, randomized, and reactive functionalities. Indeed, any functionality which can be expressed in the UC model can be considered in our framework. Furthermore, our results apply uniformly to the PPT and computationally unbounded settings. Finally, while the previous works considered protocols that used only authenticated private channels, we model the communication channel itself as an arbitrary functionality.

Splittability. We first motivate the definitions of splittability in the simpler setting for 2-party functionalities of a certain form, which we present in Section 3.2. We say that a 2-party functionality $\mathcal{F}$ is *splittable* if, informally, there is a way to synchronize two independent instances of $\mathcal{F}$ (by interacting as Alice in one instance and Bob in the other instance) so that together they behave as a single instance of $\mathcal{F}$. We express this requirement formally in the language of the UC framework by demanding that no environment can distinguish between a single instance of $\mathcal{F}$ and two independent instances synchronized in this way.

Our more general definition is as follows. Let $\mathcal{F}$ and $\mathcal{G}$ be two (m-party) functionalities. We view $\mathcal{G}$ as a potential "communication channel" for a protocol, and say that $\mathcal{F}$ is splittable with respect to $\mathcal{G}$ (written $\mathcal{F} \prec \mathcal{G}$) if no environment can distinguish between a single instance of $\mathcal{F}$ and many independent instances of $\mathcal{F}$ appropriately synchronized, where the synchronization must take place across the channel $\mathcal{G}$. Like our formulation of the UC framework itself, this definition is general and can be specialized to either the PPT or computationally unbounded setting. Likewise, our results involving splittability apply uniformly to both computational settings.

Our main technical result is that splittability characterizes secure realizability, for a large class of channels $\mathcal{G}$. In particular, call a channel $\mathcal{G}$ *self-splittable* if $\mathcal{G} \prec \mathcal{G}$. We motivate self-splittability as follows. A private channel between Alice and Bob can freely be replaced by private channels between Alice and Eve, and Eve and Bob, if Eve honestly relays messages. Similarly, the self-splittability condition implies that a "direct" channel can be decomposed into several "hops" on the same kind of channel, given the appropriate routing/synchronization. Self-splittable channels are exactly those for which we characterize secure realizability. More formally, we prove:

Theorem. *$\mathcal{G}$ is self-splittable if and only if for all $\mathcal{F}$, $\mathcal{F} \sqsubseteq_{strict} \mathcal{G} \Leftrightarrow \mathcal{F} \prec \mathcal{G}$.*

Interestingly, the proof of this theorem explicitly reflects the intuition that these natural communication channels "give no advantage" to a simulator. We use the fact that $\mathcal{F} \prec \mathcal{G}$ to construct a protocol for $\mathcal{F}$ whose simulator

faithfully simulates an honest instance of $\mathcal{G}$. Thus, we suggest that self-splittability is a useful formalization of the intuitive requirement of being a "natural" communication channel.

Our characterization can accommodate functionalities $\mathcal{F}$ which interact directly with the adversary. However, when considering functionalities that utilize delayed outputs, we generally restrict our attention to *non-trivial* protocols. Recall that a non-trivial protocol provides some guarantee that delayed outputs will eventually be delivered, provided that the adversary does not interfere with the protocol execution. Since splittability does not directly imply any such guarantee, we define an analogous condition for *non-trivial splittability*. Then we prove an analog of Theorem 3.13, showing that non-trivial splittability characterizes non-trivial realizability.

Impossibility results. In Section 3.4, we show several applications of splittability to demonstrate its power. Our first result is to re-derive and extend the impossibility results of Canetti and Fischlin [16, 17]:

Theorem. *There is no non-trivial UC-secure realization for the coin-tossing, commitment, or oblivious transfer functionalities, using any self-splittable communication channel (in the PPT or unbounded settings).*

Furthermore, there is no UC-secure realization of zero-knowledge proofs for any language in NP $\setminus$ BPP*, using any self-splittable communication channel, in the PPT setting.*

It is easy to use splittability to show the impossibility of securely realizing a functionality, with respect to *all* self-splittable channels. These proofs all follow the same general outline of showing that the given functionality $\mathcal{F}$ is not splittable with respect to any channel $\mathcal{G}$. The splittability definition requires that two instances of $\mathcal{F}$ be synchronizable so that they act indistinguishably from a single instance of $\mathcal{F}$. However, since this statement involves reasoning about only ideal functionalities, we can easily apply the security properties of the ideal functionality and show that such a synchronization must fail. For instance, any two independent instances of the coin-tossing functionality $\mathcal{F}_{\text{COIN}}$ will output differing bits with probability 1/2 by definition, so they cannot be synchronized to behave like a single instance (which always outputs the same bit to both parties).

We can also easily re-derive and extend the impossibility results of Kidron and Lindell [60]. They generalized the impossibility results from Canetti, Kushilevitz, and Lindell [22], by considering protocols which used private channels as well as access to various key-registration "set-up" functionalities. We can subsume their results by simply observing that the set-up functionalities they consider are all self-splittable.

Characterization of non-reactive functionalities. We use splittability to complete the partial characterization of 2-party non-reactive functionalities initiated by Canetti, Kushilevitz, and Lindell [22]. We give a complete characterization of secure realizability for this class of functionalities, which includes randomized functionalities and functionalities whose behavior (including input/output domain) depends on the global security parameter.

For the special case of SFE functionalities (non-reactive functionalities which are deterministic and whose behavior does not depend on the security parameter), we obtain the following simple combinatorial characterization:

Theorem. *Let $\mathcal{F}$ be a 2-party SFE functionality. Then $\mathcal{F} \sqsubseteq_{nt} \mathcal{F}_{\text{PVT}}$ if and only if $\mathcal{F}$ is isomorphic to a function of the form $\mathcal{F}(x, y) = x$. Furthermore, if $\mathcal{F}$ is not isomorphic to such a function, then $\mathcal{F} \not\sqsubseteq_{nt} \mathcal{G}$ for any self-splittable channel $\mathcal{G}$.*

Thus, only the absolute simplest SFE functionalities are UC-securely realizable. Again, the proof of this characterization crucially uses splittability.

Results for multi-party functionalities. We use the 2-party characterization and a well-known *partitioning argument* to derive some necessary conditions for multi-party SFE functionalities. Namely, if $\mathcal{F}$ is a securely realizable m-party SFE functionality, then all of its 2-party restrictions (SFE functionalities obtained by partitioning $\{1, \ldots, m\}$ into $A \cup B$, and collapsing the parties in each part into a single party) are also securely realizable. From this, we can see that every such multi-party SFE functionality has one of the following two forms: a party's output depends only on that party's input, for all but one of the parties; or, a party's input influences only that party's output, for all but one of the parties.

We do not know whether this necessary condition is also sufficient for secure realizability. As a step towards resolving this question, we show that for a special class of 3-party functionalities, this partitioning argument is both necessary and sufficient.

3.2 Splittability of Regular 2-Party Functionalities

For expositional clarity, first we present a special case of our splittability definitions and results, which nonetheless captures the essential intuition of our more general results. The simplified case involves functionalities of the following special form.

Definition 3.1 (Regular functionalities)**.** *We say that a functionality $\mathcal{F}$ is* regular *if it does not directly interact with the adversary (when no parties are corrupted), and its behavior does not depend on which parties are corrupted.*

More formally, $\mathcal{F}$ is regular if its transition function is insensitive to the contents of its corruption input tape and its communication tape with the adversary, and it never writes to its communication tape with the adversary.

We now develop the theory of splittability for the special case of *regular 2-party functionalities*. Splittability will provide a complete characterization of secure realizability for these functionalities. Note that this class of functionalities includes even randomized and reactive functionalities, and captures many functionalities of practical interest, though it does not yet allow us to consider functionalities that give delayed outputs.

We first define a specific kind of "compound" functionality, which internally simulates the interaction among several ITMs. This approach of considering such compound entities is used throughout our more general splittability characterization as well.

Definition 3.2. *Let $\mathcal{F}$ be a regular 2-party functionality and $\mathcal{T}$ be an ITM with two communication tapes. Define $\mathcal{F}_{\text{SPLIT}}^{\mathcal{T}}$ as the compound functionality that does the following (See Figure 3.1):*

$\mathcal{F}_{\text{SPLIT}}^{\mathcal{T}}$ internally simulates an instance of $\mathcal{T}$ and two independent instances of $\mathcal{F}$, which we call $\mathcal{F}_L$ and $\mathcal{F}_R$. The functionality $\mathcal{F}_{\text{SPLIT}}^{\mathcal{T}}$ is a 2-party functionality, and thus has two external communication tapes, associated with parties 1 and 2.[1] *We link the external party-1 tape with the party-1 tape of $\mathcal{F}_L$, and the external party-2 tape with the party-2 tape of $\mathcal{F}_R$. The other communication tapes of $\mathcal{F}_L$ and $\mathcal{F}_R$ are linked to the two communication tapes of $\mathcal{T}$, respectively.*

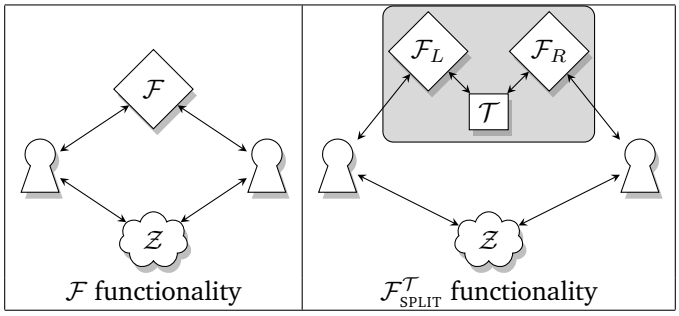

Figure 3.1: 2-party splittability (for regular 2-party $\mathcal{F}$). The shaded box shows $\mathcal{F}_{\text{SPLIT}}^{\mathcal{T}}$.

Intuitively, $\mathcal{F}_{\text{SPLIT}}^{\mathcal{T}}$ defines an interaction in which $\mathcal{T}$ is playing as a man-in-the-middle between two instances of $\mathcal{F}$. We define splittability of $\mathcal{F}$ in terms of $\mathcal{F}_{\text{SPLIT}}^{\mathcal{T}}$.

Definition 3.3. *Let $\mathcal{F}$ be a regular 2-party functionality. We say that $\mathcal{F}$ is* splittable *if there exists an ITM $\mathcal{T}$ such that $\mathcal{F}_{\text{SPLIT}}^{\mathcal{T}}$ is admissible and indistinguishable from $\mathcal{F}$. That is, for all admissible environments $\mathcal{Z}$, we have $\text{EXEC}[\mathcal{Z}, \mathcal{A}_0, \pi_{\mathsf{dummy}}, \mathcal{F}] \approx \text{EXEC}[\mathcal{Z}, \mathcal{A}_0, \pi_{\mathsf{dummy}}, \mathcal{F}_{\text{SPLIT}}^{\mathcal{T}}]$, where $\mathcal{A}_0$ is the dummy adversary that corrupts no one.*

We call $\mathcal{T}$ the translator *corresponding to this splitting of $\mathcal{F}$.*

In other words, $\mathcal{F}$ is *splittable* if there is a way to synchronize the two independent instances of $\mathcal{F}$ (i.e., construct a strategy for $\mathcal{T}$) so that the two instances behave together as a single instance (i.e., are indistinguishable from a single copy of $\mathcal{F}$).

Our main result is that splittability completely characterizes realizability, though on a stronger communication channel.

Theorem 3.4. *Let $\mathcal{F}_{\text{PVT}}^{*}$ be the 2-party private channel functionality $\mathcal{F}_{\text{PVT}}$, except without delayed outputs. That is, messages are delivered immediately, and $\mathcal{F}_{\text{PVT}}^{*}$ does not interact with the adversary.*

Let $\mathcal{F}$ be a regular 2-party functionality. Then $\mathcal{F} \sqsubseteq \mathcal{F}_{\text{PVT}}^{*}$ if and only if $\mathcal{F}$ is splittable.

The restriction to $\mathcal{F}_{\text{PVT}}^{*}$ is natural, and in keeping with our consideration of regular functionalities in this section. Since regular functionalities do not interact with the adversary, an adversary who corrupts no one gets *no information* about the honest parties' outputs. Thus regular functionalities cannot be securely realized on $\mathcal{F}_{\text{PVT}}$, in which the adversary learns (at least) if and when the parties are communicating.

Since the definition of splittability can be specialized for either the unbounded or the PPT setting, the above theorem should be interpreted with $\sqsubseteq^{u}$ in the unbounded setting, and $\sqsubseteq^{p}$ in the PPT setting (with the appropriate notion of admissibility substituted into the theorem statement).

To prove the theorem, we first make the following simple but useful observation:

[1] For uniformity with the subsequent sections, which deal with multi-party functionalities, we do not refer to the two parties as Alice in Bob in this section.

Observation 3.5. *Let D be a "dummy" ITM with two communication tapes, meaning that it does nothing but copy every new message from one tape to the other. In any execution involving two ITMs M_1 and M_2 with a shared communication tape, an identical outcome is achieved by executing M_1, M_2, and an instance of D, where M_1 and D share a communication tape, and M_2 and D share a communication tape.*

The standard dummy adversary is such a machine D, but note that so is $\mathcal{F}_{\text{PVT}}^*$. In fact, it is for this reason that $\mathcal{F}_{\text{PVT}}^*$ has a special status in our results.

Proof (of Theorem 3.4). ($\Rightarrow$) Suppose π is a secure protocol for $\mathcal{F}$ in the $\mathcal{F}_{\text{PVT}}^*$-hybrid setting, where $\mathcal{S}_X$ is the simulator for the dummy adversary which leaves parties $X \subseteq \{1,2\}$ uncorrupted.

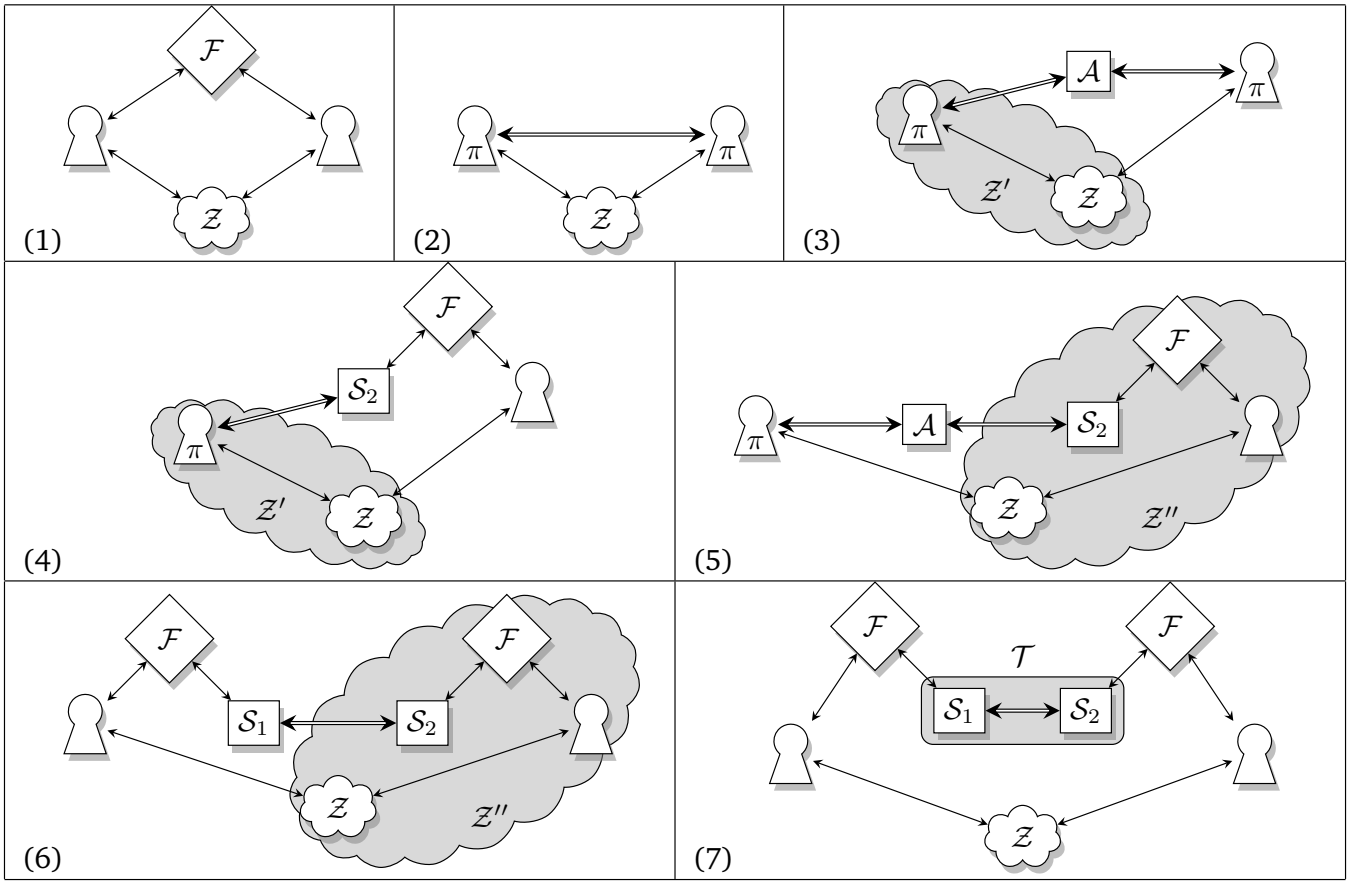

Figure 3.2: Steps in the proof of Theorem 3.4. Instances of $\mathcal{F}_{\text{PVT}}^*$ drawn as double arrows.

Take any environment $\mathcal{Z}$ and consider an ideal-world execution defined by $\text{EXEC}[\mathcal{Z}, \mathcal{A}_0, \pi, \mathcal{F}_{\text{PVT}}^*]$, where $\mathcal{A}_0$ is the dummy adversary that corrupts no one. Since $\mathcal{F}_{\text{PVT}}^*$ is regular, it does not communicate with $\mathcal{A}_0$. Since communicating with the functionality is the only role of such a dummy adversary, we can assume without loss of generality that $\mathcal{Z}$ does not communicate with $\mathcal{A}_0$.

By the security of π, this execution is indistinguishable from $\text{EXEC}[\mathcal{Z}, \mathcal{S}_{\{1,2\}}, \pi_{\mathsf{dummy}}, \mathcal{F}]$. Again, since $\mathcal{F}$ is regular, $\mathcal{S}_{\{1,2\}}$ does not communicate with $\mathcal{F}$; nor does it communicate with $\mathcal{Z}$, so without loss of generality $\mathcal{S}_{\{1,2\}} = \mathcal{A}_0$. Thus we have

$$\text{EXEC}[\mathcal{Z}, \mathcal{A}_0, \pi, \mathcal{F}_{\text{PVT}}^*] \approx \text{EXEC}[\mathcal{Z}, \mathcal{A}_0, \pi_{\mathsf{dummy}}, \mathcal{F}].$$

To complete the proof, it suffices to construct $\mathcal{T}$ such that $\mathcal{F}_{\text{SPLIT}}^{\mathcal{T}}$ is admissible and $\text{EXEC}[\mathcal{Z}, \mathcal{A}_0, \pi, \mathcal{F}_{\text{PVT}}^*] \approx \text{EXEC}[\mathcal{Z}, \mathcal{A}_0, \pi_{\mathsf{dummy}}, \mathcal{F}_{\text{SPLIT}}^{\mathcal{T}}]$.

Let $\mathcal{Z}'$ be the environment that internally simulates $\mathcal{Z}$ and the instance of π executed by party 1. Environment $\mathcal{Z}'$ uses the external communication tape of π (designed for communicating with the functionality) as its communication tape for communicating with an adversary. Let $\mathcal{A}'$ be the dummy adversary that corrupts party 1 alone. Then

$$\text{EXEC}[\mathcal{Z}, \mathcal{A}_0, \pi, \mathcal{F}_{\text{PVT}}^*] = \text{EXEC}[\mathcal{Z}', \mathcal{A}', \pi, \mathcal{F}_{\text{PVT}}^*],$$

since in the latter execution, we have simply re-packaged the ITMs, moving party 1 inside the environment, and

replaced a single communication tape with a dummy adversary. By the security of π, we have

$$\text{EXEC}[\mathcal{Z}', \mathcal{A}', \pi, \mathcal{F}^*_{\text{PVT}}] \approx \text{EXEC}[\mathcal{Z}', \mathcal{S}_{\{2\}}, \pi_{\mathsf{dummy}}, \mathcal{F}].$$

Now let us consider an environment $\mathcal{Z}''$ which does the following: It internally simulates instances of $\mathcal{Z}$, $\mathcal{S}_{\{2\}}$, $\mathcal{F}$, and party 2 executing π_{dummy}. The communication tapes of these instances are linked in the natural way. The two unlinked communication tapes are: the environment-side communication tape of $\mathcal{S}_{\{2\}}$, which $\mathcal{Z}''$ uses as its tape for communicating with the adversary; and the party-1 communication tape of $\mathcal{Z}$, which $\mathcal{Z}''$ uses for the same purpose. If $\mathcal{A}''$ is the dummy adversary which corrupts party 2 alone, then we have

$$\text{EXEC}[\mathcal{Z}', \mathcal{S}_{\{2\}}, \pi_{\mathsf{dummy}}, \mathcal{F}] = \text{EXEC}[\mathcal{Z}'', \mathcal{A}'', \pi, \mathcal{F}^*_{\text{PVT}}].$$

As before, the executions are identical except that several ITMs have been re-packaged into the environment, and a dummy adversary and $\mathcal{F}^*_{\text{PVT}}$ have replaced a single communication tape.

Again, by the security of π, we have:

$$\text{EXEC}[\mathcal{Z}'', \mathcal{A}'', \pi, \mathcal{F}^*_{\text{PVT}}] \approx \text{EXEC}[\mathcal{Z}'', \mathcal{S}_{\{1\}}, \pi_{\mathsf{dummy}}, \mathcal{F}].$$

We finally define $\mathcal{T}$ as an ITM which internally simulates instances of $\mathcal{S}_{\{1\}}$ and $\mathcal{S}_{\{2\}}$, linking their environment-side communication tapes together. $\mathcal{T}$ uses the remaining communication tape of $\mathcal{S}_{\{1\}}$ as its communication tape for $\mathcal{F}_L$, and the remaining tape of $\mathcal{S}_{\{2\}}$ as the communication tape for $\mathcal{F}_R$. Then we finally have that

$$\text{EXEC}[\mathcal{Z}'', \mathcal{S}_{\{1\}}, \pi_{\mathsf{dummy}}, \mathcal{F}] = \text{EXEC}[\mathcal{Z}, \mathcal{A}_0, \pi_{\mathsf{dummy}}, \mathcal{F}^{\mathcal{T}}_{\text{SPLIT}}],$$

since the latter execution is simply a re-packaging of the former.

Admissibility: We must show that $\mathcal{F}^{\mathcal{T}}_{\text{SPLIT}}$ is an admissible functionality. Admissibility is trivial in the unbounded setting. In this proof, we have applied the security of π against dummy adversaries only, which are always admissible in the PPT setting. By the security of π, substituting an ideal interaction involving the simulator for a real-world interaction involving an admissible adversary preserves the property that the entire system's execution is polynomial-time with overwhelming probability. The other steps in the proof involve only repackagings, so indistinguishability holds, and the running time of the execution is not affected. Thus, the final interaction involving $\mathcal{F}^{\mathcal{T}}_{\text{SPLIT}}$ executes in polynomial time with overwhelming probability. Since the environment $\mathcal{Z}$ was arbitrary, we have that $\mathcal{F}^{\mathcal{T}}_{\text{SPLIT}}$ is admissible, as desired.

($\Leftarrow$) Suppose $\mathcal{F}$ is splittable using translator ITM $\mathcal{T}$; then the following is a secure protocol for $\mathcal{F}$ (see Figure 3.3) in the $\mathcal{F}^*_{\text{PVT}}$-hybrid setting:

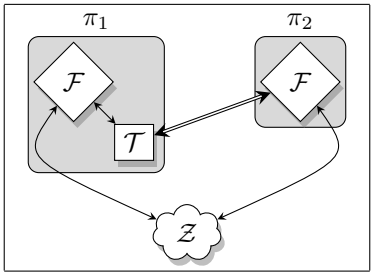

Figure 3.3: Secure protocol (in the $\mathcal{F}^*_{\text{PVT}}$-hybrid setting) for a splittable functionality $\mathcal{F}$. Instance of $\mathcal{F}^*_{\text{PVT}}$ denoted by a double arrow.

- Party 1 internally simulates instances of $\mathcal{F}_L$ and $\mathcal{T}$, with linked communication tape as in the splittability definition. He treats the unlinked communication tape of $\mathcal{F}_L$ as his environment-side communication tape, and the unlinked communication tape of $\mathcal{T}$ as his functionality-side communication tape.
- Party 2 simply runs $\mathcal{F}_R$, interpreting its party-1-side tape as the protocol's functionality-side tape, and its party-2-side tape as the protocol's environment-side tape.

It is easy to see that when neither party is corrupt, the execution of this protocol over $\mathcal{F}^*_{\text{PVT}}$ is simply a re-packaging of $\text{EXEC}[\mathcal{Z}, \mathcal{A}_0, \pi_{\mathsf{dummy}}, \mathcal{F}^{\mathcal{T}}_{\text{SPLIT}}]$. As such, it is indistinguishable from the ideal world by the guarantee of splittability.

The simulators for corrupt parties are quite simple: For a dummy adversary that corrupts party 1, the simulator is also a dummy adversary; for a dummy adversary that corrupts party 2, the simulator is $\mathcal{T}$ itself. It is easy to see that in both of these cases, the simulation is perfect since the ideal world is a repackaging of the real world. □

3.3 General Theory of Splittability

Protocols in the UC framework are generally modeled to use less idealized channels than the one considered in the previous section ($\mathcal{F}^*_{\text{PVT}}$). For instance, even the standard model of a private channel reveals to the adversary the fact that a message was sent, and typically its length. The UC framework also allows one to define functionalities which interact directly with the adversary, or whose behavior depends on which parties are corrupted. Indeed, this flexibility is often useful in obtaining correct definitions of functionalities.

In this section, we generalize the theory to apply to secure realizability of *completely arbitrary* functionalities, considering *completely arbitrary* functionalities as the "communication channel" for the protocol. Furthermore, our theory extends to multi-party (instead of only 2-party) functionalities.

3.3.1 Notational Conventions

In our proofs relating to splittability, we will often construct and manipulate "compound" ITMs which simulate other ITMs and their communication. In this section, we develop a convenient notation for reasoning about such compound ITMs, which allows for simple symbolic manipulation. Using this notation, we can carry out the proof at a reasonably high level.

Let $\mathcal{F}$ be an m-party functionality. $\mathcal{F}$ has communication tapes for the parties $\{1, \ldots, m\}$, as well as one for the adversary. By convention, we consider the adversary to be the $(m+1)$th party to the functionality. If $X \subseteq \{1, \ldots, m\}$ are the indices of uncorrupted parties in some execution, then we denote by $\overline{X} = \{1, \ldots, m+1\} \setminus X$ the indices of the corrupted parties (always containing party $m+1$).

We use labeled arrows to denote different sets of communication tapes for the ITMs in our system. For instance, an expression of the form "$\overset{a}{\leftrightarrow}\ \mathcal{M}\ \overset{b}{\leftrightarrow}$" will denote an ITM $\mathcal{M}$ with its set of communication tapes partitioned into two sets, labeled by a and b. In our notation, the left-to-right ordering is significant, and the labels will be arranged so that if ITM $\mathcal{M}_1$ has communication tapes "$\overset{a}{\leftrightarrow}$" to its right, and $\mathcal{M}_2$ has communication tapes "$\overset{a}{\leftrightarrow}$" to its left, then it will be natural for these tapes to be linked, allowing $\mathcal{M}_1$ and $\mathcal{M}_2$ to communicate.

- If $\mathcal{F}$ is an m-party functionality, then it has $m+1$ communication tapes. For $X \subseteq \{1, \ldots, m\}$, we write $\overset{X:}{\longleftrightarrow}\ \mathcal{F}\ \overset{:\overline{X}}{\longleftrightarrow}$ to denote $\mathcal{F}$ as initialized in the case where the adversary corrupts parties with indices $\overline{X}$. The arrow "$\overset{X:}{\longleftrightarrow}$" denotes the set of communication tapes for parties indexed by X; likewise, the arrow "$\overset{:\overline{X}}{\longleftrightarrow}$" denotes the communication tapes for parties indexed by $\overline{X}$.

  Thus in our notation, a functionality will always communicate with honest parties to its left and corrupt parties to its right. The colons in our notation are important to distinguish the "parity" of communication tapes. More concretely, the colon will always be "in the direction of" the functionality (or away from the environment). Thus, in $\mathcal{N}\ \overset{X:}{\longleftrightarrow}\ \mathcal{M}$, the ITM $\mathcal{M}$ is expecting to interact externally with parties indexed by X, and $\mathcal{N}$ is interacting in the role of those parties.

- A protocol ITM π has two communication tapes, one intended for the environment and one intended for the functionality. We write $\overset{i:}{\longleftrightarrow}\ \pi\ \overset{i:}{\longleftrightarrow}$ to denote the protocol initialized with party index i. The arrow on the left is associated with the protocol's environment-side communication tape, and the arrow on the right with its functionality-side communication tape. This keeps with our convention of the ":" symbol pointing away from the environment and towards the functionality.

- An adversary/simulator $\mathcal{S}$ has communication tapes intended for the environment and for the functionality. Let $X \subseteq \{1, \ldots, m\}$. Then we write $\overset{:\overline{X}}{\longleftrightarrow}\ \mathcal{S}\ \overset{:\overline{X}}{\longleftrightarrow}$ to denote an adversary who corrupts the parties indexed by $\overline{X}$. On the left are its communication tapes for the functionality, and on the right are its communication tapes for the environment. Note that this follows our convention of a functionality interacting with corrupt parties on its right side.

- Finally, in our definition of splittability, we introduce a new type of ITM called a *translator*. We write $\overset{:\overline{X}}{\longleftrightarrow}\ \mathcal{T}\ \overset{X:}{\longleftrightarrow}$ to denote a translator $\mathcal{T}$. It will have communication tapes on the left indexed by a set of corrupt parties $\overline{X}$, and communication tapes on the right indexed by the complementary set X.

Compound ITMs. These notational conventions allow us to express compositions and of several ITMs in the following ways:

- When we have two ITMs $\overset{a}{\leftrightarrow} \mathcal{M}_1 \overset{b}{\leftrightarrow}$ and $\overset{b}{\leftrightarrow} \mathcal{M}_2 \overset{c}{\leftrightarrow}$ whose labels b match completely (both their set of indices and their "parity"), we may write $\overset{a}{\leftrightarrow} \mathcal{M}_1 \overset{b}{\leftrightarrow} \mathcal{M}_2 \overset{c}{\leftrightarrow}$ to denote the natural "compound" ITM which internally simulates both $\mathcal{M}_1$ and $\mathcal{M}_2$, linking the communication tapes labeled by b. The compound ITM leaves the remaining communication tapes (labeled by a and c) unlinked.
- Let $X = \{1, \ldots, k\}$ without loss of generality, and let b be any label in our notation. When we have ITMs $\overset{X:}{\longleftrightarrow} \mathcal{M} \overset{b}{\leftrightarrow}$ and $\{\overset{i:}{\leftrightarrow} \pi \overset{i:}{\leftrightarrow} | \ i \in X\}$, we may write:

$$\overset{X:}{\longleftrightarrow} \left\{ \begin{array}{c} \overset{1:}{\leftrightarrow} \pi \overset{1:}{\leftrightarrow} \\ \vdots \\ \overset{k:}{\leftrightarrow} \pi \overset{k:}{\leftrightarrow} \end{array} \right\} \overset{X:}{\longleftrightarrow} \mathcal{M} \overset{b}{\leftrightarrow}$$

  to denote the compound ITM which internally simulates $\mathcal{M}$ and k instances of π, and links the corresponding communication tapes.

Any (possibly compound) ITM $\mathcal{M}$ of the form $\overset{X:}{\longleftrightarrow} \mathcal{M} \overset{:\overline{X}}{\longleftrightarrow}$ has an interface like that of a functionality, and we interpret the ITM as such. We say that two (compound) functionalities $\overset{X:}{\longleftrightarrow} \mathcal{F}_1 \overset{:\overline{X}}{\longleftrightarrow}$ and $\overset{X:}{\longleftrightarrow} \mathcal{F}_2 \overset{:\overline{X}}{\longleftrightarrow}$ are *indistinguishable* if for all environments $\mathcal{Z}$,

$$\text{EXEC}[\mathcal{Z}, \mathcal{A}_X, \pi_{\mathsf{dummy}}, \mathcal{F}_1] \approx \text{EXEC}[\mathcal{Z}, \mathcal{A}_X, \pi_{\mathsf{dummy}}, \mathcal{F}_2],$$

where $\mathcal{A}_X$ is the dummy adversary that corrupts parties indexed by $\overline{X}$, and π_{dummy} is the dummy protocol. As a shorthand for this definition, we write simply:

$$\overset{X:}{\longleftrightarrow} \mathcal{F}_1 \overset{:\overline{X}}{\longleftrightarrow} \quad \approx \quad \overset{X:}{\longleftrightarrow} \mathcal{F}_2 \overset{:\overline{X}}{\longleftrightarrow}.$$

Secure realizability in our notation. We can now rephrase the UC security definition in terms of our notation. Let $\mathcal{F}$ be an m-party functionality. We say that π is a *strict* secure realization of $\mathcal{F}$ in the $\mathcal{G}$-hybrid setting (that is, $\mathcal{F} \sqsubseteq_{strict} \mathcal{G}$) if for all $X = \{x_1, \ldots, x_k\} \subseteq \{1, \ldots, m\}$, there exists a simulator $\mathcal{S}_X$ such that the following two compound functionalities are indistinguishable:

$$\overset{X:}{\longleftrightarrow} \left\{ \begin{array}{c} \overset{x_1:}{\leftrightarrow} \pi \overset{x_1:}{\leftrightarrow} \\ \vdots \\ \overset{x_k:}{\leftrightarrow} \pi \overset{x_k:}{\leftrightarrow} \end{array} \right\} \overset{X:}{\longleftrightarrow} \mathcal{G} \overset{:\overline{X}}{\longleftrightarrow} \quad \approx \quad \overset{X:}{\longleftrightarrow} \mathcal{F} \overset{:\overline{X}}{\longleftrightarrow} \mathcal{S}_X \overset{:\overline{X}}{\longleftrightarrow}.$$

Note that we do not need to explicitly mention the dummy protocol. In the traditional definition, the simulator $\mathcal{S}_X$ corresponds to the simulator for the dummy adversary that corrupts $\overline{X}$.

Repackaging and rewriting. Finally, we observe the following *rewriting rule*, which generalizes the technique used in the proof of Theorem 3.4 of repackaging component ITMs into and out of the environment:

Lemma 3.6. *If functionalities $\overset{X:}{\longleftrightarrow} \mathcal{M}_1 \overset{:\overline{X}}{\longleftrightarrow}$ and $\overset{X:}{\longleftrightarrow} \mathcal{M}_2 \overset{:\overline{X}}{\longleftrightarrow}$ are indistinguishable, then also*

$$\overset{Y:}{\longleftrightarrow} \mathcal{M}_L \overset{X:}{\longleftrightarrow} \mathcal{M}_1 \overset{:\overline{X}}{\longleftrightarrow} \mathcal{M}_R \overset{:\overline{Y}}{\longleftrightarrow} \quad \approx \quad \overset{Y:}{\longleftrightarrow} \mathcal{M}_L \overset{X:}{\longleftrightarrow} \mathcal{M}_2 \overset{:\overline{X}}{\longleftrightarrow} \mathcal{M}_R \overset{:\overline{Y}}{\longleftrightarrow},$$

for any $Y \subseteq \{1, \ldots, m\}$ and any admissible (possibly compound) ITMs $\mathcal{M}_L$ and $\mathcal{M}_R$.

Proof. Consider an arbitrary environment $\mathcal{Z}$ interacting with $\overset{Y:}{\longleftrightarrow} \mathcal{M}_L \overset{X:}{\longleftrightarrow} \mathcal{M}_1 \overset{:\overline{X}}{\longleftrightarrow} \mathcal{M}_R \overset{:\overline{Y}}{\longleftrightarrow}$.

Let $\mathcal{Z}'$ denote an environment which internally simulates instances of $\mathcal{Z}$, $\mathcal{M}_L$ and $\mathcal{M}_R$, with communication tapes linked as they are in the above interaction.[2] Clearly, the original execution is identical to $\mathcal{Z}'$ interacting with

[2] According to the definition of *admissible* machines, the composition of these machines with the environment machine is itself admissible.

$\overset{X:}{\longleftrightarrow} \mathcal{M}_1 \overset{:\overline{X}}{\longleftrightarrow}$.

We may apply the indistinguishability of $\mathcal{M}_1$ and $\mathcal{M}_2$ with respect to this environment $\mathcal{Z}'$, so that $\mathcal{Z}'$ interacting with $\mathcal{M}_2$ is an indistinguishable execution from the original. Finally, it does not affect the execution to repackage $\mathcal{M}_L$ and $\mathcal{M}_R$ outside of the environment, which corresponds to the original environment $\mathcal{Z}$ interacting with $\overset{Y:}{\longleftrightarrow} \mathcal{M}_L \overset{X:}{\longleftrightarrow} \mathcal{M}_2 \overset{:\overline{X}}{\longleftrightarrow} \mathcal{M}_R \overset{:\overline{Y}}{\longleftrightarrow}$. □

3.3.2 General Splittability

Definition 3.7. *We say that an m-party functionality $\mathcal{F}$ is* splittable *with respect to another m-party functionality $\mathcal{G}$ if there exist translator ITMs $\{\mathcal{T}_X \mid X \subseteq \{1,\ldots,m\}\}$ such that for every $X = \{x_1,\ldots,x_k\} \subseteq \{1,\ldots,m\}$, the following two compound functionalities are indistinguishable:*

$$\overset{X:}{\longleftrightarrow} \mathcal{F} \overset{:\overline{X}}{\longleftrightarrow} \mathcal{T}_X \overset{X:}{\longleftrightarrow} \mathcal{G} \overset{:\overline{X}}{\longleftrightarrow} \quad \approx \quad \overset{X:}{\longleftrightarrow} \left\{ \begin{array}{c} \overset{x_1:}{\longleftrightarrow} \mathcal{F} \overset{:\overline{x_1}}{\longleftrightarrow} \mathcal{T}_{x_1} \overset{x_1:}{\longleftrightarrow} \\ \vdots \\ \overset{x_k:}{\longleftrightarrow} \mathcal{F} \overset{:\overline{x_k}}{\longleftrightarrow} \mathcal{T}_{x_k} \overset{x_k:}{\longleftrightarrow} \end{array} \right\} \overset{X:}{\longleftrightarrow} \mathcal{G} \overset{:\overline{X}}{\longleftrightarrow}.$$

When this is the case, we write $\mathcal{F} \prec \mathcal{G}$.

Intuitively, an instance of $\mathcal{F}$ interacting with honest parties indexed by X may be "split" into separate independent instances of $\mathcal{F}$, one for each of the honest parties. The translator ITMs, which can communicate only via $\mathcal{G}$, must synchronize all of the instances of $\mathcal{F}$ so that they behave as a single instance.

As with all of our results in this chapter, splittability can be specialized to either the unbounded or PPT settings. We superscript the $\prec$ relation as $\prec^u$ and $\prec^p$ to indicate these specific computational model, respectively.

Example. For comparison with the simpler splittability definition for 2-party regular functionalities, we show the general definition of splittability restricted to the two-party case. For two-party functionalities, all but one of the requirements in the splittability definition are tautological, leaving only the requirement for $X = \{1,2\}$. Thus we have:

Definition 3.8. *Let $\mathcal{F}$ and $\mathcal{G}$ be 2-party functionalities. Then $\mathcal{F}$ is splittable with respect to $\mathcal{G}$ if and only if there exist translator ITMs $\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_{12}$ such that:*

$$\overset{1,2:}{\longleftrightarrow} \mathcal{F} \overset{:3}{\longleftrightarrow} \mathcal{T}_{12} \overset{1,2:}{\longleftrightarrow} \mathcal{G} \overset{:3}{\longleftrightarrow} \quad \approx \quad \overset{1,2:}{\longleftrightarrow} \left\{ \begin{array}{l} \overset{1:}{\longleftrightarrow} \mathcal{F} \overset{:2,3}{\longleftrightarrow} \mathcal{T}_1 \overset{1:}{\longleftrightarrow} \\ \overset{2:}{\longleftrightarrow} \mathcal{F} \overset{:1,3}{\longleftrightarrow} \mathcal{T}_2 \overset{2:}{\longleftrightarrow} \end{array} \right\} \overset{1,2:}{\longleftrightarrow} \mathcal{G} \overset{:3}{\longleftrightarrow}.$$

See Figure 3.4 for a visual overview of the two-party case. See also Figure 3.5 for one of the splittability conditions ($X = \{1,2,3\}$) for 3-party functionalities.

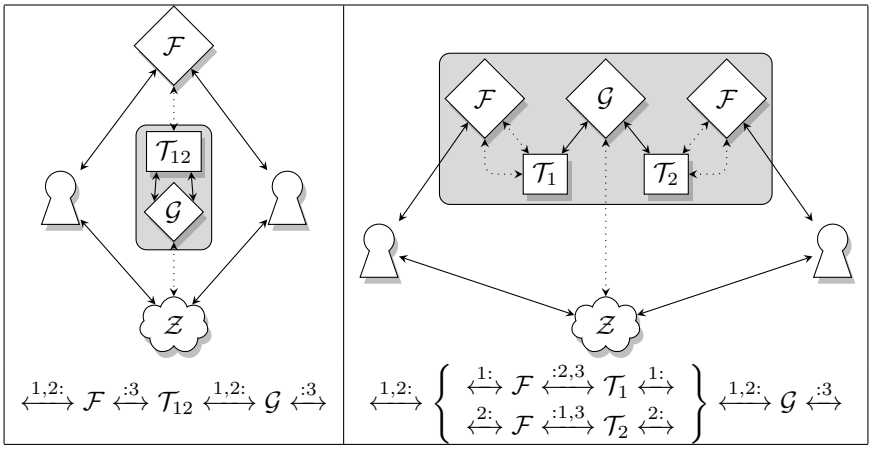

Figure 3.4: Splittability definition for general 2-party functionalities. Dotted lines indicate interactions as an adversary and/or corrupted party.

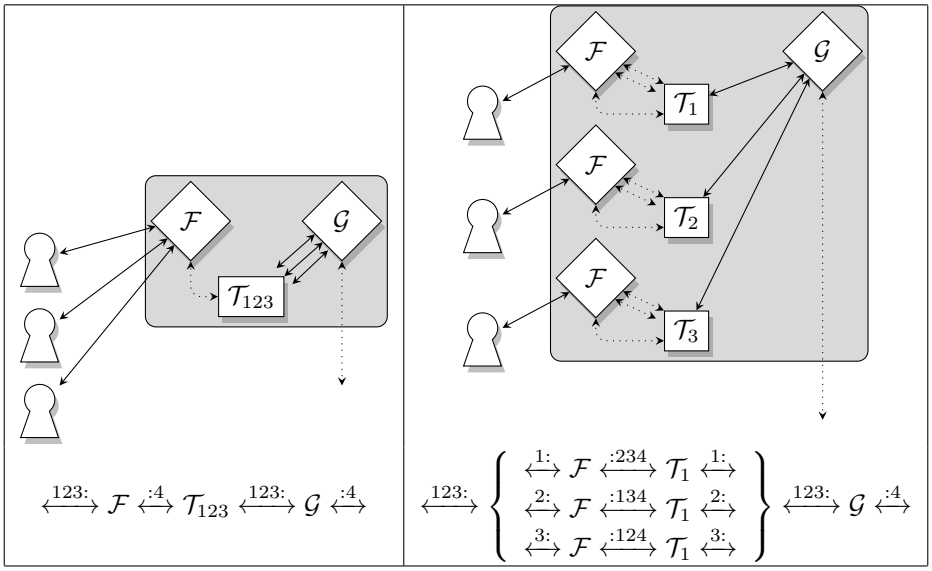

Figure 3.5: One of the conditions in required by the definition of $\mathcal{F} \prec \mathcal{G}$, for 3-party functionalities, and with $X = \{1, 2, 3\}$. Dotted lines indicate interactions as an adversary and/or corrupted party.

As in Section 3.2, we aim to establish a relationship between splittability and realizability. Our main technical results relating the two notions are proven in the following two lemmas. Again, we emphasize that the statements of these lemmas can be specialized to the computationally unbounded and the PPT settings.

Lemma 3.9. *If $\mathcal{F} \prec \mathcal{G}$ then $\mathcal{F} \sqsubseteq_{strict} \mathcal{G}$.*

Proof. Suppose that $\mathcal{F} \prec \mathcal{G}$, using translator ITMs $\{\mathcal{T}_X^{\mathcal{F}\prec\mathcal{G}} \mid X \subseteq \{1, \ldots, m\}\}$. Then we claim that the protocol wherein party i executes $\overset{i:}{\longleftrightarrow}\ \mathcal{F}\ \overset{:i}{\longleftrightarrow}\ \mathcal{T}_i^{\mathcal{F}\prec\mathcal{G}}\ \overset{i:}{\longleftrightarrow}$ (following the notational conventions for protocol ITMs) is a secure protocol for $\mathcal{F}$ in the $\mathcal{G}$-hybrid setting (which uses only one instance of $\mathcal{G}$ for communication). To show the security of this protocol, we must show that the UC security definition holds. For each subset of parties $X = \{x_1, \ldots, x_k\}$, the splittability condition is the following (with grouping brackets added for emphasis):

$$\overset{X:}{\longleftrightarrow}\left\{\begin{array}{c}\overset{x_1:}{\longleftrightarrow}\ [\mathcal{F}\ \overset{:\overline{x_1}}{\longleftrightarrow}\ \mathcal{T}_{x_1}^{\mathcal{F}\prec\mathcal{G}}]\ \overset{x_1:}{\longleftrightarrow}\\ \vdots\\ \overset{x_k:}{\longleftrightarrow}\ [\mathcal{F}\ \overset{:\overline{x_k}}{\longleftrightarrow}\ \mathcal{T}_{x_k}^{\mathcal{F}\prec\mathcal{G}}]\ \overset{x_k:}{\longleftrightarrow}\end{array}\right\}\overset{X:}{\longleftrightarrow}\ \mathcal{G}\ \overset{:\overline{X}}{\longleftrightarrow}$$

$$\approx \quad \overset{X:}{\longleftrightarrow}\ \mathcal{F}\ \overset{:\overline{X}}{\longleftrightarrow}\ [\mathcal{T}_X^{\mathcal{F}\prec\mathcal{G}}\ \overset{X:}{\longleftrightarrow}\ \mathcal{G}]\ \overset{:\overline{X}}{\longleftrightarrow}\ .$$

Thus, the ITM $\overset{:\overline{X}}{\longleftrightarrow}\ \mathcal{T}_X^{\mathcal{F}\prec\mathcal{G}}\ \overset{X:}{\longleftrightarrow}\ \mathcal{G}\ \overset{:\overline{X}}{\longleftrightarrow}$ satisfies the condition to be the simulator for the dummy adversary who corrupts parties indexed by $\overline{X}$. We conclude that the above protocol is secure. □

Lemma 3.10. *If $\mathcal{F} \sqsubseteq_{strict} \mathcal{G} \prec \mathcal{H}$, then $\mathcal{F} \prec \mathcal{H}$.*

Our proof of this lemma generalizes the proof of Theorem 3.4.

Proof. Suppose $\mathcal{F} \sqsubseteq_{strict} \mathcal{G}$ as witnessed by a protocol π and corresponding simulators $\{\mathcal{S}_X^{\mathcal{F}\sqsubseteq\mathcal{G}} \mid X \subseteq \{1, \ldots, m\}\}$; and that $\mathcal{G} \prec \mathcal{H}$ using translators $\{\mathcal{T}_X^{\mathcal{G}\prec\mathcal{H}} \mid X \subseteq \{1, \ldots, m\}\}$.

To show that $\mathcal{F} \prec \mathcal{H}$, we must demonstrate suitable translators $\{\mathcal{T}_X^{\mathcal{F}\prec\mathcal{H}} \mid X \subseteq \{1, \ldots, m\}\}$. Applying the above

conditions, we have that for each subset of parties $X = \{x_1, \ldots, x_k\}$:

$$
\begin{aligned}
&\xleftrightarrow{X:} \mathcal{F} \xleftrightarrow{:\overline{X}} [\mathcal{S}_X^{\mathcal{F}\sqsubseteq\mathcal{G}} \xleftrightarrow{:\overline{X}} \mathcal{T}_X^{\mathcal{G}\prec\mathcal{H}}] \xleftrightarrow{X:} \mathcal{H} \xleftrightarrow{:\overline{X}} \\
&\quad\approx\quad \xleftrightarrow{X:} \left\{ \begin{array}{c} \xleftrightarrow{x_1:} \pi \xleftrightarrow{x_1:} \\ \vdots \\ \xleftrightarrow{x_k:} \pi \xleftrightarrow{x_k:} \end{array} \right\} \xleftrightarrow{X:} \mathcal{G} \xleftrightarrow{:\overline{X}} \mathcal{T}_X^{\mathcal{G}\prec\mathcal{H}} \xleftrightarrow{X:} \mathcal{H} \xleftrightarrow{:\overline{X}} \\
&\quad\approx\quad \xleftrightarrow{X:} \left\{ \begin{array}{c} \xleftrightarrow{x_1:} \pi \xleftrightarrow{x_1:} \\ \vdots \\ \xleftrightarrow{x_k:} \pi \xleftrightarrow{x_k:} \end{array} \right\} \xleftrightarrow{X:} \left\{ \begin{array}{c} \xleftrightarrow{x_1:} \mathcal{G} \xleftrightarrow{:\overline{x_1}} \mathcal{T}_{x_1}^{\mathcal{G}\prec\mathcal{H}} \xleftrightarrow{x_1:} \\ \vdots \\ \xleftrightarrow{x_k:} \mathcal{G} \xleftrightarrow{:\overline{x_k}} \mathcal{T}_{x_k}^{\mathcal{G}\prec\mathcal{H}} \xleftrightarrow{x_k:} \end{array} \right\} \xleftrightarrow{X:} \mathcal{H} \xleftrightarrow{:\overline{X}} \\
&\quad\equiv\quad \xleftrightarrow{X:} \left\{ \begin{array}{c} \xleftrightarrow{x_1:} \pi \xleftrightarrow{x_1:} \mathcal{G} \xleftrightarrow{:\overline{x_1}} \mathcal{T}_{x_1}^{\mathcal{G}\prec\mathcal{H}} \xleftrightarrow{x_1:} \\ \vdots \\ \xleftrightarrow{x_k:} \pi \xleftrightarrow{x_k:} \mathcal{G} \xleftrightarrow{:\overline{x_k}} \mathcal{T}_{x_k}^{\mathcal{G}\prec\mathcal{H}} \xleftrightarrow{x_k:} \end{array} \right\} \xleftrightarrow{X:} \mathcal{H} \xleftrightarrow{:\overline{X}} \\
&\quad\approx\quad \xleftrightarrow{X:} \left\{ \begin{array}{c} \xleftrightarrow{x_1:} \mathcal{F} \xleftrightarrow{:\overline{x_1}} [\mathcal{S}_{x_1}^{\mathcal{F}\sqsubseteq\mathcal{G}} \xleftrightarrow{:\overline{x_1}} \mathcal{T}_{x_1}^{\mathcal{G}\prec\mathcal{H}}] \xleftrightarrow{x_1:} \\ \vdots \\ \xleftrightarrow{x_k:} \mathcal{F} \xleftrightarrow{:\overline{x_k}} [\mathcal{S}_{x_k}^{\mathcal{F}\sqsubseteq\mathcal{G}} \xleftrightarrow{:\overline{x_k}} \mathcal{T}_{x_k}^{\mathcal{G}\prec\mathcal{H}}] \xleftrightarrow{x_k:} \end{array} \right\} \xleftrightarrow{X:} \mathcal{H} \xleftrightarrow{:\overline{X}} .
\end{aligned}
$$

The steps in this derivation follow due to the security of the π protocol, splittability of $\mathcal{G}$ with respect to $\mathcal{H}$, simple rearranging/simplification, and the security of the π protocol again (applied k times), respectively.

Thus, setting

$$\mathcal{T}_X^{\mathcal{F}\prec\mathcal{H}} \quad = \quad \xleftrightarrow{:\overline{X}} \mathcal{S}_X^{\mathcal{F}\sqsubseteq\mathcal{G}} \xleftrightarrow{:\overline{X}} \mathcal{T}_X^{\mathcal{G}\prec\mathcal{H}} \xleftrightarrow{X:}$$

completes the proof to show $\mathcal{F} \prec \mathcal{H}$. □

Corollary 3.11. *The splittability relation $\prec$ is transitive.*

Proof. If $\mathcal{F} \prec \mathcal{G} \prec \mathcal{H}$, then $\mathcal{F} \sqsubseteq_{strict} \mathcal{G}$ from Lemma 3.9. Then Lemma 3.10 implies $\mathcal{F} \prec \mathcal{H}$. □

Splittability characterizing realizability. In the simplified exposition of Section 3.2, splittability provided an exact characterization of realizability with respect to the completely private channel functionality $\mathcal{F}_{\text{PVT}}^*$.

A completely ideal characterization would be a generalization of the form: $\mathcal{F} \sqsubseteq_{strict} \mathcal{G}$ if and only if $\mathcal{F} \prec \mathcal{G}$. Unfortunately, this is demonstrably not the case. For instance $\mathcal{F} \sqsubseteq_{strict} \mathcal{F}$ for all $\mathcal{F}$, but it is easy to see that $\mathcal{F} \not\prec \mathcal{F}$ for several functionalities, (for example, $\mathcal{F}_{\text{COM}}$). However, the characterization does generalize for a certain class of interesting functionalities.

Definition 3.12. *Call a functionality $\mathcal{F}$* self-splittable *if $\mathcal{F} \prec \mathcal{F}$.*

When using a self-splittable functionality $\mathcal{G}$ as the "channel" for a protocol, splittability with respect to $\mathcal{G}$ completely characterizes secure realizability. Furthermore, the self-splittable functionalities are the only ones for which this characterization holds.

Theorem 3.13. *$\mathcal{G}$ is self-splittable if and only if for all $\mathcal{F}$, $\mathcal{F} \sqsubseteq_{strict} \mathcal{G} \Leftrightarrow \mathcal{F} \prec \mathcal{G}$.*

Proof. ($\Rightarrow$) $\mathcal{F} \prec \mathcal{G}$ implies $\mathcal{F} \sqsubseteq_{strict} \mathcal{G}$ by Lemma 3.9. Then Lemma 3.10 shows that $\mathcal{F} \sqsubseteq_{strict} \mathcal{G} \prec \mathcal{G}$ implies $\mathcal{F} \prec \mathcal{G}$.

($\Leftarrow$) Suppose that for all $\mathcal{F}$, we have $\mathcal{F} \sqsubseteq_{strict} \mathcal{G} \Leftrightarrow \mathcal{F} \prec \mathcal{G}$. Then since $\mathcal{G} \sqsubseteq_{strict} \mathcal{G}$ unconditionally, we must have $\mathcal{G} \prec \mathcal{G}$. □

Interpreting self-splittability. We propose self-splittability as a useful *formal definition* of what it means to be a "communication channel." It can be easily seen that all typical communication channels (e.g., authenticated or unauthenticated, public or private, multicast or point-to-point), which are often implicitly incorporated into the network model, are self-splittable.

We see from the protocols constructed in the proof of Lemma 3.9 that the simulator for a protocol on a self-splittable channel $\mathcal{G}$ can, without loss of generality, faithfully simulate an honest instance of $\mathcal{G}$ (in fact, even interacting with this instance of $\mathcal{G}$ using the interface for the honest parties). Indeed, this is a property shared

by all standard communication channels — that the simulator gains no advantage over honest parties by being able to simulate the channel itself.

In our simplified exposition, the functionality $\mathcal{F}_{\text{PVT}}^{*}$ had a special status in that it could be freely inserted in place of any communication tape linked between two entities in the network. Self-splittability generalizes this important property, since $\mathcal{F} \prec \mathcal{F}$ implies that "routing" communication through an instance of $\mathcal{F}$ is indistinguishable from routing communication through three instances of $\mathcal{F}$. The $\{T_X\}$ ITMs from the splittability definition effect the routing through the different instances.

Note, however, that the self-splittability of a functionality depends crucially on whether considering unbounded or PPT settings. For instance, a channel which applies a one-way permutation to its input is self-splittable in the unbounded setting, but not in the PPT setting. Indeed, in the PPT setting, there is an advantage in simulating such a channel versus accessing it as an honest party — the simulator gets access to the preimage under the one-way permutation, whereas an honest party cannot compute the preimage.

3.3.3 Non-Trivial Protocols

As discussed in Section 2.2.4, we are most often interested in functionalities with delayed output, and protocols which are *non-trivial*. Recall that a secure protocol for $\mathcal{F}$ in the $\mathcal{G}$-hybrid setting is non-trivial if for every adversary which corrupts no one and eventually delivers all delayed outputs of $\mathcal{G}$, the corresponding simulator eventually delivers all delayed outputs in $\mathcal{F}$.

We can easily incorporate this non-triviality requirement into our framework of splittability, as follows. Recall that $\mathcal{F} \sqsubseteq_{nt} \mathcal{G}$ is defined as $\mathcal{F} \sqsubseteq_{strict} \widetilde{\mathcal{G}}$ via a non-trivial protocol, where $\widetilde{\mathcal{G}}$ is the multi-session version of $\mathcal{G}$ augmented with $\mathcal{F}_{\text{PVT}}$.

Definition 3.14. *Let $\mathcal{F}$ and $\mathcal{G}$ be m-party functionalities and suppose $\mathcal{F} \prec \mathcal{G}$ using translator ITMs $\{\mathcal{T}_X \mid X \subseteq \{1,\ldots,m\}\}$. Recall that in the splittability definition, each ITM $\mathcal{T}_X$ interacts with an instance of $\mathcal{G}$ (on behalf of the honest parties) and with an instance of $\mathcal{F}$ (as the adversary and corrupt parties).*

We say that $\mathcal{F}$ is non-trivially splittable with respect to *$\mathcal{G}$, and write $\mathcal{F} \prec_{nt} \mathcal{G}$, if $\mathcal{T}_{\{1,\ldots,m\}}$ has the property that when it interacts with an instance of $\mathcal{G}$ in which all delayed outputs are eventually delivered, then $\mathcal{T}_{\{1,\ldots,m\}}$ eventually delivers all delayed outputs of $\mathcal{F}$.*

Note that non-triviality for splittability only involves an additional constraint on one of the translator ITMs, just as the definition of non-triviality for protocols only involves an additional constraint on the simulator in one corruption scenario.

Theorem 3.15. *$\widetilde{\mathcal{G}}$ is self-splittable if and only if for all $\mathcal{F}$, $\mathcal{F} \sqsubseteq_{nt} \mathcal{G} \Leftrightarrow \mathcal{F} \prec_{nt} \widetilde{\mathcal{G}}$.*

Proof. It suffices to show that Lemma 3.9 and Lemma 3.10 hold with respect to the $\prec_{nt}$ and $\sqsubseteq_{nt}$ relations.

In Lemma 3.9, we construct a secure protocol from a splitting of $\mathcal{F}$ with respect to $\mathcal{G}$. Suppose $\mathcal{F}$ and $\mathcal{G}$ are m-party functionalities. Then the simulator for this protocol in the case where no parties are corrupt is $\overset{:\overline{X}}{\longleftrightarrow} \mathcal{T}_X^{\mathcal{F} \prec \mathcal{G}} \overset{X:}{\longleftrightarrow} \mathcal{G} \overset{:\overline{X}}{\longleftrightarrow}$. Thus if the real-world adversary eventually delivers all delayed outputs, and the split of $\mathcal{F}$ with respect to $\mathcal{G}$ is non-trivial, then this simulator will will eventually deliver all delayed inputs on $\mathcal{F}$.

In Lemma 3.10, we derive a splitting of $\mathcal{F}$ with respect to $\mathcal{H}$, given that $\mathcal{F} \sqsubseteq \mathcal{G} \prec \mathcal{H}$. Suppose that $\mathcal{F} \sqsubseteq_{nt} \mathcal{G} \prec_{nt} \mathcal{H}$ and let $X = \{1,\ldots,m\}$. Then the proof of Lemma 3.10 constructs the translator $\mathcal{T}_X$ to be $\overset{:\overline{X}}{\longleftrightarrow} \mathcal{S} \overset{:\overline{X}}{\longleftrightarrow} \mathcal{T}'_X \overset{X:}{\longleftrightarrow}$, where $\mathcal{S}$ is the simulator for the protocol that realizes $\mathcal{F} \sqsubseteq \mathcal{G}$, and $\mathcal{T}'_X$ is the translator ITM from $\mathcal{G} \prec \mathcal{H}$. Both of these ITMs have the property that whenever the interface to their right eventually delivers all delayed outputs, then they eventually deliver all outputs on the interface to their left. Thus the proof of Lemma 3.10 constructs a non-trivial splitting in this case. □

3.4 Applications of the Theory

In this section, we apply the general theory developed in the previous section to specific settings and classes of functionalities, to obtain several new, concrete results as easy consequences.

3.4.1 Elementary Impossibility Proofs

A compelling aspect of our splittability characterization is that all previous impossibility results for the UC model can be re-derived quite easily, because the splittability definition involves interactions only with ideal functionalities,

which give ideal guarantees that are easy to interpret and apply. Furthermore, splittability allows us to derive with only one argument impossibility results with respect to *all* natural communication channels (i.e., self-splittable functionalities).

As concrete examples, we can give elementary proofs that the following functionalities have no non-trivially secure protocols using *any* natural communication channel. While these impossibility results are not new (except for our consideration of arbitrary communication channels), we include them here to demonstrate how useful splittability is for deriving impossibility results.

Theorem 3.16. *There is no non-trivial secure realization for any of the following functionalities, using any self-splittable communication channel, in either the PPT or computationally unbounded setting:*

- *Coin-tossing: $\mathcal{F}_{\text{COIN}}$ (Figure 2.4)*
- *Commitment: $\mathcal{F}_{\text{COM}}$ (Figure 2.3)*
- *Oblivious transfer: $\mathcal{F}_{\text{OT}}$ (Figure 2.2)*

Proof. In all of the following examples, we consider a dummy adversary which corrupts no parties, and environments which instruct the dummy adversary to immediately deliver all delayed outputs.

For each of the listed functionalities $\mathcal{F}$, we show that $\mathcal{F} \not\approx_{nt} \mathcal{G}$ for *any* functionality $\mathcal{G}$. Then by Theorem 3.15, we have that $\mathcal{F} \not\sqsubseteq_{nt} \mathcal{G}$ for all self-splittable $\mathcal{G}$. The 2-party splittability definition involves a single indistinguishability constraint between two interactions. The first interaction involves one copy of $\mathcal{F}$ and another ITM acting as an adversary. The second interaction involves two independent instances of $\mathcal{F}$ being synchronized by other ITMs. We will simply show an environment that distinguishes between the two interactions, provided that the splitting of $\mathcal{F}$ is non-trivial.

Coin-tossing: Consider an environment that gives input OK to both parties and outputs 1 if both parties report the same output from $\mathcal{F}_{\text{COIN}}$. This environment outputs 1 with probability 1 in the first interaction, since all delayed outputs are delivered assuming the split is non-trivial. However, in the second interaction, the two instances of $\mathcal{F}_{\text{COIN}}$ are independent, and regardless of the behavior of other ITMs in the compound functionality, the two parties will receive different outputs with probability 1/2.

Commitment: Consider an environment that gives input (COMMIT, b) to Alice for a randomly chosen bit b, waits for Bob to report output COMMITTED, then gives input REVEAL to Alice and outputs 1 if Bob reports output (REVEAL, b). Again, in the first interaction, the environment outputs 1 with probability 1 if the split is non-trivial. However, in the second interaction, a bit must be committed in Bob's instance of $\mathcal{F}_{\text{COM}}$, independent of the environment's choice of b since the other instance of $\mathcal{F}_{\text{COM}}$ has not yet revealed b. Thus Bob will eventually report the wrong output with probability 1/2.

Oblivious transfer: Consider an environment that supplies random input x_0, x_1 for Alice and random input b for Bob, and outputs 1 if Bob reports output x_b. Again, in the first interaction, the environment outputs 1 with probability 1 if the split is non-trivial. However, in the second interaction, input (x'_0, x'_1) is given to Bob's instance of $\mathcal{F}_{\text{OT}}$ with at least one bit of uncertainty about the environment's choice of (x_0, x_1) by the properties of the other $\mathcal{F}_{\text{OT}}$ instance. Thus, Bob will report the wrong output with probability at least 1/4. □

We can also show an impossibility result that is specific to the PPT setting. Let $\mathcal{F}_{\text{ZK}}$ be a *zero-knowledge* functionality, parameterized by a polynomial-time computable relation R. It takes as input (x, w) from Alice, and if $R(x, w) = 1$ it gives delayed output x to Bob. We denote $L_R = \{x \mid \exists w : R(x, w) = 1\}$.

Theorem 3.17. *There is no non-trivial secure realization for $\mathcal{F}_{\text{ZK}}$ when $L_R \in \text{NP} \setminus \text{BPP}$, using any self-splittable communication channel, in the PPT setting.*

Proof. Following the outline of the previous proofs, we consider a class of environments parameterized by pairs (x, w) such that $R(x, w) = 1$. These environments supply input (x, w) to Alice and output 1 if Bob outputs x. As before, these environments output 1 with probability 1 in the first interaction, assuming that the split is non-trivial so that all delayed outputs are delivered. But in the second interaction, the ensemble of ITMs between the two instances of $\mathcal{F}_{\text{ZK}}$ (which execute in the same way for all environments in this class) must compute a witness w' such that $R(x, w') = 1$, given x alone. However, this is not possible for all $x \in L_R$ by a PPT algorithm, unless $L_R \in \text{BPP}$. □

3.4.2 Characterization of Non-Reactive Functionalities

We now use splittability for 2-party functionalities to give an explicit, *combinatorial* characterization for 2-party SFE (which are regular). This subsumes and completes the characterizations initiated by Canetti, Kushilevitz, and Lindell [22], who gave broad impossibility results for several large subclasses of 2-party SFE. We will also strengthen the impossibility results to show that they hold with respect to any self-splittable communication channel.

The impossibility results of Canetti, Kushilevitz, and Lindell [22] were later extended by Kidron and Lindell [60] to the setting where certain "trusted setup" functionalities $\mathcal{G}$ are also available for protocols to use. These extensions can also be shown in our framework by observing that these particular functionalities $\mathcal{G}$ are self-splittable.

Since splittability applies to completely arbitrary 2-party functionalities, we are actually able to characterize a larger class than SFE functionalities; in fact, we will characterize all 2-party non-reactive functionalities. Note that these functionalities may be randomized and have input domains of unbounded size. However, we use a similar convention to describe non-reactive functionalities. Namely, a non-reactive functionality $\mathcal{F}$ waits for inputs x from Alice and y from Bob, then it samples a random tape r and gives delayed output $f_A(x, y, r)$ to Alice, and delayed output $f_B(x, y, r)$ to Bob. For simplicity, we write $f_A(x, y)$ to denote the distribution of outputs $f_A(x, y, r)$ induced by a random choice of r.

Definition 3.18. *Let $\mathcal{F} = (f_A, f_B)$ be a 2-party non-reactive functionality. We say that $\mathcal{F}$ has* unidirectional influence *if one party's output does not depend on the other party's input. That is, if $f_A(x, y) \approx f'_A(x)$ for some randomized function f'_A, or if $f_B(x, y) \approx f'_B(y)$ for some randomized function f'_B. Here, "$\approx$" is meant to denote statistical indistinguishability in the computationally unbounded setting, or computational indistinguishability in the PPT setting. Otherwise $\mathcal{F}$ has* bidirectional influence.

Note that unidirectional influence implies that $f_A(x, y) \approx f_A(x, y')$ for all x, y, and y', by the transitivity of the $\approx$ relation.

Definition 3.19. *Let $\mathcal{F} = (f_A, f_B)$ be a 2-party non-reactive functionality with unidirectional influence; say, the first party's output does not depend on the second party's input. We say that $\mathcal{F}$ is* completely invertible *if there exists admissible ITMs R_1 and R_2 such that for all inputs x, y, the outcome of:*

$$(y^*, s) \leftarrow R_1; f_B\Big(R_2\big(s, f_B(x, y^*)\big), y\Big)$$

is (computationally or statistically, depending on the computational setting) indistinguishable from the random variable $f_B(x, y)$, where the probability is over the randomness of R_1 and R_2 and of both calls to the randomized function $f_B(\cdot, \cdot)$.

Note that complete invertibility is essentially a property of f_B (the randomized function which may depend on the other party's input). It implies that when carefully choosing his input to $\mathcal{F}$, Bob's output from $\mathcal{F}$ is sufficient to compute an input x^* which is "equivalent" to the input x that Alice used, in the sense that $f_B(x^*, y) \approx f_B(x, y)$. This definition succinctly incorporates both the *completely revealing* and *efficiently invertible* properties of Canetti, Kushilevitz, and Lindell [22].

Theorem 3.20. *Let $\mathcal{F} = (f_A, f_B)$ be a 2-party non-reactive functionality. Then $\mathcal{F} \sqsubseteq_{nt} \mathcal{F}_{\text{PVT}}$ if and only if $\mathcal{F}$ has unidirectional influence and is completely invertible.*

Furthermore, if $\mathcal{F} \not\sqsubseteq_{nt} \mathcal{F}_{\text{PVT}}$, then $\mathcal{F} \not\sqsubseteq_{nt} \mathcal{G}$ for any self-splittable $\mathcal{G}$.

Proof. ($\Leftarrow$) Suppose $\mathcal{F}$ has unidirectional influence and is completely invertible. By Theorem 3.13, it suffices to show that $\mathcal{F} \prec_{nt} \mathcal{F}_{\text{PVT}}$. Without loss of generality, suppose that Alice's output is not affected by Bob's input (so $f_A(x, y) \approx f'_A(x)$ for some randomized function f'_A). Then the following is a non-trivial split of $\mathcal{F}$ with respect to $\mathcal{F}_{\text{PVT}}$: The ITM $\mathcal{T}_1$ runs runs $(y^*, s) \leftarrow R_1$, then sends y^* to $\mathcal{F}_L$ and delivers both delayed outputs on $\mathcal{F}_L$. It receives an output $z \leftarrow f_B(x, y^*)$ and computes $x^* \leftarrow R_2(s, z)$. It sends x^* across $\mathcal{F}_{\text{PVT}}$ to $\mathcal{T}_2$. $\mathcal{T}_2$ simply sends x^* to its instance of $\mathcal{F}_R$ and delivers both delayed outputs on $\mathcal{F}_R$.

When $\mathcal{T}_{12}$ is informed that $\mathcal{F}$ has generated delayed outputs, $\mathcal{T}_{12}$ delivers the output for Alice and simulates that a message was sent across $\mathcal{F}_{\text{PVT}}$ from Alice to Bob. If this message is delivered, then $\mathcal{T}_{12}$ delivers the delayed output for Bob in $\mathcal{F}$.

We claim that the two splittability interactions are indistinguishable. By the correctness of R_1 and R_2, Bob's output is indistinguishable between the interactions. By the unidirectional influence property of f_A, Alice's output is indistinguishable between the interactions.

($\Rightarrow$) Suppose $\mathcal{F}$ does not have unidirectional influence or is not completely invertible. It suffices to show that $\mathcal{F} \not\prec_{nt} \mathcal{G}$ for any $\mathcal{G}$. We consider two cases.

First, suppose $\mathcal{F}$ has bidirectional influence. Then there exist inputs x_0, x_1, x_2 for Alice and inputs y_0, y_1, y_2 for Bob such that $f_A(x_0, y_1) \not\approx f_A(x_0, y_2)$ and $f_B(x_1, y_0) \not\approx f_B(x_2, y_0)$. Consider any environment $\mathcal{Z}$ that satisfies the following properties:

- It runs with a dummy adversary that corrupts no parties.
- It chooses random $i, j \leftarrow \{0, 1, 2\}$ and supplies inputs x_i and y_j to Alice and Bob, respectively.
- Its instructions to the dummy adversary are independent of its choice of i and j, and these instructions cause both delayed outputs of $\mathcal{F}$ to be eventually delivered in both splittability interactions (if this is not possible, then no split can be non-trivial, and we are done).
- It waits for both parties to return output, and its final output is a function of only one of the parties' outputs.

Since we are dealing with an arbitrary channel $\mathcal{G}$, we must make the requirements of $\mathcal{Z}$ suitably generic. We will show that for every $\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_{12}$, some environment of this form will distinguish between the two splittability interactions.

Consider the first splittability interaction (involving $\mathcal{F}_L$ and $\mathcal{F}_R$). One of $\{\mathcal{T}_1, \mathcal{T}_2\}$ must be the first to send an input to one of these instances of $\mathcal{F}$, and the choice of the instance is independent of the environment's choice of i and j (since $\mathcal{F}_L$ and $\mathcal{F}_R$ do not release any output until receiving both outputs, and since the adversary's communication is also independent of i and j). By symmetry, suppose an input y^* is sent to $\mathcal{F}_L$ first. With probability at least $1/9$, the environment will select $i = 0$ and j such that $f_A(x_0, y_j) \not\approx f_A(x_0, y^*)$; such a j must exist by the choice of x_0, y_1, and y_2. Then there is some distinguisher with non-negligible advantage ϵ in distinguishing $f_A(x_0, x_j)$ and $f_A(x_0, y^*)$. The environment that runs this distinguisher therefore distinguishes between the two splittability interactions with non-negligible bias $\epsilon/9$.

For the other case, suppose that $\mathcal{F}$ has unidirectional influence, say with Alice's output not depending on Bob's input, but is not completely invertible. Then consider an environment with the following properties;

- It runs with a dummy adversary that corrupts no parties.
- It is parameterized by x and y, and provides input x for Alice, and input y for Bob.
- Its instructions to the dummy adversary are independent of its choice of x and y, and these instructions cause both delayed outputs of $\mathcal{F}$ to be eventually delivered in both splittability interactions.
- It waits for both parties to return output, and its final output is a function of Bob's output.

Again we will show that for any purported splitting of $\mathcal{F}$, one environment of this kind can distinguish between the two interactions in the splittability definition.

Define R_1 as the compound ITM which runs $\mathcal{T}_1$, $\mathcal{T}_2$, $\mathcal{G}$, and the dummy adversary interaction (which is fixed and independent of the environment's choice of x and y), as in the first splittability interaction, until $\mathcal{T}_1$ sends an input to $\mathcal{F}_L$. The output of R_1 is defined as the input y^* sent to $\mathcal{F}_L$, as well as the internal state s of all component ITMs. Then R_2 is defined similarly, computing the input sent by $\mathcal{T}_2$ to $\mathcal{F}_R$ when receiving an output from $\mathcal{F}_L$ and initialized with a given internal state. Then the output of Bob in this splittability interaction is as in the definition of complete invertibility.

But for any such R_1 and R_2, there is a choice of x and y such that the output induced by Bob is distinguishable from $f_B(x, y)$, as is Bob's output in the other splittability interaction. Thus the environment that uses these inputs x and y, and runs the appropriate distinguisher, can distinguish the two splittability interactions with some non-negligible bias. □

SFE functionalities. We have given a combinatorial characterization for the large class of non-reactive functionalities. For the special case of 2-party SFE functionalities (which are deterministic, and have constant-sized input domains), our characterization collapses to a very simple combinatorial condition:

Theorem 3.21. *Let $\mathcal{F}$ be a 2-party SFE functionality. Then $\mathcal{F} \sqsubseteq_{nt} \mathcal{F}_{\text{PVT}}$ if and only if $\mathcal{F}$ is isomorphic to a function of the form $\mathcal{F}(x, y) = x$. Furthermore, if $\mathcal{F}$ is not isomorphic to such a function, then $\mathcal{F} \not\sqsubseteq_{nt} \mathcal{G}$ for any self-splittable channel $\mathcal{G}$.*

We emphasize that this same characterization applies to both the computationally unbounded and PPT settings.

Proof. Let $\mathcal{F} = (f_A, f_B)$. By Theorem 3.20, it suffices to show that for SFE functionalities, unidirectional influence and complete invertibility collapse to the condition of being isomorphic to a function of the form $\mathcal{F}(x, y) = x$.

Unidirectional influence demands that, by symmetry, $f_A(x, y) \approx f'_A(x)$ for some function f'_A. However, when f_A is deterministic, and x is from a finite domain, this condition is equivalent (in both the unbounded and PPT settings) to $f_A(x, y) = f'_A(x)$ for some deterministic function f'_A. By isomorphism-preserving operations, we can have $f_A(x, y) = x$ without loss of generality.

Next, $\mathcal{F}$ must be completely invertible. Similar to before, in the case of SFE, the complete invertibility condition is equivalent to the condition that for some choice of y^*, $f_B(x, y^*)$ determines an x' such that $f_B(x, \cdot) \equiv f_B(x', \cdot)$. In other words, if $f_B(x, \cdot) \not\equiv f_B(x', \cdot)$, then $f_B(x, y^*) \neq f_B(x', y^*)$. It is easy to see that every input for Bob other than y^* is redundant. Thus, we can safely remove these inputs while preserving isomorphism to the original functionality. Then, any inputs x and x' for Alice which have $f_B(x, y^*) = f_B(x', y^*)$ are also redundant, and can be collapsed, preserving isomorphism. Finally, we may consistently re-label Bob's outputs to obtain $f_B(x, y^*) = x$; thus both parties' outputs are $\mathcal{F}(x, y) = x$. □

3.4.3 Results for Multi-Party Functionalities

For functionalities involving more than two parties, the splittability definition is much more complicated, and combinatorial characterizations like Theorem 3.21 seem difficult to come by. Nonetheless, we can use 2-party results to obtain some strong necessary conditions for the multi-party setting.

A well-known technique for studying m-party SFE functionalities is the *partitioning argument*: consider *2-party SFE functionalities* induced by partitioning the m parties into two sets. If the original functionality is realizable, then clearly so is each induced 2-party functionality.

To exploit the partitioning argument, first we extend the notion of *influence* from the 2-party case to multi-party SFE: If in $\mathcal{F}$ there is a fixed setting of inputs for parties other than i, such that there exist two inputs for party i which induce different outputs for party $j \neq i$, then we say party i *influences* party j, and write $i \overset{\mathcal{F}}{\rightsquigarrow} j$.

Corollary 3.22. *If $\mathcal{F}$ is an m-party SFE functionality securely realizable using completely private channels, then in the directed graph induced by $\overset{\mathcal{F}}{\rightsquigarrow}$, either all edges have a common source, or all edges have a common destination.*

Proof. Suppose the graph induced by $\overset{\mathcal{F}}{\rightsquigarrow}$ has two edges ij and $i'j'$, where $i \neq i'$ and $j \neq j'$. Then any bipartition of $[m]$ which separates $\{i, j'\}$ and $\{i', j\}$ induces a 2-party functionality which has bidirectional influence. Thus $\mathcal{F}$ cannot be realizable. □

We see that there are only two simple kinds of securely realizable SFE functionalities. Let P be the common vertex in the graph induced by $\overset{\mathcal{F}}{\rightsquigarrow}$. If all edges are directed towards P, then we say that $\mathcal{F}$ is *aggregated via party P*. If all edges are directed away from P, then we say that $\mathcal{F}$ is *disseminated via party P*.

3-party characterization. The partitioning argument gives a necessary condition, but not generally a sufficient condition. However, we do identify one restricted setting in which the 2-party restrictions do provide a sufficient condition for secure realizability.

Our restricted setting is the case of 3-party *regular* (Definition 3.1) functionalities. That is, those functionalities whose behavior does not depend on which parties are corrupt, and which do not interact with the adversary. For these functionalities, we prove the following result:

Theorem 3.23. *Let $\mathcal{F}$ be a 3-party regular functionality that has an honest-majority protocol using $\mathcal{F}^*_{\text{PVT}}$. Then $\mathcal{F} \sqsubseteq \mathcal{F}^*_{\text{PVT}}$ if and only if all 2-party restrictions of $\mathcal{F}$ are UC-realizable using $\mathcal{F}^*_{\text{PVT}}$.*

Proof. The forward implication is trivially true. To show the other direction, assume each 2-party restriction is realizable (and therefore splittable according to the simplified definition Definition 3.3). Let $\mathcal{T}_i$ be the machine guaranteed by splittability on the 2-party restriction induced by the partition $\{i\}, (\{1, 2, 3\} \setminus \{i\})$. Let π be the honest-majority protocol for $\mathcal{F}$.

Now the protocol π' we construct is as follows: Party i internally simulates an instance of $\mathcal{F}$, $\mathcal{T}_i$, and of π (initialized for party i). The communication tapes of $\mathcal{F}$ and $\mathcal{T}_i$ are linked as in the splittability interaction, and the remaining communication tape of $\mathcal{F}$ (for honest party i) is treated as the protocol's environment-side communication tape. The environment-side communication tape of π and the remaining tape of $\mathcal{T}_i$ are linked, and the functionality-side tape of π is treated as the protocol's functionality-side communication tape.

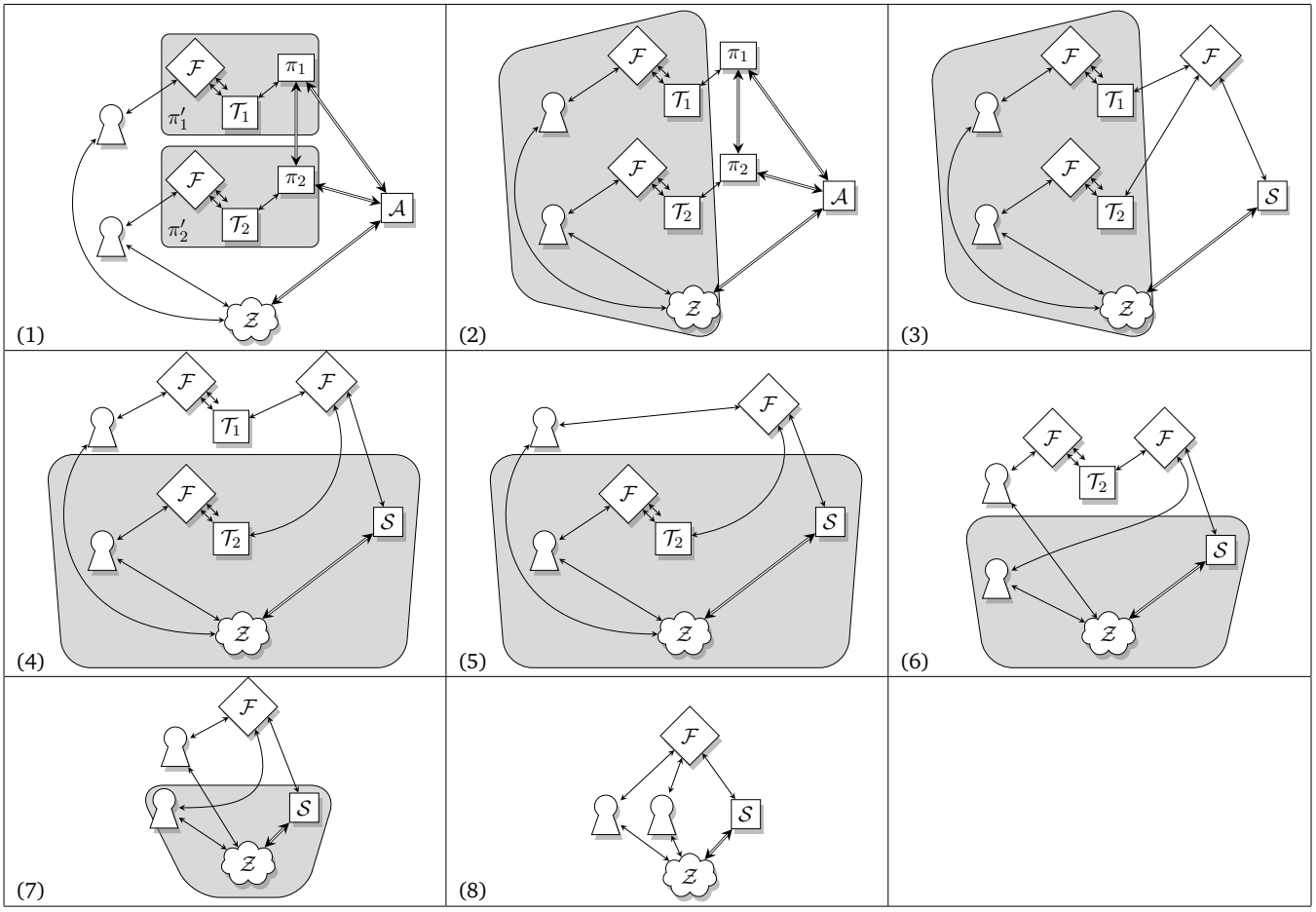

Figure 3.6: Steps in the proof of Theorem 3.23, when one party is corrupted. Communication via $\mathcal{F}_{\text{PVT}}^{*}$ is denoted by a double line.

To show that this protocol π' is secure, we must demonstrate a simulator for each dummy adversary. The cases when the adversary corrupts all or no parties are trivial. We focus on the case when the adversary corrupts 1 or 2 parties.

Suppose the adversary $\mathcal{A}$ corrupts 2 parties (by symmetry, parties 2 and 3). This adversary interacts on the private channel on behalf of parties 2 and 3, while the honest party 1 is running π'. However, in this instance of π', the honest party is faithfully running an instance of $\mathcal{F}$. Thus, the simulator can reflect a repackaging of the real-world interaction, simulating the instances of $\mathcal{T}_1$, π_1, and $\mathcal{F}_{\text{PVT}}^{*}$, as they are interacting in the real-world interaction. Being a simple repackaging, this simulation is perfectly indistinguishable from the real-world interaction.

Now suppose the adversary $\mathcal{A}$ corrupts only one party, by symmetry, party 3 (see Figure 3.6). In the real-world interaction, parties 1 and 2 are simulating independent instances of $\mathcal{F}$, as well as their respective $\mathcal{T}_i$ and π_i instances (box 1 in Figure 3.6). If we repackage this interaction so that everything but the π_i instances and $\mathcal{F}_{\text{PVT}}$ are inside the environment, we have an interaction in which two honest parties are running π on the channel $\mathcal{F}_{\text{PVT}}^{*}$ with an adversary (box 2 in Figure 3.6). This is an honest majority, and the security of π holds. Thus there is a simulator $\mathcal{S}$ for the adversary so that this interaction is indistinguishable from an ideal-world interaction with $\mathcal{F}$ itself (box 3 in Figure 3.6).

Next, we again repackage so that party 1's instance of $\mathcal{F}$ and $\mathcal{T}_1$ are outside of the environment, and $\mathcal{S}$ is inside the environment (box 4 in Figure 3.6). Since the behavior of $\mathcal{F}$ does not depend on which parties are corrupt, this interaction is an interaction with two instances of $\mathcal{F}$ being coordinated by an instance of $\mathcal{T}_1$. This is exactly a splittability interaction for the 2-party restriction of $\mathcal{F}$ induced by $\{1\}, \{2, 3\}$. Thus, this interaction is indistinguishable from the interaction with a single $\mathcal{F}$ alone (box 5 in Figure 3.6).

Finally, we can again repackage so that party 2's instance of $\mathcal{F}$ and $\mathcal{T}_2$ are outside of the environment (box 6 in Figure 3.6). Again, what remains is exactly a splittability interaction for another 2-party restriction of $\mathcal{F}$. Then the interaction is indistinguishable from one involving a single instance of $\mathcal{F}$ (box 7 in Figure 3.6). Then, repackaging to

remove $\mathcal{S}$ from the environment, we have an ideal-world interaction with $\mathcal{F}$ and simulator $\mathcal{S}$ (box 8 in Figure 3.6). By construction, this interaction is indistinguishable from the real-world interaction, so $\mathcal{S}$ is a suitable simulator for our protocol π'. □

Note that our protocol requires each player to indirectly simulate executions of another protocol with a weaker/different security guarantee (in this case, the 2-party restrictions and the honest-majority protocol). This approach is somewhat comparable to the "MPC in the head" approach recently introduced and explored in several works [55, 49]. There, significant efficiency gains are achieved in the standard corruption model by leveraging MPC protocols with security in the honest-majority settings. Our constructions indicate the possibility of extending this approach by having the parties carry out not a direct protocol execution, but a related simulation.

The same approach does not seem to apply for functionalities which interact with the adversary, whose behavior depends on which parties are corrupt, or which involve more than three parties (so that two parties do not form a strict majority).

3.5 Conclusion & Open Problems

We have given the first complete characterization of realizability in the UC framework for *arbitrary* functionalities and arbitrary "natural" communication channels. However, the characterization is still somewhat unwieldy in its full generality for multi-party functionalities.

Interpretations of splittability. We leave open the problem of applying splittability to provide more "simple" characterizations of realizability, as in our characterizations for 2-party non-reactive and SFE functionalities. In particular, we derived necessary conditions for multi-party SFE functionalities, but do not know whether they are sufficient. Our approach of using a partitioning argument to reduce a multi-party functionality to several 2-party functionalities has been studied in the passive security setting by Chor and Ishai [27]. In that setting, there exist m-party SFE functionalities which are not securely realizable, but all of whose k-party partitions (for $k < m$) are realizable. We leave open the question of whether such counterexamples exist in the context of UC security.

Another natural class of functionalities are reactive, deterministic finite-memory functionalities (DFF). We do not consider any broad characterizations for reactive functionalities in this chapter. However, later in Chapter 5, we develop some techniques for reasoning about DFFs. As a consequence of our results in that chapter, we implicitly derive a combinatorial characterization of secure realizability for DFFs (Section 5.3.3).

Non-natural communication channels. Splittability is a useful tool for understanding secure protocols on natural communication channels (those which are self-splittable). However, these channels are functionalities of relatively low cryptographic sophistication. Thus, in our landscape of cryptographic complexity, splittability can be used only to prove separations of the form $\mathcal{F} \not\sqsubseteq \mathcal{G}$ when $\mathcal{G}$ itself has low complexity. In particular, splittability is of no use in discriminating between the complexities of more sophisticated functionalities.

Later in Chapter 4, we develop some new techniques for deriving complexity separations between more sophisticated functionalities. However, these techniques appear to be tied to the particular subclass of functionalities that we consider, and are still not applicable to functionalities of arbitrary high complexity.

CHAPTER 4

Unbounded Cryptographic Complexity

4.1 Overview

In this chapter, we study 2-party symmetric-output SFE (SSFE) functionalities, in the computationally unbounded setting. Two-party SSFE is perhaps the most natural and widely studied subclass of MPC functionalities, dating back to the first MPC paper by Yao [96]. Beaver [4] and Kushilevitz [67] independently gave a combinatorial characterization of the SSFE functionalities that have *perfect* passive-secure protocols against unbounded adversaries. In other security settings (honest-majority and passive/standalone security against PPT adversaries), there exist secure protocols for all functionalities, and SSFE functionalities were among the first to be considered [97, 41, 9, 25]. Finally, Kilian [62] gave a combinatorial characterization of *completeness* for this class of functionalities against unbounded adversaries.

Yet, despite these fundamental results, our understanding of the complexity of securely realizing such functionalities remains far from complete, especially in the computationally unbounded setting. In this chapter, we present several new results to shed more light on the cryptographic complexity of SSFE functionalities in the unbounded setting. We provide the first complete characterizations of passive security[1] and of standalone security. We also show new impossibility results for concurrent self-composition, and identify a complex landscape of structures for SSFE functionalities under the $\sqsubseteq^u$ reduction. A visual overview of all of the cryptographic complexity we uncover is given in Figure 4.1.

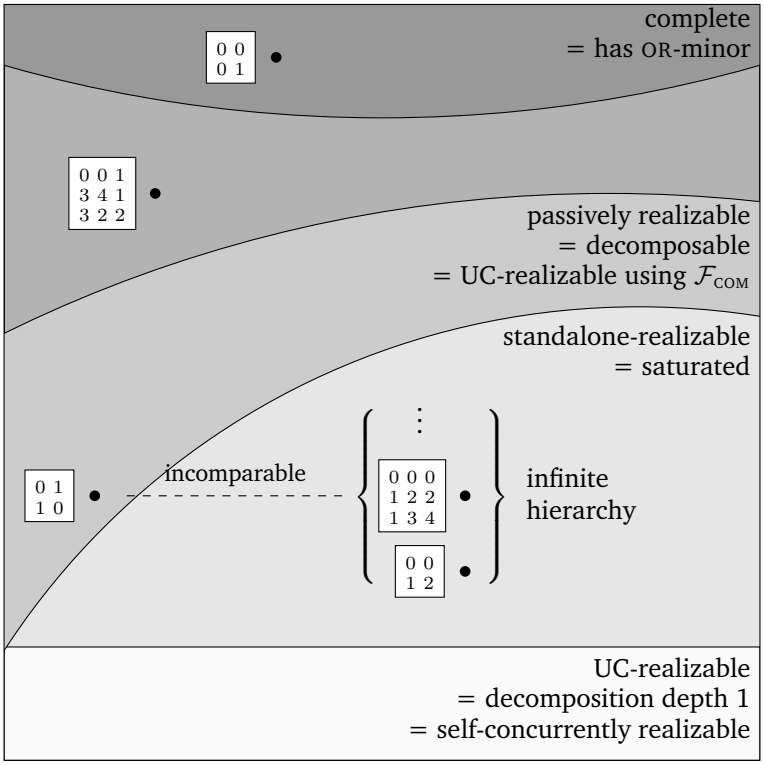

Figure 4.1: Cryptographic complexity landscape of 2-party SSFE, with several example functionalities.

[1] Although Beaver [4] and Kushilevitz [67] already have a characterization for *perfectly* secure protocols, ours is the first in the more natural model in which protocols are allowed to have negligible error.

4.1.1 Our Results

Leveraging passive security. The complete characterization of passive security by Beaver and Kushilevitz is purely combinatorial. They prove that a 2-party SSFE functionality $\mathcal{F}$ has a perfect, passive-secure protocol if and only if $\mathcal{F}$ satisfies a recursive combinatorial condition called *decomposability*. Decomposable functionalities have very simple *deterministic* secure protocols, whose round structure has a direct, natural correspondence with the recursive structure required by the definition of decomposition. We call these simple deterministic protocols *canonical protocols*.

Underpinning all of our results in this chapter is the intuition that these canonical protocols reflect the structure of *all* protocols for SSFE functionalities, in a wide variety of security settings that are much more demanding than passive security. Intuitively, canonical protocols so fundamentally express the structure of an SSFE functionality, that any protocol for that functionality in any setting will have to disclose information in essentially the same sequence. More formally, we prove the following technical theorem which unifies all of our further cryptographic complexity results:

Lemma. *Let $\mathcal{F}$ be a uniquely decomposable 2-party SSFE. Then $\mathcal{F}$ has a non-trivial protocol that is UC-secure (resp. standalone secure) if and only if the canonical protocol for $\mathcal{F}$ is UC-secure (resp. standalone secure).*

Thus, to show that $\mathcal{F}$ is *not* securely realizable, it suffices to show a single attack against the (very simple) canonical protocol for $\mathcal{F}$. Consequently, we are able to derive impossibility results which are simple and intuitively clear. The restriction that $\mathcal{F}$ be uniquely decomposable is necessary, since multiple decompositions can yield very different canonical protocols.

To prove Corollary 4.15, we show that if π is some purported protocol for $\mathcal{F}$, then for every adversary attacking the canonical protocol for $\mathcal{F}$, there is a corresponding adversary attacking π that achieves the same effect in all environments. The proof requires us to establish a correspondence between steps in an arbitrary protocol π and steps in the canonical protocol for $\mathcal{F}$ (equivalently, steps in the decomposition of $\mathcal{F}$), in terms of how much information about the parties' inputs has been revealed so far. As a direct consequence of the proof, we also show that canonical protocols have optimal round complexity among passive-secure protocols — even among randomized protocols with negligible error probability.

Alternative characterizations of realizability. We first use some of the techniques developed for Corollary 4.15 to extend the characterization of Beaver and Kushilevitz to the more natural model in which protocols can have negligible error probability. Thus we have the following:

Theorem. *Let $\mathcal{F}$ be a 2-party SSFE functionality. $\mathcal{F}$ has a (statistically secure) passive-secure protocol if and only if $\mathcal{F}$ is decomposable.*

We also show a surprising characterization of passive security as a natural statement in the language of UC security:

Theorem. *Let $\mathcal{F}$ be a 2-party SSFE functionality. Then $\mathcal{F}$ is passively realizable if and only if $\mathcal{F} \sqsubseteq^{u}_{nt} \mathcal{F}_{\text{COM}}$.*

Intuitively, the commitment functionality $\mathcal{F}_{\text{COM}}$ captures the essence of passive security. However, $\mathcal{F}_{\text{COM}}$ cannot be said to be *complete*, since it is not an SSFE functionality itself. To prove this characterization, we develop a general purpose compiler which converts any passive-secure SSFE protocol into a UC-secure protocol in the $\mathcal{F}_{\text{COM}}$-hybrid setting.

Next, we give the first complete characterization of standalone security for SSFE functionalities. We define an additional property of decomposable functionalities called *saturation*. This property completely characterizes standalone security:

Theorem. *Let $\mathcal{F}$ be a 2-party SSFE functionality. Then $\mathcal{F}$ is standalone-realizable if and only if $\mathcal{F}$ is saturated.*

Intuitively, a functionality is saturated if every traversal of the decomposition structure (equivalently, set of messages in the canonical protocol) corresponds to a legitimate input to the functionality. We define saturation explicitly as a special case of unique decomposability. As such, our technical results regarding canonical protocols apply, and we crucially use the connection to canonical protocols to prove the characterization.

Concurrent self-composition. Lindell [70] considered the *concurrent self-composition* setting, in which a protocol is analyzed in the context of many concurrent instances of the *same* protocol. He showed that, in the PPT setting, security under general (that is, with no limit on the number of protocol instances) concurrent self-composition is

equivalent to full-fledged UC security, for almost all functionalities. This impossibility result stands in contrast with the SSFE protocols of Lindell [69] and Pass and Rosen [80], which are secure for a *fixed* number of concurrent protocol instances in the PPT setting.

In the unbounded setting, however, Backes, Müller-Quade, and Unruh [3] demonstrated a protocol for an SSFE functionality which was standalone-secure, and even had a perfect, efficient (rewinding) simulator, but was not secure against two concurrent instances. In fact, their protocol is the canonical protocol for that functionality. Using our new techniques, we are able to extend and greatly generalize their attack, to obtain the following:

Theorem. *Let $\mathcal{F}$ be a 2-party SSFE functionality with unique decomposition. If $\mathcal{F}$ is not UC-realizable, then $\mathcal{F}$ has no protocol secure against even two concurrent instances.*

Thus concurrent attacks are an inherent property of the SSFE functionalities themselves, and are not particular to the specific protocol considered by Backes, Müller-Quade, and Unruh [3]. As is the case with all of the impossibility results in this chapter, to prove this theorem, we need only show an attack against two concurrent instances of the canonical protocol for $\mathcal{F}$.

Impossibility results in the UC framework. Finally, we apply our technical tools in the UC framework to derive cryptographic complexity separations with respect to the $\sqsubseteq^u_{nt}$ relation. We prove the following fundamental result identifying decomposition depth (i.e., the number of recursive steps in the decomposition of an SSFE functionality) as one indicator of cryptographic complexity:

Theorem. *If $\mathcal{F}$ is a 2-party SSFE functionality with unique decomposition depth m and $\mathcal{G}$ is a 2-party SSFE functionality with a (not necessarily unique) decomposition of depth $n < m$, then $\mathcal{F} \not\sqsubseteq^u_{nt} \mathcal{G}$.*

Since decomposition depth is equivalent to round complexity for passive security, another interpretation of this result is that round complexity against *passive* adversaries is an indicator of cryptographic complexity as measured in the *UC setting*.

Applying Corollary 4.26, we can easily identify the following interesting complexity structures uncovered by the $\sqsubseteq^u_{nt}$ reduction:

Theorem. *There exists a strict, infinite hierarchy of $\sqsubseteq^u_{nt}$-complexity; that is, functionalities $\mathcal{G}_1, \mathcal{G}_2, \ldots$ such that $\mathcal{G}_i \sqsubseteq^u_{nt} \mathcal{G}_j$ if and only if $i \leq j$.*

There also exist functionalities $\mathcal{F}$ and $\mathcal{G}$ whose complexities are incomparable; that is, $\mathcal{F} \not\sqsubseteq^u_{nt} \mathcal{G}$ and $\mathcal{G} \not\sqsubseteq^u_{nt} \mathcal{F}$.

We note that these are the first results differentiating the relative complexities of *intermediate* MPC functionalities (that is, functionalities which are neither trivial nor complete under the reduction in question).

Acknowledgement. Our characterizations of passive security (Theorem 4.17) and standalone security (Theorem 4.23) were independently discovered by Künzler, Müller-Quade, and Raub [66]. They also extend these results to a multi-party setting, and beyond the symmetric-output case.

4.2 SSFE Preliminaries

We review some relevant previous results classifying SSFE functionalities, mainly regarding passive security and completeness.

Definition 4.1 (Decomposable [4, 67])**.** *A 2-party SSFE functionality $\mathcal{F} : X \times Y \to D$ is* row decomposable *if there exists a partition $X = X_1 \cup \cdots \cup X_t$, with $t \geq 2$, such that the following hold for all $i \leq t$:*

- *$X_i \neq \emptyset$;*
- *for all $y \in Y$, $x \in X_i$, $x' \in (X \setminus X_i)$, we have $\mathcal{F}(x, y) \neq \mathcal{F}(x', y)$; and*
- *$\mathcal{F}\big|_{X_i \times Y}$ is either a constant function or column decomposable, where $\mathcal{F}\big|_{X_i \times Y}$ denotes the restriction of $\mathcal{F}$ to the domain $X_i \times Y$.*

We define being column decomposable *symmetrically with respect to X and Y. We say that $\mathcal{F}$ is simply* decomposable *if it is either constant, row decomposable, or column decomposable.*

For instance, the functionality $\boxed{\begin{smallmatrix}0&0\\1&2\end{smallmatrix}}$ is row-decomposable but not column-decomposable. $\boxed{\begin{smallmatrix}0&1\\1&0\end{smallmatrix}}$ is both row- and column-decomposable. Neither $\boxed{\begin{smallmatrix}0&0\\0&1\end{smallmatrix}}$ nor $\boxed{\begin{smallmatrix}0&0&1\\3&4&1\\3&2&2\end{smallmatrix}}$ are decomposable. Decomposability completely characterizes *perfect* passive security in the unbounded setting:

Theorem 4.2 ([4, 67])**.** *Let $\mathcal{F}$ be a 2-party SSFE functionality. Then $\mathcal{F}$ is decomposable if and only if it has a perfectly secure protocol against passive, unbounded adversaries.*

For convenience in our proofs, we have presented a slightly different definition of decomposability than that of [4, 67]. We insist that row and column decomposition steps strictly alternate, and thus each decomposition step must select the "most refined" partition possible. We say that a function $\mathcal{F}$ is *uniquely decomposable* if all of its decompositions are equivalent up to re-indexing the sets $X_1, \ldots, X_t$ (resp. $Y_1, \ldots, Y_t$) at each step. Thus $\boxed{\begin{smallmatrix}0&0\\1&2\end{smallmatrix}}$ is uniquely decomposable, but $\boxed{\begin{smallmatrix}0&1\\1&0\end{smallmatrix}}$ is not, since the first step may be either a row- or a column-decomposition step.

Canonical protocols [4, 67]. If $\mathcal{F}$ is decomposable, then a *canonical protocol* for $\mathcal{F}$ is a deterministic protocol defined inductively as follows:

- If $\mathcal{F}$ is a constant function, then both parties output the value, without interaction.
- If $\mathcal{F} : X \times Y \to Z$ is row decomposable as $X = X_1 \cup \cdots \cup X_t$, then Alice announces the unique i such that her input $x \in X_i$. Then both parties run a canonical protocol for $\mathcal{F}\big|_{X_i \times Y}$.
- If $\mathcal{F} : X \times Y \to Z$ is column decomposable as $Y = Y_1 \cup \cdots \cup Y_t$, then Bob announces the unique i such that his input $y \in Y_i$. Then both parties run a canonical protocol for $\mathcal{F}\big|_{X \times Y_i}$.

It is an easy exercise to see that a canonical protocol is a perfectly secure protocol for $\mathcal{F}$ against unbounded passive adversaries.

Deviation revealing. Prabhakaran and Rosulek [84] showed that for a class of functionalities called "deviation revealing" functionalities, if a protocol π is a UC-secure realization of that functionality, then that same protocol is also secure against passive adversaries. Note that this property is not true in general for all SFE functionalities. For example, the SFE in which Alice gets no output but Bob gets the boolean-OR of both parties' inputs is not passively realizable. However, the protocol where Alice simply sends her input to Bob is UC-secure, since a malicious Bob can always learn Alice's input in the ideal world by choosing 0 as its input to the functionality. It turns out that SSFE functions are deviation revealing. For completeness, we include here an adapted version of this result:

Lemma 4.3 ([84])**.** *Let π be a non-trivial UC-secure (perhaps in a hybrid world) or a standalone-secure protocol for a 2-party SSFE functionality $\mathcal{F}$. Then π is also passive-secure protocol for $\mathcal{F}$ as well (in the same hybrid setting as π).*

Proof. We show that, without loss of generality, the simulator for π maps passive real-world adversaries to passive ideal-world adversaries. A passive adversary $\mathcal{A}$ for π is one which receives an input x from the environment, runs π honestly, and reports its view to the environment. Note that in the context of SSFE functionalities, the relevant kinds of environments comprise a special class of standalone environments, so that even if π is only standalone secure, its security still holds with respect to the environments we consider for passive security.

Suppose $\mathcal{S}$ is the simulator for $\mathcal{A}$. In the ideal world, both parties produce output with overwhelming probability since π is a non-trivial protocol (and $\mathcal{A}$ is passive). Thus $\mathcal{S}$ must also allow the other party to generate output in the ideal world with overwhelming probability, meaning that $\mathcal{S}$ must receive x from the environment, send some x' to the ideal functionality $\mathcal{F}$, receive the output $\mathcal{F}(x', y)$ and deliver the output. Without loss of generality, we may assume $\mathcal{S}$ does so with probability 1.

Suppose x' is the input sent by $\mathcal{S}$ to $\mathcal{F}$. This input is selected independently of the environment's choice of input for the honest party. If $\mathcal{F}(x, y) \neq \mathcal{F}(x', y)$ for some input y, then consider an environment that uses y for the honest party's input. In this environment, the honest party will report $\mathcal{F}(x, y)$ in the real world, but $\mathcal{F}(x', y)$ in the ideal world, so the simulation is unsound. Thus with overwhelming probability, $\mathcal{S}$ sends an input x' such that $\mathcal{F}(x, \cdot) \equiv \mathcal{F}(x', \cdot)$. We may modify $\mathcal{S}$ by adding a simple wrapper which ensures that x (the input originally obtained from the environment) is always sent to $\mathcal{F}$. With overwhelming probability, the reply from $\mathcal{F}$ is unaffected by this change. Conditioned on these overwhelming probability events, the output of the wrapped $\mathcal{S}$ is identical to that of the original $\mathcal{S}$. However, the wrapped $\mathcal{S}$ is a *passive* ideal-world adversary: it receives x from the environment, sends x to $\mathcal{F}$, and delivers the output. □

Normal form for protocols. For simplicity in our proofs, we will often assume that a protocol is given in the following normal form:

1. If $m_1, m_2, \ldots$ are the messages exchanged in a run of the protocol, then $(m_1, m_2, \ldots)$ can be uniquely and unambiguously obtained from the string $m_1 m_2 \cdots$. This can be achieved without loss of generality by encoding all protocol messages in a prefix-free code.
2. As the last steps of the protocol, both parties announce their final output. It is easy to see that this is without loss of generality when the functionality's output is symmetric, even for standalone or UC security.
3. The honest protocol does not require the parties to maintain a persistent state besides their private input (in particular, no random tape). Instead, the protocol program for each party is simply a mapping $P : \{0,1\}^* \times \{0,1\}^* \times \{0,1\}^* \to [0,1]$, indicating that if τ is the transcript so far, and a party's input is x, then its next message is m with probability $P(x, \tau, m)$. In other words, randomness can be sampled as needed and immediately discarded. This requirement is without loss of generality for computationally unbounded parties, since at each step a party can (re)sample a random tape from the set of tapes consistent with their private input and transcript so far.

The consequence of this normal form is that it becomes easier to reason about parties who keep no internal state, and easier to reason about protocols in which the final output is a public function of the transcript (i.e., the output does not depend on a party's private state).

Completeness. We also review the known results characterizing completeness for SSFE functionalities.

Definition 4.4. *We say that $\{x, x'\} \times \{y, y'\}$ is an* OR-minor *in $\mathcal{F}$ if:*

$$\begin{array}{ccc} \mathcal{F}(x,y) & = & \mathcal{F}(x,y') \\ = & & \neq \\ \mathcal{F}(x',y) & \neq & \mathcal{F}(x',y') \end{array}$$

Lemma 4.5 ([65])**.** *Let $\mathcal{F}$ be a 2-party SSFE functionality. Then $\mathcal{F}$ is complete under the $\sqsubseteq^u_{nt}$ reduction if and only if $\mathcal{F}$ contains an* OR*-minor.*

In fact, Kraschewski and Müller-Quade [65] prove a more general theorem than stated above, in which the functionality need not have symmetric output. We use this more general result later in Chapter 5.

4.3 Simulation of Canonical Protocol in a General Protocol

In this section, we develop our main new technical tool, the protocol simulation theorem. Throughout the section we fix a 2-party SSFE functionality $\mathcal{F}$ with input domain $X \times Y$ and fix a passive-secure protocol π for $\mathcal{F}$, in normal form.

4.3.1 Structure of Protocols

We first introduce some terminology, notation, and simple technical observations relating to the structure of protocol transcripts.

Definition 4.6. *Let $\Pr[u|x,y]$ denote the probability that π generates a transcript that has u as a valid prefix (that is, u consists of a sequence of* complete *messages between parties), when executed honestly with x and y as inputs.*

Let F be a set of partial π-transcripts that is prefix-free.[2] *Define $\Pr[F|x,y] = \sum_{u \in F} \Pr[u|x,y]$. We call F a* frontier *if F is maximal – that is, if $\Pr[F|x,y] = 1$ for all x, y. We denote as $\mathcal{D}_F^{x,y}$ the probability distribution over F where $u \in F$ is chosen with probability $\Pr[u|x,y]$.*

Claim 4.7. *For any protocol π, there exist functions α and β such that $\Pr[u|x,y] = \alpha(u,x)\beta(u,y)$. In other words, the probability of a transcript being generated can be expressed as the product of two probabilities, each of which depends on only one party's input.*

[2] That is, no string in F is a proper prefix of another string in F.

Proof. The next message function depends only on the active party's input and the transcript so far. For a message m and partial transcript u, denote by $\pi_A(um, x)$ is the probability that Alice's next message is m when running on input x and the transcript so far is u. Define π_B for Bob analogously. Suppose we run π on inputs $(x, y) \in X \times Y$, where Alice sends the first message The probability of obtaining a particular partial transcript $u = u_1 \cdots u_n$ is:

$$\begin{aligned}\Pr[u|x,y] &= \pi_A(u_1, x) \cdot \pi_B(u_1u_2, y) \cdot \pi_A(u_1u_2u_3, x) \cdots \\ &= \left(\prod_{\substack{i=1 \\ i \text{ odd}}}^{n} \pi_A(u_1 \cdots u_i, x)\right)\left(\prod_{\substack{i=1 \\ i \text{ even}}}^{n} \pi_B(u_1 \cdots u_i, y)\right) = \alpha(u, x)\beta(u, y).\end{aligned}$$

where we define $\alpha(u, x)$ and $\beta(u, y)$ to be the parenthesized quantities, respectively. □

The following lemma is a convenient tool for proving bounds on statistical differences:

Lemma 4.8. *Let F be a frontier of partial transcripts in which Alice has just spoken, and let S be the set of partial transcripts in which Alice just spoken, which are proper prefixes of elements in F.*

Then there exist constants $c_\pi \geq 0$ and $\lambda(v, y, y') \geq 0$ for all $v \in S$ such that:

$$\sum_{u \in F} \alpha(u, x)|\beta(u, y) - \beta(u, y')| = c_\pi + \sum_{v \in S} \alpha(v, x)\lambda(v, y, y').$$

Proof. For clarity, assume that each message in the protocol is a single bit, and that the parties strictly alternate rounds. The proof also holds for unrestricted protocols, but is more cumbersome.

Let S_0 be the prefix-minimal elements of S (if Alice speaks first in the protocol, then $S_0 = \{0, 1\}$, otherwise $S_0 = \{\epsilon\}$). We will define a sequence $S_0 \subset S_1 \subset \cdots \subset S_t$ inductively as follows. Let $L(S_i)$ be the set of elements of S_i that have no extensions in S_i; that is, $L(S) = \{x \in S \mid (\forall y)\ xy \notin S\}$. We define a *pop* operation on elements of $L(S_i)$ as follows. When we pop an element $u \in L(S_i)$, we expand the partial transcript u by two moves in the protocol, setting $S_{i+1} = S_i \cup \{u00, u01, u10, u11\}$. After some finite number of operations we reach S_t such that $L(S_t) = F$.

We define the *weight* of a tree S_i as:

$$w(S_i) = \sum_{u \in L(S_i)} \alpha(u, x)|\beta(u, y) - \beta(u, y')|.$$

Observe that $w(S_t) = \Delta\left(\mathcal{D}_F^{x,y}, \mathcal{D}_F^{x,y'}\right)$. We define the quantity $c_\pi = w(S_0) \geq 0$. We define the intermediate quantity $\delta(u, y, y') = |\beta(u, y) - \beta(u, y')|$. Suppose the node v was popped in S_i to obtain S_{i+1}. Then,

$$\begin{aligned}w(S_{i+1}) - w(S_i) = {} & \alpha(v00, x)\delta(v00, y, y') + \alpha(v01, x)\delta(v01, y, y') \\ & + \alpha(v10, x)\delta(v10, y, y') + \alpha(v11, x)\delta(v11, y, y') \\ & - \alpha(v, x)\delta(v, y, y').\end{aligned}$$

Since $v0$ and $v1$ are nodes where Bob has just spoken, we get that $\delta(v00, y, y') = \delta(v01, y, y') = \delta(v0, y, y')$ and $\delta(v10, y, y') = \delta(v11, y, y') = \delta(v1, y, y')$. Similarly, $\alpha(v0, x) = \alpha(v1, x) = \alpha(v, x)$. By definition, we have $\alpha(v00, x) + \alpha(v01, x) = \alpha(v0, x)$ and $\alpha(v10, x) + \alpha(v11, x) = \alpha(v1, x)$. Using these observation we get:

$$\begin{aligned}w(S_{i+1}) - w(S_i) &= \left(\alpha(v00, x) + \alpha(v01, x)\right)\delta(v0, y, y') \\ &\quad + \left(\alpha(v10, x) + \alpha(v11, x)\right)\delta(v1, y, y') - \alpha(v, x)\delta(v, y, y') \\ &= \alpha(v0, x)\delta(v0, y, y') + \alpha(v1, x)\delta(v1, y, y') - \alpha(v, x)\delta(v, y, y') \\ &= \alpha(v, x)\left[\delta(v0, y, y') + \delta(v1, y, y') - \delta(v, y, y')\right].\end{aligned}$$

Define $\lambda(v, y, y') = \delta(v0, y, y') + \delta(v1, y, y') - \delta(v, y, y')$ and it is easy to verify that $\lambda(v, y, y') \geq 0$. The expression for $w(S_t)$ telescopes, and we obtain the desired result:

$$w(S_t) = c_\pi + \sum_{v \in S} \alpha(x, v)\lambda(v, y, y').$$

□

4.3.2 Associating Minors with Frontiers

Consider the 2×2 SSFE functionality $\left[\begin{smallmatrix} 0 & 0 \\ 1 & 2 \end{smallmatrix}\right]$, which is the smallest non-trivial (decomposition depth greater than one), uniquely decomposable SSFE. Observe that in the canonical protocol for this SSFE, Alice completely reveals her input before Bob reveals anything about his input. We now present a technical result that generalizes this observation about when information can be revealed, to *arbitrary* protocols.

Definition 4.9. *We say that $\{x, x'\} \times \{y, y'\}$ is a $\boxplus$-minor (resp. $\boxplus$-minor) in $\mathcal{F}$ if:*

$$\begin{array}{ccc} \mathcal{F}(x,y) & = & \mathcal{F}(x,y') \\ \neq & & \neq \\ \mathcal{F}(x',y) & \neq & \mathcal{F}(x',y') \end{array} \qquad \left(\begin{array}{ccc} & \mathcal{F}(x,y) & \neq \quad \mathcal{F}(x,y') \\ \textit{resp. if} & = & \neq \\ & \mathcal{F}(x',y) & \neq \quad \mathcal{F}(x',y') \end{array} \right)$$

We now show that in *any* secure protocol for *any* $\mathcal{F}$, the order in which parties disclose information about their inputs must respect *all* $\boxplus$- and $\boxplus$-minors within $\mathcal{F}$. That is, if $\{x, x'\} \times \{y, y'\}$ is a $\boxplus$-minor in $\mathcal{F}$, then there must be a point in the protocol at which Alice has (almost) *entirely* made the distinction between x and x', but Bob has not made *any* (noticeable) distinction between y and y'.

More formally,

Lemma 4.10 ($\boxplus$ Frontiers)**.** *For all $x \neq x' \in X$ and any parameter $\mu(k) < 1$, there is a frontier F such that, for all $y, y' \in Y$:*

- *if $\mathcal{F}(x,y) \neq \mathcal{F}(x',y)$, then $\Delta\left(\mathcal{D}_F^{x,y}, \mathcal{D}_F^{x',y}\right) \geq \mu(k)\left(1 - \frac{\epsilon(k)}{1-\mu(k)}\right)$, and*
- *$\Delta\left(\mathcal{D}_F^{x',y}, \mathcal{D}_F^{x',y'}\right) \leq \left(\frac{1+\mu(k)}{1-\mu(k)}\right) \Delta\left(\mathcal{D}_F^{x,y}, \mathcal{D}_F^{x,y'}\right)$.*

where $\epsilon(k)$ is the simulation error of π (i.e., the statistical difference between the real and ideal interactions).

Note that when $\{x, x'\} \times \{y, y'\}$ is a $\boxplus$-minor, we have $\Delta\left(\mathcal{D}_F^{x,y}, \mathcal{D}_F^{x,y'}\right) \leq \epsilon(k)$ for all frontiers F by the security of the scheme. By setting $\mu = 1 - \sqrt{\epsilon}$ and $\nu = 4\sqrt{\epsilon}$, we immediately obtain the following corollary:

Corollary 4.11 ($\boxplus$ Frontiers)**.** *If $\{x, x'\} \times \{y, y'\}$ is a $\boxplus$-minor in $\mathcal{F}$, then there exists a frontier F and a negligible function ν (which depend only on x and x') such that: $\Delta\left(\mathcal{D}_F^{x,y}, \mathcal{D}_F^{x',y}\right) \geq 1 - \nu(k)$, and $\Delta\left(\mathcal{D}_F^{x,y}, \mathcal{D}_F^{x,y'}\right), \Delta\left(\mathcal{D}_F^{x',y}, \mathcal{D}_F^{x',y'}\right) \leq \nu(k)$.*

In other words, at F, Alice has overwhelmingly made the distinction between inputs x and x', while Bob has not made any noticeable distinction between inputs y and y' (when Alice and Bob are executing the protocol on inputs in the $\boxplus$-minor).

Proof of Lemma 4.10. Suppose we have a protocol π in normal form, and let ϵ denote the simulation error of π. Let α and β be defined as in Claim 4.7.

Given parameters $x, x' \in X$ and $\mu < 1$, we define the frontier F as the *prefix-minimal* elements of:

$$\left\{ u \;\middle|\; u \text{ is a complete transcript, or } |\alpha(u,x) - \alpha(u,x')| \geq \mu(\alpha(u,x) + \alpha(u,x')) \right\}.$$

By "complete transcript", we mean one on which the parties terminate and give output. Intuitively, F is the first place at which x and x' induce significantly different probabilities on partial transcripts, where μ measures the significance. We extend F so that it becomes a complete frontier, by adding complete transcripts where necessary.

We partition F into $F_{\text{good}} \cup F_{\text{bad}}$, where F_{good} are the elements $u \in F$ which satisfy $|\alpha(u,x) - \alpha(u,x')| \geq \mu(\alpha(u,x) + \alpha(u,x'))$. Each element $u \in F_{\text{bad}}$ therefore satisfies $\alpha(u,x)/\alpha(u,x') < (1+\mu)/(1-\mu)$.

Consider any $y \in Y$ such that $\mathcal{F}(x,y) \neq \mathcal{F}(x',y)$. Let $A \subseteq F_{\text{bad}}$ be the transcripts in F_{bad} which do *not* induce output $\mathcal{F}(x,y)$ (all elements of F_{bad} are complete transcripts, and the output is a function of the transcript alone). Similarly let $B \subseteq F_{\text{bad}}$ be the transcripts in F_{bad} which do *not* induce output $\mathcal{F}(x',y)$. Thus $\Pr[A|x,y]$ and $\Pr[B|x',y]$

must each be at most ϵ. Observe that $F_{\text{bad}} \subseteq A \cup B$. Thus we have:

$$\begin{aligned}
\Pr[F_{\text{good}}|x,y] &= 1 - \Pr[F_{\text{bad}}|x,y] \\
&\geq 1 - \sum_{u\in A} \alpha(u,x)\beta(u,y) - \sum_{u\in B} \alpha(u,x)\beta(u,y) \\
&\geq 1 - \sum_{u\in A} \alpha(u,x)\beta(u,y) - \left(\frac{1+\mu}{1-\mu}\right) \sum_{u\in B} \alpha(u,x')\beta(u,y) \\
&= 1 - \Pr[A|x,y] - \left(\frac{1+\mu}{1-\mu}\right) \Pr[B|x',y] \\
&\geq 1 - \epsilon - \left(\frac{1+\mu}{1-\mu}\right)\epsilon = 1 - \frac{2\epsilon}{1-\mu}.
\end{aligned}$$

The same bound holds for $\Pr[F_{\text{good}}|x',y]$. Now,

$$\begin{aligned}
\Delta\left(\mathcal{D}_F^{x,y}, \mathcal{D}_F^{x',y}\right) &= \frac{1}{2}\sum_{u\in F} |\alpha(u,x) - \alpha(u,x')|\beta(u,y) \\
&\geq \frac{1}{2}\sum_{u\in F_{\text{good}}} \mu\left(\alpha(u,x) + \alpha(u,x')\right)\beta(u,y) \\
&= \frac{\mu}{2}\Big(\Pr[F_{\text{good}}|x,y] + \Pr[F_{\text{good}}|x',y]\Big) \\
&\geq \mu\left(1 - \frac{2\epsilon}{1-\mu}\right).
\end{aligned}$$

To show the other part of the lemma, consider any $y, y' \in Y$. Let S be the set of all proper prefixes of elements F that correspond to points where Alice has just spoken. By Lemma 4.8, we can express the statistical difference at F in terms of a positive linear combination of $\alpha(v,x)$ values for $v \in S$. Since by definition every $v \in S$ satisfies $\alpha(v,x')/\alpha(v,x) \leq (1+\mu)/(1-\mu)$, we have:

$$\begin{aligned}
\Delta\left(\mathcal{D}_F^{x',y}, \mathcal{D}_F^{x',y'}\right) &= \frac{1}{2}\sum_{u\in F} \alpha(u,x')|\beta(u,y) - \beta(u,y')| \\
&= \frac{1}{2}\left(c_\pi + \sum_{v\in S} \alpha(v,x')\lambda(v,y,y')\right) \\
&\leq \frac{1}{2}\left(c_\pi + \frac{1+\mu}{1-\mu}\sum_{v\in S} \alpha(v,x)\lambda(v,y,y')\right) \\
&\leq \frac{1}{2}\left(\frac{1+\mu}{1-\mu}\right)\left(c_\pi + \sum_{v\in S} \alpha(v,x)\lambda(v,y,y')\right) \\
&= \left(\frac{1+\mu}{1-\mu}\right)\Delta\left(\mathcal{D}_F^{x,y}, \mathcal{D}_F^{x,y'}\right).
\end{aligned}$$

□

4.3.3 Protocol Simulation Theorem

Our main protocol simulation theorem extends Lemma 4.10 to show that the information disclosed during any protocol must come in the same order as in the canonical protocol, provided that the canonical protocol is unique. This restriction on the canonical protocol is necessary, since different non-isomorphic canonical protocols for the same $\mathcal{F}$ can admit completely different kinds of attacks (e.g., for the XOR function, depending on which party speaks first).

Theorem 4.12 (Protocol Simulation)**.** *If $\mathcal{F}$ is uniquely decomposable, and π is a passive-secure protocol for $\mathcal{F}$ (in the unbounded setting), then the canonical protocol π_c for $\mathcal{F}$ is "as secure as" π, in the following sense: For all adversaries $\mathcal{A}$, there exists a simulator $\mathcal{S}$ such that for all environments $\mathcal{Z}$, we have* $\text{EXEC}[\mathcal{Z}, \mathcal{A}, \pi_c, \mathcal{F}_{\text{PVT}}] \approx \text{EXEC}[\mathcal{Z}, \mathcal{S}, \pi, \mathcal{F}_{\text{PVT}}]$.

In other words, every effective attack on the canonical protocol in the UC setting can be translated into an attack against π. Thus, if the canonical protocol for $\mathcal{F}$ is not secure against a given class of environments, then $\mathcal{F}$ is not

securely realizable (by any protocol) against that class of environments (see Corollary 4.15).

To prove the theorem, we first establish two intermediate lemmas:

Lemma 4.13. *For frontiers F and G, let "$G \ll F|x,y$" denote the event that when running the protocol on inputs x, y, the transcript reaches G strictly before reaching F. Then*

$$\Delta\Big(\mathcal{D}_F^{x,y}, \mathcal{D}_F^{x',y'}\Big) \leq \Delta\Big(\mathcal{D}_G^{x,y}, \mathcal{D}_G^{x',y'}\Big) + \frac{1}{2}\big(\Pr[G \ll F|x,y] + \Pr[G \ll F|x',y']\big).$$

Proof. Partition F into F_1 and F_2, where F_1 are the partial transcripts in F which are prefixes of elements in G. Thus the distribution of transcripts at frontier F_1 can be expressed as a function of the distribution of transcripts at frontier G. Then,

$$\begin{aligned}
\Delta\Big(\mathcal{D}_F^{x,y}, \mathcal{D}_F^{x',y'}\Big) &= \frac{1}{2}\sum_{u \in F_1 \cup F_2}\Big|\Pr[u|x,y] - \Pr[u|x',y']\Big| \\
&\leq \Delta\Big(\mathcal{D}_G^{x,y}, \mathcal{D}_G^{x',y'}\Big) + \frac{1}{2}\sum_{u \in F_2}\big(\Pr[u|x,y] + \Pr[u|x',y']\big) \\
&= \Delta\Big(\mathcal{D}_G^{x,y}, \mathcal{D}_G^{x',y'}\Big) + \frac{1}{2}\big(\Pr[F_2|x,y] + \Pr[F_2|x',y']\big) \\
&= \Delta\Big(\mathcal{D}_G^{x,y}, \mathcal{D}_G^{x',y'}\Big) + \frac{1}{2}\big(\Pr[G \ll F|x,y] + \Pr[G \ll F|x',y']\big).
\end{aligned}$$

This completes the proof. □

Lemma 4.14. *Let $\mathcal{F}$ be a uniquely decomposable SSFE functionality. Then for each step $X \times Y$ in the decomposition of $\mathcal{F}$ (by symmetry, suppose $X \times Y$ is row-decomposable as $X = X_1 \cup \cdots \cup X_n$), there exists a frontier $F(X,Y)$ and a negligible quantity λ such that for all $(x,y),(x',y') \in X \times Y$,*

$$\begin{aligned}
&\Delta\Big(\mathcal{D}^{x,y}_{F(X,Y)}, \mathcal{D}^{x',y'}_{F(X,Y)}\Big) \leq \lambda, && \text{if } x, x' \text{ belong to the same part } X_i \\
&\Delta\Big(\mathcal{D}^{x,y}_{F(X,Y)}, \mathcal{D}^{x',y'}_{F(X,Y)}\Big) \geq 1 - \lambda, && \text{otherwise.}
\end{aligned}$$

Proof. We prove the claim by induction on the decomposition depth d of $\mathcal{F}\big|_{X\times Y}$. When $d = 1$, the claim follows trivially by the security of the protocol and triangle inequality of statistical difference, setting $\lambda_d = 2\epsilon$.

Now, suppose $d \geq 2$, and suppose by symmetry that $\mathcal{F}\big|_{X\times Y}$ is row-decomposable as $X = X_1 \cup \cdots \cup X_n$. For each $i \leq n$, we have an associated frontier $F(X_i, Y)$ and negligible quantity λ_{d-1} by the inductive hypothesis. We will use this fact to construct a frontier $F(X,Y)$ with the desired properties.

Step 1: $F'(i,j)$. For every distinct $i, j \leq n$, we will construct a frontier $F'(i,j)$ with the following property: For all $(x,y),(x',y') \in (X_i \cup X_j) \times Y$,

- if $x, x' \in X_i$ or $x, x' \in X_j$ (i.e., x and x' are in the same part of the row-decomposition), then $\Delta\Big(\mathcal{D}^{x,y}_{F'(i,j)}, \mathcal{D}^{x',y'}_{F'(i,j)}\Big) \leq \gamma$;
- otherwise (x and x' are in different parts of the row-decomposition), $\Delta\Big(\mathcal{D}^{x,y}_{F'(i,j)}, \mathcal{D}^{x',y'}_{F'(i,j)}\Big) \geq 1 - \gamma$,

where γ is negligible in the security parameter.

If $\mathcal{F}\big|_{X_i\times Y}$ and $\mathcal{F}\big|_{X_j\times Y}$ are both constant functions, then $F'(i,j)$ is simply the set of complete transcripts. By the security of the protocol, the above claim is satisfied with $\gamma = 2\epsilon$.

Otherwise, our construction is symmetric with respect to i and j, so assume that $\mathcal{F}\big|_{X_i\times Y}$ is column-decomposable as $Y = Y_1 \cup \cdots \cup Y_m$. Then $\mathcal{F}\big|_{X_j\times Y}$ is either column-decomposable as $Y = Y'_1 \cup \cdots \cup Y'_{m'}$; or it is constant, in which case we define $m' = 1$ and $Y'_1 = Y$ for simplicity. Furthermore, we assume without loss of generality that the sets are indexed so that $Y_1 \cap Y'_1 \neq \emptyset$.

We will construct $F'(i,j)$ and prove the above properties by induction. More specifically, we will show by induction on $\ell + \ell'$ that:

- For all $(x,y),(x',y') \in X_i \times (Y_1 \cup \cdots \cup Y_\ell)$, we have $\Delta\Big(\mathcal{D}^{x,y}_{F'(i,j)}, \mathcal{D}^{x',y'}_{F'(i,j)}\Big) \leq \gamma_{\ell+\ell'}$.

- For all $(x, y), (x', y') \in X_j \times (Y'_1 \cup \cdots \cup Y'_{\ell'})$, we have $\Delta\left(\mathcal{D}^{x,y}_{F'(i,j)}, \mathcal{D}^{x',y'}_{F'(i,j)}\right) \leq \gamma_{\ell+\ell'}$.

The remaining condition to be shown will then follow easily.

To prove the base case, we first observe that since $\mathcal{F}$ is uniquely decomposable, $Y = (Y_1 \cap Y'_1) \cup (Y \setminus (Y_1 \cap Y'_1))$ is not a valid decomposition of *both* $\mathcal{F}\big|_{X_i \times Y}$ and $\mathcal{F}\big|_{X_j \times Y}$. By symmetry, assume that it is not a valid decomposition of $\mathcal{F}\big|_{X_i \times Y}$; thus there exists $x \in X_i$, $y \in Y_1 \cap Y'_1$, and $y' \notin Y_1 \cap Y'_1$, such that $\mathcal{F}(x, y) = \mathcal{F}(x, y')$. By the properties of decompositions, this equality implies that $y' \in Y_1$; therefore $y' \notin Y'_1$. Without loss of generality, assume that the sets are indexed so that $y' \in Y'_2$.

Choose $x' \in X_j$ arbitrarily; then $\{x, x'\} \times \{y, y'\}$ is a $\boxplus$-minor. We will define $F'(i, j)$ as the frontier given by Lemma 4.10 for inputs x, x' and parameter μ to be fixed later. By the properties of this frontier, we then have

$$\Delta\left(\mathcal{D}^{x',y}_{F'(i,j)}, \mathcal{D}^{x',y'}_{F'(i,j)}\right) \leq \left(\frac{1+\mu}{1-\mu}\right)\epsilon.$$

Our choice of μ will ensure that this statistical difference is negligible. However, by the inductive hypothesis, these two input pairs (one in $X_j \times Y'_1$ and one in $X_j \times Y'_2$) induce overwhelmingly different distributions at $F(X_j, Y)$; the distributions have statistical difference at least $1 - \lambda_{d-1}$. Thus (intuitively) $F'(i, j)$ must be encountered before $F(X_j, Y)$, with high probability. More formally, by Lemma 4.13, we have:

$$\Pr[F'(i,j) \ll F(X_j, Y)|x', y] \geq 1 - 2\left(\lambda_{d-1} + \left(\frac{1+\mu}{1-\mu}\right)\epsilon\right);$$
$$\Pr[F'(i,j) \ll F(X_j, Y)|x', y'] \geq 1 - 2\left(\lambda_{d-1} + \left(\frac{1+\mu}{1-\mu}\right)\epsilon\right).$$

The event "$F'(i, j) \ll F(X_j, Y)$" can be expressed naturally as a statistical test on the distribution of transcripts at frontier $F(X_j, Y)$. For any $(x^*, y^*) \in X_j \times Y'_1$, we have that $\Delta\left(\mathcal{D}^{x',y}_{F(X_j,Y)}, \mathcal{D}^{x^*,y^*}_{F(X_j,Y)}\right) \leq \lambda_{d-1}$ by the inductive hypothesis. Thus the probability of the event "$F'(i, j) \ll F(X_j, Y)$" can differ by no more than λ_{d-1} when executing the protocol on inputs (x', y) versus inputs (x^*, y^*). Thus, we have:

$$\Pr[F'(i,j) \ll F(X_j, Y)|x^*, y^*] \geq 1 - 3\lambda_{d-1} - 2\left(\frac{1+\mu}{1-\mu}\right)\epsilon,$$

for all $(x^*, y^*) \in X_j \times Y'_1$.

Then, intuitively, since $F'(i, j)$ is almost always encountered before $F(X_j, Y)$, for all inputs in $X_j \times Y'_1$, and all of these inputs induce negligibly close distributions at $F(X_j, Y)$, then all of these inputs must also induce negligibly close distributions at $F'(i, j)$ as well. More formally, we have the following bound for all $(x_0, y_0), (x_1, y_1) \in X_j \times Y'_1$:

$$\begin{aligned}
\Delta\left(\mathcal{D}^{x_0,y_0}_{F'(i,j)}, \mathcal{D}^{x_1,y_1}_{F'(i,j)}\right) &= \frac{1}{2} \sum_{\substack{u \in F'(i,j) \\ F'(i,j) \ll F(X_j,Y)}} \Big|\Pr[u|x_0, y_0] - \Pr[u|x_1, y_1]\Big| \\
&\qquad + \frac{1}{2} \sum_{\substack{u \in F'(i,j) \\ F'(i,j) \not\ll F(X_j,Y)}} \Big|\Pr[u|x_0, y_0] - \Pr[u|x_1, y_1]\Big| \\
&\leq \Delta\left(\mathcal{D}^{x_0,y_0}_{F(X_j,Y)}, \mathcal{D}^{x_1,y_1}_{F(X_j,Y)}\right) \\
&\qquad + \frac{1}{2} \sum_{\substack{u \in F'(i,j) \\ F'(i,j) \not\ll F(X_j,Y)}} \Big(\Pr[u|x_0, y_0] + \Pr[u|x_1, y_1]\Big) \\
&= \Delta\left(\mathcal{D}^{x_0,y_0}_{F(X_j,Y)}, \mathcal{D}^{x_1,y_1}_{F(X_j,Y)}\right) \\
&\qquad + \frac{1}{2}\Pr[F'(i,j) \not\ll F(X_j, Y)|x_0, y_0] \\
&\qquad + \frac{1}{2}\Pr[F'(i,j) \not\ll F(X_j, Y)|x_1, y_1] \\
&\leq 4\lambda_{d-1} + 2\left(\frac{1+\mu}{1-\mu}\right)\epsilon.
\end{aligned}$$

We can apply the same reasoning for $X_i \times Y_1$ and $X_j \times Y_2'$; for each of these domains of $\mathcal{F}$, its input pairs all induce pairwise negligibly close distributions at the frontier $F'(i,j)$.

However, we have also shown above that, due to the $\boxplus$-minor $\{x,x'\} \times \{y,y'\}$, some particular input pair in $X_j \times Y_1'$ induces a negligibly close (within $\left(\frac{1+\mu}{1-\mu}\right)\epsilon$) distribution to an input pair in $X_j \times Y_2'$, at the frontier $F'(i,j)$. Thus by the triangle inequality, we have that

$$\Delta\left(\mathcal{D}^{x_0,y_0}_{F'(i,j)}, \mathcal{D}^{x_1,y_1}_{F'(i,j)}\right) \leq 8\lambda_{d-1} + 5\left(\frac{1+\mu}{1-\mu}\right)\epsilon,$$

for all $(x_0,y_0),(x_1,y_1) \in X_j \times (Y_1' \cup Y_2')$.

This establishes the inductive claim for $\ell = 1$, $\ell' = 2$, with $\gamma_3 = 8\lambda_{d-1} + 5\left(\frac{1+\mu}{1-\mu}\right)\epsilon$.

To complete the inductive step, we use a very similar argument as in the base case. Set $Y^* = (Y_1 \cup \cdots \cup Y_\ell) \cap (Y_1' \cup \cdots \cup Y_{\ell'}')$, and observe that (unless $Y^* = Y$, in which case we are done) $Y = Y^* \cup (Y \setminus Y^*)$ is not a valid column-decomposition of both $\mathcal{F}\big|_{X_i \times Y}$ and $\mathcal{F}\big|_{X_j \times Y}$. By symmetry, suppose it is not a column-decomposition of $\mathcal{F}\big|_{X_i \times Y}$. As above, there exists $x_0 \in X_i$, $y \in Y^*$, $y' \in Y \setminus Y^*$ such that $\mathcal{F}(x_0,y) = \mathcal{F}(x_0,y')$. Then $y' \in Y_1 \cup \cdots \cup Y_\ell$, and $y' \notin Y_1' \cup \cdots \cup Y_{\ell'}'$. Assume that the sets are indexed so that $y' \in Y_{\ell'+1}'$.

Let $x \in X_i$ and $x' \in X_j$ be the values used in the definition of $F'(i,j)$. By the inductive hypothesis, $\Delta\left(\mathcal{D}^{x,y}_{F'(i,j)}, \mathcal{D}^{x,y'}_{F'(i,j)}\right) \leq \gamma_{\ell+\ell'}$, so by the construction of $F'(i,j)$, we have $\Delta\left(\mathcal{D}^{x',y}_{F'(i,j)}, \mathcal{D}^{x',y'}_{F'(i,j)}\right) \leq \left(\frac{1+\mu}{1-\mu}\right)\gamma_{\ell+\ell'}$.

Then applying the same argument as before, (since $(x',y') \in X_j \times Y_{\ell'+1}'$) we can show:

$$\Delta\left(\mathcal{D}^{x_0,y_0}_{F'(i,j)}, \mathcal{D}^{x_1,y_1}_{F'(i,j)}\right) \leq 8\lambda_{d-1} + 5\left(\frac{1+\mu}{1-\mu}\right)\gamma_{\ell+\ell'},$$

for all $(x_0,y_0),(x_1,y_1) \in X_j \times (Y_1' \cup \cdots \cup Y_{\ell'+1}')$.

This completes the inductive claim for $(\ell, \ell'+1)$, setting $\gamma_{\ell+\ell'+1} = 8\lambda_{d-1} + 5\left(\frac{1+\mu}{1-\mu}\right)\gamma_{\ell+\ell'}$.

Finally, if we have established the inductive claim, then it is easy to establish the second desired condition of $F'(i,j)$. The definition of $F'(i,j)$ implies that for any $y \in Y$, $x \in X_i$, and $x' \in X_j$, we have $\Delta\left(\mathcal{D}^{x,y}_{F'(i,j)}, \mathcal{D}^{x',y}_{F'(i,j)}\right) \geq \mu\left(1 - \frac{2\epsilon}{1-\mu}\right)$. So by the triangle inequality, we have:

$$\Delta\left(\mathcal{D}^{x_0,y_0}_{F'(i,j)}, \mathcal{D}^{x_1,y_1}_{F'(i,j)}\right) \geq \mu\left(1 - \frac{2\epsilon}{1-\mu}\right) - \gamma_{m+m'},$$

for any $x_0 \in X_i$, $x_1 \in X_j$, and $y_0, y_1 \in Y$.

It finally suffices to set μ appropriately. First observe that solving the recurrence γ_n yields that $\gamma_n = O\left(\left(5\left(\frac{1+\mu}{1-\mu}\right)\right)^{n-1}(\lambda_{d-1}+\epsilon)\right)$. Then setting $\mu = 1 - (\lambda_{d-1}+\epsilon)^{1/(m+m')}$, we have that $\gamma_{m+m'} = O(10^{m+m'}(\lambda_{d-1} + \epsilon)^{1/(m+m')})$, which is negligible when λ_{d-1} and ϵ are negligible, and m, m' are constant. Then all the desired bounds are negligible, with negligible quantity $\gamma = O(\gamma_{m+m'})$. △

Step 2: $R(i)$. Next, for each $i \leq n$, we define a set $R(i)$ of transcript prefixes, as follows. First, for each X_i, we arbitrarily choose a representative $x_i \in X_i$. Define the following two sets:

$$L(i) = \{u \mid (\forall j \neq i) \text{ some prefix of } u \text{ appears in } F'(i,j)\};$$
$$R(i) = \left\{u \in L(i) \;\middle|\; \alpha(u,x_i) > \max_{j \neq i} \alpha(u,x_j)\right\}.$$

Thus $L(i)$ is a frontier, consisting of the partial transcripts which have reached every $F'(i,j)$ frontier. Suppose $(x,y),(x',y') \in X_i \times Y$. Then we have:

$$\begin{aligned}\Delta\left(\mathcal{D}^{x,y}_{L(i)}, \mathcal{D}^{x',y'}_{L(i)}\right) &\leq \frac{1}{2}\sum_{j \neq i}\sum_{u \in L(i) \cap F'(i,j)} \left|\Pr[u|x,y] - \Pr[u|x',y']\right| \\ &\leq \sum_{j \neq i} \Delta\left(\mathcal{D}^{x,y}_{F'(i,j)}, \mathcal{D}^{x',y'}_{F'(i,j)}\right) \leq n\gamma.\end{aligned}$$

Note that the distribution of partial transcripts at any $F'(i,j)$ can be naturally expressed as a function of the distribution of transcripts at $L(i)$. Thus for any input pairs (x_i, y) with $x_i \in X_i$, and (x_j, y) with $x_j \in X_j$, we have:

$$\Delta\Big(\mathcal{D}^{x_i,y}_{L(i)}, \mathcal{D}^{x_j,y}_{L(i)}\Big) \geq \Delta\Big(\mathcal{D}^{x_i,y}_{F'(i,j)}, \mathcal{D}^{x_j,y}_{F'(i,j)}\Big) \geq 1 - \gamma.$$

Note that the optimal statistical test for distinguishing $\mathcal{D}^{x_i,y}_{L(i)}$ and $\mathcal{D}^{x_j,y}_{L(i)}$ is to output (x_i, y) on input u if $\alpha(u, x_i) < \alpha(u, x_j)$ and output (x_j, y) otherwise. Since this statistical test is optimal and the statistical difference is at least $1 - \gamma$, the probability of reaching a transcript $u \in L(i)$ such that $\alpha(u, x_i) \leq \alpha(u, x_j)$ when running the protocol on input x_i is at most γ. This observation motivates the definition of $R(i)$, and for all $y \in Y$, we have:

$$\Pr[\neg R(i)|x_i, y] \leq \sum_{j \neq i} \sum_{\substack{u \in L(i) \\ \alpha(u,x_i) \leq \alpha(u,x_j)}} \Pr[u|x_i, y] \leq \sum_{j \neq i} \gamma = (n-1)\gamma.$$

Then it follows that $\Pr[\neg R(i)|x, y] \leq 2n\gamma$ for all $(x, y) \in X_i \times Y$, since $\Delta\Big(\mathcal{D}^{x,y}_{L(i)}, \mathcal{D}^{x_i,y}_{L(i)}\Big) \leq n\gamma$. △

Step 3: The final construction. Note that for distinct i and j, the set $R(i) \cup R(j)$ is prefix-free by definition. We finally define $F(X, Y)$ as the partial transcripts of $\bigcup_i R(i)$, along with any complete transcripts needed to extend $F(X, Y)$ to a frontier.

Let $(x, y), (x', y') \in X_i \times Y$. Then we have:

$$\begin{aligned}
&\Delta\Big(\mathcal{D}^{x,y}_{F(X,Y)}, \mathcal{D}^{x',y'}_{F(X,Y)}\Big) \\
&\quad = \frac{1}{2} \sum_{u \in F(X,Y) \cap R(i)} \Big|\Pr[u|x, y] - \Pr[u|x', y']\Big| \\
&\qquad\qquad + \frac{1}{2} \sum_{u \in F(X,Y) \setminus R(i)} \Big|\Pr[u|x, y] - \Pr[u|x', y']\Big| \\
&\quad \leq \Delta\Big(\mathcal{D}^{x,y}_{L(i)}, \mathcal{D}^{x',y'}_{L(i)}\Big) + \frac{1}{2} \sum_{u \in F(X,Y) \setminus R(i)} \Big(\Pr[u|x, y] + \Pr[u|x', y']\Big) \\
&\quad \leq \Delta\Big(\mathcal{D}^{x,y}_{L(i)}, \mathcal{D}^{x',y'}_{L(i)}\Big) + \frac{1}{2}\Big(\Pr[\neg R(i)|x, y] + \Pr[\neg R(i)|x', y']\Big) \\
&\quad \leq 3n\gamma.
\end{aligned}$$

This establishes half of the desired claim.

Now let $(x, y) \in X_i \times Y$ and $(x', y') \in X_j \times Y$, where $i \neq j$. Consider the statistical test on partial transcripts in $F(X, Y)$ in which we output (x, y) on input u if $\alpha(u, x_i) > \alpha(u, x_j)$, and output (x', y') otherwise. The probability that this statistical test gives an incorrect answer is at most $\Pr[\neg R(i)|x, y] + \Pr[\neg R(j)|x', y'] = 4n\gamma$. Thus $\Delta\Big(\mathcal{D}^{x,y}_{F(X,Y)}, \mathcal{D}^{x',y'}_{F(X,Y)}\Big) \geq 1 - 4n\gamma$. Finally, setting $\lambda = 4n\gamma$ completes the proof. □

Completing the proof. Finally, the proof of Theorem 4.12 follows naturally from the previous lemma:

Proof of Theorem 4.12. Given such frontiers corresponding to each decomposition step, the required simulation is natural. Suppose by symmetry that the adversary $\mathcal{A}$ corrupts Alice alone. The simulator's job is to simulate the canonical protocol to $\mathcal{A}$, while interacting with the honest Bob in the protocol π. The simulator $\mathcal{S}$ does the following:

1. Maintain variables X, Y, and $x \in X$. X and Y are initialized to the domain of function $\mathcal{F}$, and x is initialized to an arbitrary element of X.
2. For each step in the canonical protocol for $\mathcal{F}$, do:
   (a) If this is a step in which Alice speaks, then receive a message in the canonical protocol from $\mathcal{A}$. The message determines a subset $X' \subseteq X$ upon which to recurse in the canonical protocol. Update $X \leftarrow X'$, and if $x \notin X'$, choose an arbitrary $x' \in X'$ and update $x \leftarrow x'$.
   (b) If this is a step in which Bob speaks, then run the π protocol honestly on input x, with the external honest Bob. Run until the partial transcript reaches the frontier in π corresponding to Bob's next decomposition step. If the frontier for a different Bob decomposition step is encountered first, then abort.

Otherwise, determine the subdomain $Y' \subseteq Y$ for this decomposition step such that Bob's input lies in Y'; by the construction of the frontiers, there is an accurate statistical test to determine this subdomain with overwhelming probability. Update $Y \leftarrow Y'$, and simulate to $\mathcal{A}$ the corresponding message (Y') in the canonical protocol.

3. After the last step of the canonical protocol, continue honestly running the protocol π on the current value of input x, with honest Bob.

We note that the simulator $\mathcal{S}$ may change its input x several times during its (otherwise honest) execution of the π protocol. Here it is important that π be in normal form, so that the next-message function depends only on the input and the transcript generated so far (in particular, not on a long-term private random tape).

Without loss of generality since the canonical protocol is deterministic, suppose that $\mathcal{A}$ is deterministic. For $y \in Y$, define $\mathcal{A}(y) \in X$ to be an input $x \in X$ such that if $\mathcal{A}$ interacts with an honest Bob who runs the (deterministic) canonical protocol on input y, then $\mathcal{A}$ will produce messages in the canonical protocol consistent with having input $x \in X$. To show the soundness of the simulation, it suffices to show that whenever $\mathcal{S}$ is interacting with an honest Bob with input y in protocol π, the π transcript and the simulated canonical protocol transcript are indistinguishable from respective transcripts honestly generated on inputs $\mathcal{A}(y)$ and y. In particular, this implies that the output of Bob will be indistinguishable in the two settings, by the correctness of π and the canonical protocol.

We prove the correctness of the simulation via a sequence of hybrid interactions. Suppose the maximum number of Alice-rounds in the canonical protocol is n, and let y be the input upon which the honest Bob is executing the protocol π. For $i \in \{0, \ldots, n\}$, let x_i be the value of variable x after i times through step (2a). Thus in the ith time through step (2b), the simulator will honestly run protocol π on input x_i, and $x_n = \mathcal{A}(y)$.

We define Hybrid i, for $i \in \{0, \ldots, n\}$, to be identical to the simulation, except that in the first i times through step (2b), we set the variable $x \leftarrow x_i$; thereafter the simulator updates variable x as normal. Thus Hybrid 0 is exactly the simulation. In Hybrid n, the simulator runs π honestly on input $\mathcal{A}(y)$ the entire time, and generates a simulated canonical protocol transcript indistinguishable from one generated with Bob's true input y, as desired. The latter condition is because by the correctness of the frontiers, it is only with negligible probability that in step (2b), the simulator will abort or incorrectly determine Y' such that Bob's input $y \in Y'$.

Finally, we must show the indistinguishability of consecutive hybrids. The difference between Hybrids i and $i+1$ is that in the first i times through step (2b), the simulator honestly runs the π protocol with input x_i instead of x_{i+1}. By construction, both x_i and x_{i+1} induce the same messages in the first i rounds of the canonical protocol, when Bob has input y. Thus by the correctness of the frontiers, these two inputs induce statistically indistinguishable partial transcripts in the first i times through step (2b). The remainder of the interaction depends only on the partial transcript; in particular, not on the past values of the x variable. Thus the entire interaction is indistinguishable between the two hybrids, as desired. □

Implications for other security settings. Theorem 4.12 proves an equivalence between attacks against the canonical protocol and against any other protocol, *in the UC sense*. Consequently, the theorem also has implications for other security settings as well:

Corollary 4.15. *Let $\mathcal{F}$ be a uniquely decomposable 2-party SSFE. Then $\mathcal{F}$ has a non-trivial protocol that is UC-secure against a class of environments $\mathcal{C}$ if and only if the canonical protocol for $\mathcal{F}$ is such a protocol.*

Proof. One direction ($\Leftarrow$) of the claim is immediate. For the other direction, suppose π is such a secure protocol for $\mathcal{F}$, but the canonical protocol is not. By the properties of $\mathcal{F}$, we have that π is also a passive-secure protocol for $\mathcal{F}$. As such, Theorem 4.12 holds. There is an adversary attacking the canonical protocol in some environment in the class $\mathcal{C}$ which violates the security of $\mathcal{F}$. Theorem 4.12 implies that there is a corresponding adversary attacking π, in the same environment, which also violates the security of $\mathcal{F}$. This contradicts the supposed security of π. □

In particular, standalone security and concurrent-self-composable security are both defined as UC security for a restricted class of environments; and Corollary 4.15 applies to those security settings (as well as to full UC security, of course).

4.3.4 Round Complexity

Let $R(\pi, x, y)$ denote the random variable indicating the number of rounds taken by π on inputs x, y, and let $R(\mathcal{F}, x, y)$ denote the same quantity with respect to the canonical protocol for $\mathcal{F}$ (which is deterministic).

Corollary 4.16. *Let $\mathcal{F}$ be a 2-party SSFE functionality. If $\mathcal{F}$ is uniquely decomposable, then its canonical protocol achieves the optimal round complexity. That is, for every secure protocol π for $\mathcal{F}$, we have $\mathbb{E}[R(\pi, x, y)] \geq R(\mathcal{F}, x, y) - \nu$, where ν is a negligible function in the security parameter of π.*

Proof. The proof of Theorem 4.12 constructs for π a frontier for each step in the decomposition of $\mathcal{F}$ (corresponding to each step in the canonical protocol). By the required properties of the frontiers, the transcript for π must visit all the relevant frontiers in order, one *strictly* after the next, with overwhelming probability. □

4.4 Characterizing Passive Security

In this section, we apply Lemma 4.10 to extend the characterization of Beaver [4] and Kushilevitz [67] to the case of statistical security. We also show a new characterization of passive security in terms of the ideal commitment functionality.

Theorem 4.17. *Let $\mathcal{F}$ be a 2-party SSFE functionality. Then $\mathcal{F}$ has a (statistically secure) passive-secure protocol if and only if $\mathcal{F}$ is decomposable.*

Proof. ($\Leftarrow$) Trivial by the (perfect) security of canonical protocols.

($\Rightarrow$) Suppose $\mathcal{F}$ has a passive-secure protocol π. Then it cannot have an OR-minor, since such functions are complete for passive security [63]. One can easily verify that if $\mathcal{F}$ does not have both a $\boxplus$- and $\boxplus$-minor, then it is decomposable (in fact, $\mathcal{F}$ is isomorphic to the function $\mathcal{F}(x, y) = x$), and the claim is proven. Otherwise, suppose for contradiction that $\mathcal{F}$ is not decomposable. If one decomposition step can be made in $\mathcal{F}$, say, a partition of X into $X_1 \cup X_2$ such that for all $y \in Y$, $x \in X_1$, and $x' \in X_2$, $\mathcal{F}(x, y) \neq \mathcal{F}(x', y)$, then at least one of $\mathcal{F}\big|_{X_1 \times Y}$ and $\mathcal{F}\big|_{X_2 \times Y}$ are also not decomposable. The corresponding condition also holds for column decomposition steps as well. The protocol in which parties run π, but restricted their inputs to a subdomain of $X' \times Y' \subseteq X \times Y$, is also a passive-secure protocol for $\mathcal{F}\big|_{X' \times Y'}$. Thus without loss of generality, we assume that $\mathcal{F}$ is such that *no* single decomposition step can be made in $\mathcal{F}$.

Intuitively, by Lemma 4.10, each $\boxplus$-minor $\{x, x'\} \times \{y, y'\}$ corresponds to a combination of inputs where Alice must differentiate her input before Bob. Similarly, for a $\boxplus$-minor, Bob must differentiate first. We will leverage these constraints to obtain a contradiction to the assumption that π is a passively secure protocol.

For every $\boxplus$-minor $\{x, x'\} \times \{y, y'\}$, we compute the associated frontier $F_{x,x'}$. Similarly, for every $\boxplus$-minor $\{x, x'\} \times \{y, y'\}$, we compute the associated frontier $F_{y,y'}$. Let F_X be the prefix-minimal elements of $\bigcup_{x,x'} F_{x,x'}$, and let F_Y be the prefix-minimal elements of $\bigcup_{y,y'} F_{y,y'}$. Intuitively, F_X is the first place where Alice finalizes any distinctions among her inputs, and F_Y is the first place where Bob finalizes any distinctions among his inputs.

For an arbitrary $x \in X$, $y^* \in Y$, we will we will inductively construct a sequence $Y_1 \subset \cdots \subset Y_n = Y$ such that $\Delta\left(\mathcal{D}_{F_X}^{x,y^*}, \mathcal{D}_{F_X}^{x,y}\right) \leq i \cdot \nu$ for all $y \in Y_i$, where ν is the negligible quantity guaranteed by Lemma 4.10. That is, at F_X, transcripts are nearly independent of Bob's input. The base case is $Y_1 = \{y^*\}$.

For the inductive step, we know that $Y = Y_i \cup \overline{Y_i}$ is not a valid column decomposition step in $\mathcal{F}$. Thus we must have $\mathcal{F}(x', y) = \mathcal{F}(x', y')$ for some $x' \in X$, $y \in Y_i$, $y' \in \overline{Y_1}$. We now consider the 2×2 minor $\{x, x'\} \times \{y, y'\}$. If $\mathcal{F}(x, y) = \mathcal{F}(x, y')$, then clearly by the security of the protocol, $\Delta\left(\mathcal{D}_{F_X}^{x,y}, \mathcal{D}_{F_X}^{x,y'}\right) \leq \epsilon \leq \nu$. If $\mathcal{F}(x, y) \neq \mathcal{F}(x, y')$, then we cannot have $\mathcal{F}(x, y) = \mathcal{F}(x', y)$ or $\mathcal{F}(x, y') = \mathcal{F}(x', y')$, since this would make the 2×2 minor in question an OR-minor. The only other case is that this 2×2 minor is a $\boxplus$-minor. As such, it has an associated frontier $F_{x,x'}$. Since F_X contains only prefixes of $F_{x,x'}$, we must have $\Delta\left(\mathcal{D}_{F_X}^{x,y}, \mathcal{D}_{F_X}^{x,y'}\right) \leq \Delta\left(\mathcal{D}_{F_{x,x'}}^{x,y}, \mathcal{D}_{F_{x,x'}}^{x,y'}\right) \leq \nu$. Thus in either of the above cases, we have:

$$\Delta\left(\mathcal{D}_{F_X}^{x,y^*}, \mathcal{D}_{F_X}^{x,y'}\right) \leq \Delta\left(\mathcal{D}_{F_X}^{x,y^*}, \mathcal{D}_{F_X}^{x,y}\right) + \Delta\left(\mathcal{D}_{F_X}^{x,y}, \mathcal{D}_{F_X}^{x,y'}\right) \leq (i-1)\nu + \nu = i \cdot \nu$$

Setting $Y_{i+1} = Y_i \cup \{y'\}$ completes the inductive step.

Summarizing, for any $x \in X$, the transcript distribution at F_X is nearly independent of Bob's input. Symmetrically, for any $y \in Y$, the transcript distribution at F_Y is nearly independent of Alice's input. We will now obtain a contradiction by showing that the probability of reaching F_X strictly before F_Y is overwhelming, as is the probability of reaching F_Y strictly before F_X — a clear contradiction since these two events are mutually exclusive.

Let $\{x, x'\} \times \{y, y'\}$ be a $\boxplus$-minor, and let $F_{x,x'}$ be its corresponding frontier. As in Lemma 4.13, we let "$F \ll G \mid x, y$" denote the event that the frontier F is encountered strictly before frontier G when honestly executing

the protocol on inputs x, y. We know that $\Delta\left(\mathcal{D}^{x,y}_{F_{x,x'}}, \mathcal{D}^{x',y}_{F_{x,x'}}\right) \geq 1 - \nu$, and also that $\Delta\left(\mathcal{D}^{x,y}_{F_Y}, \mathcal{D}^{x',y}_{F_Y}\right) \leq O(\nu)$. From Lemma 4.13, this implies that $\Pr[F_Y \ll F_{x,x'} \mid x, y] \geq 1 - O(\nu)$.

Now, checking whether a transcript reaches F_Y or $F_{x,x'}$ first is a statistical test that can be carried out on both of the distributions $\mathcal{D}^{x,y}_{F_Y}$ and $\mathcal{D}^{x,y}_{F_{x,x'}}$. Since $\Delta\left(\mathcal{D}^{x,y}_{F_{x,x'}}, \mathcal{D}^{x,y''}_{F_{x,x'}}\right) \leq O(\nu)$ for all $y'' \in Y$, the outcome of this statistical test must differ by at most $O(\nu)$ between input pairs (x, y) and (x, y''); that is, $\Pr[F_Y \ll F_{x,x'} \mid x, y''] \geq 1 - O(\nu)$. Similarly, since $\Delta\left(\mathcal{D}^{x,y''}_{F_Y}, \mathcal{D}^{x'',y''}_{F_Y}\right) \leq O(\nu)$ for all $x'' \in X$, we must also have $\Pr[F_Y \ll F_{x,x'} \mid x'', y''] \geq 1 - O(\nu)$. Thus, for an arbitrary $\boxplus$ minor with associated frontier $F_{x,x'}$, we have

$$\Pr[F_Y \ll F_{x,x'} \mid x'', y''] \geq 1 - O(\nu)$$

for all $x'' \in X$, $y'' \in Y$. Intuitively, when running the protocol on *any* input, the transcript will reach F_Y strictly before $F_{x,x'}$ with overwhelming probability. Since F_X is defined as the prefix-minimal elements of at most $\binom{|X|}{2}$ frontiers of the form $F_{x,x'}$, we have by a union bound that $\Pr[F_Y \ll F_X \mid x'', y''] \geq 1 - O(\nu)$ for all $x'' \in X$, $y'' \in Y$. In other words, the protocol (when running on any inputs) will encounter the F_Y frontier before the F_X frontier with overwhelming probability.

However, a symmetric argument using $\boxplus$-minors shows that the protocol will encounter the F_X frontier before the F_Y frontier with overwhelming probability. This is the desired contradiction. □

On SSFE functionalities with security parameters. The above result crucially relies on the fact that the domain of $\mathcal{F}$ does not grow too much as a function of the security parameter. (We use the fact that $|X|\nu$ and $|Y|\nu$ are both negligible in the security parameter if ν is negligible.) When statistical error is allowed, this restriction is necessary, as illustrated by the following example $\mathcal{F} : \mathbb{Z}_n \times \mathbb{Z}_n \to (\mathbb{Z}_n \cup \{\perp\})^2$:

$$\mathcal{F}(x, y) = \begin{cases} (\perp, y) & \text{if } x \in \{y, y+1\} \\ (x, \perp) & \text{if } y \in \{x+1, x+2\} \\ (x, y) & \text{otherwise} \end{cases}$$

All additions are modulo n. $\mathcal{F}$ is not row decomposable, since any partition of X would separate some adjacent pair of inputs $\{x, x+1\}$, and yet $\mathcal{F}(x, x+1) = \mathcal{F}(x+1, x+1)$. Similarly, $\mathcal{F}$ is not column decomposable.

However, one may also verify that $\mathcal{F}\big|_{X' \times Y'}$ is decomposable for any $X' \times Y' \subsetneq X \times Y$. This suggests the following protocol in which Alice randomly picks a value $x' \in X$, then announces x' and also announces whether her input is equal to x'. Her announcement induces a restriction of the function to either $\{x'\} \times Y$ or $(X \setminus \{x'\}) \times Y$, and the parties continue with the canonical protocol for the appropriate restriction of $\mathcal{F}$. The protocol is perfectly secure except when $x' \in \{x, x+1\}$, which happens with probability $2/n$. Thus when n is allowed to be superpolynomial in the security parameter, this protocol is statistically secure.

4.4.1 Characterization in Terms of UC Security

We also show that the natural class of passively realizable SSFE functionalities also has a simple characterization in terms of the $\sqsubseteq^u_{nt}$ reduction:

Theorem 4.18. *Let $\mathcal{F}$ be a 2-party SSFE functionality. Then $\mathcal{F}$ is passively realizable if and only if $\mathcal{F} \sqsubseteq^u_{nt} \mathcal{F}_{\text{COM}}$.*

To show a reduction to $\mathcal{F}_{\text{COM}}$, we give a general-purpose "compiler" that compiles any passive-secure protocol into a UC-secure protocol in the $\mathcal{F}_{\text{COM}}$-hybrid setting. In particular, this step of the proof applies to any passive-secure protocol for a non-reactive functionality, not just canonical protocols or protocols for SSFE functionalities. In particular, the output of the functionality may be non-symmetric or randomized. This compiler, like the well-known GMW compiler [41], may be of independent interest.

We break the proof into several steps, the first of which is the simple direction:

Proof of ($\Leftarrow$). Suppose π is a secure protocol for $\mathcal{F}$ in the $\mathcal{F}_{\text{COM}}$-hybrid setting. Recall that any non-trivial UC-secure protocol for a SSFE functionality $\mathcal{F}$ (in any hybrid setting) is also passive-secure (Lemma 4.3). There is a trivial passive-secure protocol for $\mathcal{F}_{\text{COM}}$ (the committing party sends "COMMITTED" in the commit phase and honestly sends the correct value in the reveal phase). We can compose π with the passive-secure commitment protocol to obtain a passive-secure protocol for $\mathcal{F}$ in the plain $\mathcal{F}_{\text{PVT}}$-hybrid setting. □

- On input (COMMIT, x, X) from Alice, where X is a set: Abort if $x \notin X$. Otherwise, send $(\text{COMMITTED}, X)$ to Bob.
- On input (PROVE, S) from Alice: Abort if $S \not\subseteq X$ or if $x \notin S$. Otherwise, send (PROVE, S) to Bob.

Figure 4.2: Commit-and-prove functionality $\mathcal{F}_{\text{C\&P}}$

For the other direction, we modularize the protocol construction into two steps, carried out in the following lemmas.

Lemma 4.19. *$\mathcal{F}_{\text{C\&P}} \sqsubseteq^{u}_{nt} \mathcal{F}_{\text{COM}}$, where $\mathcal{F}_{\text{C\&P}}$ is the commit-and-prove functionality defined in Figure 4.2.*

Proof. Our protocol for $\mathcal{F}_{\text{C\&P}}$ is as follows, with security parameter k:

1. On input (COMMIT, x, X), where $x \in X$, Alice lets $\chi \in \{0,1\}^{|X|}$ denote the characteristic vector of x in X; that is, $\chi_i = 1$ if $i = x$ and 0 otherwise. Alice also chooses k random permutations of X: $\sigma_1, \ldots, \sigma_k$.
2. Alice sends X to Bob. Then she instantiates $2k|X|$ instances of $\mathcal{F}_{\text{COM}}$, which we denote as $\{\mathcal{F}^{\sigma}_{i,j}, \mathcal{F}^{\chi}_{i,j} \mid i \in [k], j \in X\}$. In each $\mathcal{F}^{\sigma}_{i,j}$, she commits to the value $\sigma_i(j) \in X$. In each $\mathcal{F}^{\chi}_{i,j}$, she commits to the value $\chi_{\sigma_i^{-1}(j)} \in \{0,1\}$.
3. Bob receives X and expects to receive $2k|X|$ commitments from the appropriate size domains. He then chooses a random string $c \leftarrow \{0,1\}^k$ and sends it to Alice.
4. For each $i \in [k]$, Alice does the following: If $c_i = 0$, then she opens the commitments $\mathcal{F}^{\sigma}_{i,j}$ for all j. Otherwise, if $c_i = 1$, then she opens the commitments $\mathcal{F}^{\chi}_{i,j}$ for all j.
5. Bob checks that Alice opened the appropriate commitments. For each i, Bob does the following: If $c_i = 0$, then he checks that the decommitments of $\{\mathcal{F}^{\sigma}_{i,j} \mid j \in X\}$ constitute a valid permutation of X. If $c_i = 0$, then he checks that the decommitments of $\{\mathcal{F}^{\chi}_{i,j} \mid j \in X\}$ contain exactly one 1 and the others 0. If not, then he aborts the protocol; otherwise, he outputs $(\text{COMMITTED}, X)$.
6. On input (PROVE, S), where $x \in S \subseteq X$, then Alice sends S to Bob and does the following: For each $i \in [k]$ and $j \in X \setminus S$, she opens the commitments of $\mathcal{F}^{\sigma}_{i,j}$ and of $\mathcal{F}^{\chi}_{i,\sigma_i(j)}$, if they have not already been opened at an earlier time.
7. Bob receives S and verifies that $S \subseteq X$. He ensures that Alice has opened (either in this round or at some earlier time) $\mathcal{F}^{\sigma}_{i,j}$ for each $i \in [k]$ and $j \in X \setminus S$. Suppose $\mathcal{F}^{\sigma}_{i,j}$ has been opened to $\hat{\sigma}_{i,j}$ (either in this round or at some earlier time). For each $i \in [k]$, he ensures that the known values $\{\hat{\sigma}_{i,j}\}$ contain no duplicates, and are all elements of X. Then he ensures that Alice has opened (either in this round or at some earlier time) $\mathcal{F}^{\chi}_{i,\hat{\sigma}_{i,j}}$ to the value 0, for each $i \in [k]$ and $j \in X \setminus S$. If all of these checks succeed, then he outputs (PROVE, S).

Correctness of the protocol when both parties are honest is easily verified by inspection. Further, $\mathcal{F}_{\text{C\&P}}$ never receives input from Bob, so simulation is trivial when the adversary corrupts Bob alone. We now construct a simulator for the case when the adversary corrupts Alice alone.

The simulator faithfully simulates an honest Bob (who has no input) and $\mathcal{F}_{\text{COM}}$ instances in all steps. If the simulated Bob aborts in step (5), then the simulation also aborts. The simulation is clearly perfect in this case. We now analyze the simulation in the case where Bob does not abort in step (5).

The simulator sees the values that Alice has sent to the $\mathcal{F}_{\text{COM}}$ instances. Let $\hat{\chi}_{i,j}$ and $\hat{\sigma}_{i,j}$ be the values that Alice sent to $\mathcal{F}^{\chi}_{i,j}$ and $\mathcal{F}^{\sigma}_{i,j}$, respectively. Call an index $i \in [k]$ *valid* if the values $\{\hat{\chi}_{i,j} \mid j \in X\}$ contain exactly one 1 and the rest 0s, and if the values $\{\hat{\sigma}_{i,j} \mid j \in X\}$ are a permutation of X. If all indices $i \in [k]$ are invalid, then the simulated Bob would have aborted in step (5) with overwhelming probability $1 - 2^{-k}$. Thus we condition on the event that some index i is valid.

Let i be any valid index, and let $\hat{x} \in X$ be the unique value such that $\hat{\chi}_{i,\hat{\sigma}_{i,\hat{x}}} = 1$. The simulator sends $(\text{COMMIT}, \hat{x})$ to $\mathcal{F}_{\text{C\&P}}$. Now throughout the simulation, if the simulated Bob outputs (PROVE, S), then the simulator sends (PROVE, S) to $\mathcal{F}_{\text{C\&P}}$. It suffices to show that the simulated Bob never outputs (PROVE, S) such that $\hat{x} \notin S$. However, to convince Bob that the committed value lies in S, Alice must first open $\mathcal{F}^{\sigma}_{i,\hat{x}}$ for all $i \in [k]$. In particular, she must do this for the valid i considered above. Then for that i, she must open $\mathcal{F}^{\chi}_{i,\hat{\sigma}_{i,\hat{x}}}$. By our assumption, this commitment can only be opened to the value 1, so the simulated Bob does not output (PROVE, S) in this case. Thus, apart from the negligible error probability 2^{-k}, the overall simulation is perfect. □

Lemma 4.20. *Let $\mathcal{F}$ be any non-reactive functionality. If $\mathcal{F}$ is securely realizable against passive, unbounded adversaries, then $\mathcal{F} \sqsubseteq^{u}_{nt} \mathcal{F}_{\text{C\&P}}$.*

Proof. We describe a general-purpose procedure for compiling an arbitrary passive-secure protocol to a UC-secure protocol in the $\mathcal{F}_{\text{C\&P}}$-hybrid setting. Suppose π is a passive-secure protocol for $\mathcal{F}$, in random-tape normal form (that is, all random coin tosses are made ahead of time, and the protocol's next message function is a deterministic function of the random tape, input, and transcript so far). Since $\mathcal{F}$ is non-reactive, π takes from each party a random tape from some domain R_k and a single input, both chosen at the beginning of the interaction. The compiled protocol is as follows, with security parameter k:

1. On input $x \in X$, Alice chooses a random $r_A \leftarrow R_k$ and sends $(\text{COMMIT}, (x, r_A), X \times R_k)$ to an instance of $\mathcal{F}_{\text{C\&P}}$.
2. Bob expects $(\text{COMMITTED}, X \times R_k)$ from the first instance of $\mathcal{F}_{\text{C\&P}}$. Otherwise he aborts.
3. On input $y \in Y$, Bob chooses a random $r_B \leftarrow R_k$ and sends $(\text{COMMIT}, (y, r_B), Y \times R_k)$ to a new instance of $\mathcal{F}_{\text{C\&P}}$.
4. Alice expects $(\text{COMMITTED}, Y \times R_k)$ from the second instance of $\mathcal{F}_{\text{C\&P}}$. Otherwise she aborts.
5. Bob chooses a random $r'_A \leftarrow R_k$ and sends it to Alice.
6. Alice chooses a random $r'_B \leftarrow R_k$ and sends it to Bob.
7. Both parties maintain a partial transcript τ for the protocol π, initialized to the empty transcript. For each step of the protocol π:
   (a) If it is Alice's turn, then she determines the next message m in the π protocol on input x, random tape $r_A \oplus r'_A$, and transcript τ seen before. She sends m to Bob. Both parties can compute S, the subset of $X \times R_k$ of input/random tape pairs that are consistent with the message m after seeing transcript τ so far. Alice sends (PROVE, S) to his instance of $\mathcal{F}_{\text{C\&P}}$, and Bob expects to receive (PROVE, S); otherwise he aborts. Both parties update $\tau \leftarrow \tau \| m$.
   (b) If it is Bob's turn, then the parties do as above, with their roles reversed.

   When π terminates, the parties output its result.

The correctness of the protocol is clear, when both parties are honest. We show a simulator for the case where the adversary corrupts Alice alone; the other case is symmetric. It suffices to reduce the security of the compiled protocol to the passive security of π. For this, we show a simulator interacting as a *passively* corrupt Alice in the π protocol achieves the same effect as the corrupt Alice in the compiled protocol interaction.

The simulation is as follows: It first honestly samples a random tape $r \leftarrow R_k$ for its external π interaction. Then it starts internally running the corrupt Alice. In step (1), when Alice sends $(\text{COMMIT}, (x, r_A), X \times R_k)$ to $\mathcal{F}_{\text{C\&P}}$, the simulator internally records x. In step (3), it sends $(\text{COMMITTED}, Y \times R_k)$ to Alice, and in step (5) sends $r'_A = r \oplus r_A$. Then the simulator starts running π honestly on input x and random tape r. If the messages sent by the corrupt Alice in step (7a) ever disagree with the simulator's execution of π, or if the corrupt Alice's inputs to $\mathcal{F}_{\text{C\&P}}$ would cause an honest Bob to abort, then the simulator aborts. Whenever the simulator receives a message m in the π protocol from the external honest Bob, it simulates that Bob sent m and that Alice received (PROVE, S) for the appropriate S.

It is clear that the simulator is a passive adversary in the π protocol – it honestly samples a random tape and executes π honestly on some fixed input throughout its interaction. It aborts whenever Bob would abort in the compiled protocol, except possibly also in the case where the corrupt Alice sends a message that disagrees with the simulator's honest execution of π. However, it is easy to see that an honest Bob would have immediately aborted in these cases as well. By construction, the simulator has computed message m as the next message in the π protocol. If the corrupt Alice sends message $m' \neq m$, then she would have to use $\mathcal{F}_{\text{C\&P}}$ to prove that her committed value (x, r_A) is consistent with m' and r'_A from step (5). But by construction, these values are consistent only with m, as computed by the simulator. □

We note that for deterministic protocols (which are all that are strictly necessary to prove the characterization for SSFE), the overhead of the compiler is linear in the input size, while for randomized protocols, it is exponential in the randomness complexity.

4.5 Characterizing Standalone Security

Our main tool, Theorem 4.12, can apply to the standalone security setting, but only to protocols for uniquely decomposable SSFE functionalities. In this section we first use a specialized argument for standalone security to prove that all standalone-realizable SSFE functionalities are uniquely decomposable. Then, we can continue our approach of applying Theorem 4.12 to identify the additional properties that exactly characterize standalone realizability.

Lemma 4.21. *Let $\mathcal{F}$ be a 2-party SSFE functionality. If $\mathcal{F}$ is standalone-realizable, then $\mathcal{F}$ is uniquely decomposable.*

Proof. Suppose $\mathcal{F}$ has a standalone-secure protocol π. By Lemma 4.3, we have that π is also passive-secure, and thus $\mathcal{F}$ is decomposable. Without loss of generality, assume that π and $\mathcal{F}$ are in normal form, with the following additional properties:

- If inputs (x, y) and (x', y') induce different transcripts in some canonical protocol for $\mathcal{F}$ (i.e., they are in different parts of some decomposition for $\mathcal{F}$), then $\mathcal{F}(x, y) \neq \mathcal{F}(x', y')$. This is without loss of generality for decomposable functionalities, since $\mathcal{F}$ is isomorphic to such a functionality.
- If either party aborts the protocol, it first announces in the protocol that it is aborting. Also recall that in our normal form, honest parties announce in the protocol their final outputs as their last message.

In an interaction between a corrupt party and an honest party running the π protocol, call an *outcome* of the interaction either the output of the honest party, or the symbol $\perp_A$ or $\perp_B$, indicating an abort by honest Alice or by honest Bob, respectively. By the properties of our normal form, the outcome of the interaction is a publicly computable function of the transcript.

Suppose we partition the set of possible *outcomes* into two sets, red and blue. Let the predicate $R(m_1 m_2 \cdots m_n)$ denote the event that the outcome of transcript $m_1 \cdots m_n$ is red, and let $B(m_1 m_2 \cdots m_n)$ denote the event that the outcome is blue. Then for any protocol whose outcome is a public function of the transcript, the protocol is one of the following two kinds (suppose Alice sends the first message in the protocol):

- $\exists m_1 \forall m_2 \cdots : R(m_1 \cdots m_n)$. This corresponds to a strategy for a corrupt Alice to induce a red outcome with probability 1.
- $\forall m_1 \exists m_2 \cdots : B(m_1 \cdots m_n)$. This corresponds to a strategy for a corrupt Bob to induce a blue outcome with probability 1.

Now for the sake of contradiction, assume that $\mathcal{F}$ has two distinct decompositions. We will show that there is then a way to partition the outcomes into red and blue so that neither of the two strategies above can be effected in the ideal world. This will contradict the supposed security of π.

We first claim that there exists a decomposition of $\mathcal{F}$ that contains a step $X' \times Y' \subseteq X \times Y$ where $\mathcal{F}\big|_{X' \times Y'}$ is simultaneously row- and column-decomposable. Take any two distinct decompositions of $\mathcal{F}$, and consider the earliest step at which they differ. Without loss of generality, assume that they differ at the first step. If one decomposition starts with a row-decomposition step and the other starts with a column-decomposition step, then we are done. Otherwise, $\mathcal{F}$ is (by symmetry) row-decomposable in two distinct ways, say, $X \times Y = (X_1 \cup \cdots \cup X_m) \times Y = (X'_1 \cup \cdots \cup X'_n) \times Y$, with $\{X_1, \ldots, X_m\} \neq \{X'_1, \ldots, X'_n\}$. Then by symmetry, there must be some X_i and X'_j such that $X_i \cap X'_j$ and $X_i \cap (X \setminus X'_j)$ are both non-empty. One can easily verify that $\mathcal{F}\big|_{X_i \times Y}$ can be row-decomposed starting with the step $X_i = (X_i \cap X'_j) \cup (X_i \setminus X'_j)$. In particular, this implies that $\mathcal{F}$ is not constant over $X_i \times Y$. Also, $X_i \times Y$ appears in one of the two row-decompositions of $\mathcal{F}$, so $\mathcal{F}\big|_{X_i \times Y}$ must also be column-decomposable (row- and column-decomposition steps must alternate).

We will let the outcome $\perp_A$ (an abort by honest Alice) be red, and the outcome $\perp_B$ (an abort by honest Bob) be blue. Now we inductively color the output-outcomes as follows. In the base case, $\mathcal{F}$ is both row- and column-decomposable, say, as $X \times Y = (X_1 \cup \cdots \cup X_m) \times Y$ and $X \times Y = X \times (Y_1 \cup \cdots \cup Y_n)$. The output ranges of $\mathcal{F}\big|_{X_i \times Y_j}$ and $\mathcal{F}\big|_{X_k \times Y_l}$ are disjoint when $(X_i, Y_j) \neq (X_k, Y_l)$, due to the normal form of $\mathcal{F}$. We will color the outputs as a checkerboard; that is, the outputs of $\mathcal{F}\big|_{X_i \times Y_j}$ are red if $i + j$ is even and blue if $i + j$ is odd. Note that there is no monochromatic red row or monochromatic blue column in the function table for $\mathcal{F}$.

For the inductive step, if $\mathcal{F}$ is row-decomposable as $X \times Y = (X_1 \cup \cdots \cup X_m) \times Y$, then for some i, $\mathcal{F}\big|_{X_i \times Y}$ contains a step which is both row- and column-decomposable. We color these outputs inductively (they are disjoint from the outputs in other parts $X_j \times Y$, for $j \neq i$), and color the remaining outputs of $\mathcal{F}$ red. Note that this cannot

have induced a monochromatic red row or monochromatic blue column in the function table for $\mathcal{F}$. Similarly, if $\mathcal{F}$ is column-decomposable, we inductively color one of the subproblems and color the remaining outputs blue.

The resulting coloring leaves no monochromatic red row or monochromatic blue column. That is, there is no input for Alice which guarantees a red output-outcome; and by not delivering Bob's output in the ideal interaction, Alice can only induce the blue outcome $\perp_B$. Similarly, there is no strategy for Bob which guarantees a blue outcome in the ideal interaction. This suggests the following environment to distinguish the real and ideal interactions: Expect the adversary to corrupt one party, and choose a random input for the honest party. If the honest party is Alice and the outcome is blue, or if the honest party is Bob and the outcome is red, then output 1. We have shown that an adversary attacking π can always cause such an environment to output 1 with probability 1. However, in the ideal world, the environment outputs 1 with probability at most $1 - \min\{1/|X|, 1/|Y|\}$, which is bounded away from 1 by a constant. This environment distinguishes between π and the ideal interaction with constant probability, so π is not secure. Thus we conclude that $\mathcal{F}$ must be uniquely decomposable. □

We now develop our characterization of standalone realizability.

Decomposition strategies. Let $\mathcal{F}$ be a uniquely decomposable 2-party SSFE functionality. We define an *Alice-strategy* as a function that maps every row decomposition step $X \times Y = (X_1 \cup \cdots \cup X_t) \times Y$ to one of the X_i's. Similarly we define a *Bob-strategy* for the set of column decomposition steps. If A and B are Alice and Bob-strategies for $\mathcal{F}$, respectively, then we define $\mathcal{F}^*(A, B)$ to be the subset $X' \times Y' \subseteq X \times Y$ obtained by "traversing" the decomposition of $\mathcal{F}$ according to the choices of A and B.

The definition of $\mathcal{F}^*$ is easy to motivate: it denotes the outcome in the canonical protocol for $\mathcal{F}$, as a function of the strategies of (possibly corrupt) Alice and Bob.

Definition 4.22 (Saturation)**.** *Let $\mathcal{F}$ be a uniquely decomposable 2-party SSFE functionality. We say that $\mathcal{F}$ is* saturated *if $\mathcal{F}$ is isomorphic to $\mathcal{F}^*$.*

To understand this condition further, we provide an alternate description for it. For every $x \in X$ we define an Alice-strategy A_x such that at any row decomposition step $X' \times Y' = (X_1 \cup \cdots \cup X_t) \times Y'$ such that $x \in X'$, the strategy chooses the unique X_i such that $x \in X_i$. For X' such that $x \notin X'$, the choice is arbitrary, say, X_1. Similarly for all $y \in Y$ we define a corresponding Bob-strategy B_y. Note that in the canonical protocol, on inputs x and y, Alice and Bob traverse the decomposition of $\mathcal{F}$ according to the strategy (A_x, B_y), to compute the set $\mathcal{F}^*(A_x, B_y)$. If $\mathcal{F}$ is saturated, then all Alice strategies should correspond to some x that Alice can use as an input to $\mathcal{F}$. That is, for all Alice-strategies A, there exists an $x \in X$ such that for all $y \in Y$, we have $\mathcal{F}^*(A, B_y) = \mathcal{F}^*(A_x, B_y)$; similarly each Bob strategy B is equivalent to some B_y.

As an example, $\boxed{\begin{smallmatrix}0&1\\2&3\end{smallmatrix}}$ is not uniquely decomposable. $\boxed{\begin{smallmatrix}0&1&1\\2&3&2\end{smallmatrix}}$ is uniquely decomposable, but not saturated. Finally, $\boxed{\begin{smallmatrix}0&1&1&0\\2&3&2&3\end{smallmatrix}}$ is saturated.

Note that there is exactly one saturated function (up to isomorphism) for each decomposition structure.

Theorem 4.23. *Let $\mathcal{F}$ be a 2-party SSFE functionality. Then $\mathcal{F}$ is standalone-realizable if and only if $\mathcal{F}$ is saturated.*

Proof. ($\Leftarrow$) Suppose $\mathcal{F}$ is saturated, then we will show that its canonical protocol is standalone secure. Clearly the canonical protocol is correct, so it suffices to construct a (possibly rewinding) simulator for a corrupt party. The simulator for an adversary $\mathcal{A}$ does the following: By symmetry, suppose $\mathcal{A}$ corrupts Alice. First fix a random tape ω for $\mathcal{A}$, then for every Bob-strategy B, run the canonical protocol against $\mathcal{A}$ (using random tape ω), effecting the strategy B. The (deterministic) choices of $\mathcal{A}$ at each decomposition step (equivalently, step of the canonical protocol) uniquely determine an Alice-strategy A. By the saturation of $\mathcal{F}$, A is equivalent to some A_x strategy, and the simulator sends x to the functionality $\mathcal{F}$ in the ideal world. After receiving the output $\mathcal{F}(x, y)$, the simulator gives to the adversary the (unique) canonical-protocol transcript consistent with $\mathcal{F}(x, y)$.

($\Rightarrow$) Suppose $\mathcal{F}$ is standalone-realizable. Then by Lemma 4.21, we know that $\mathcal{F}$ is uniquely decomposable. By Corollary 4.15, then the canonical protocol for $\mathcal{F}$ is standalone-secure. However, suppose for contradiction that $\mathcal{F}$ is not saturated. Then there is (by symmetry) an Alice-strategy A that corresponds to no input-strategy A_x. We will show that the canonical protocol is in fact not standalone-secure in this case, a contradiction. The attack on the canonical protocol is for a corrupt Alice to effect the strategy A on the canonical protocol. The environment chooses a random input y for Bob and outputs 1 if Bob reports an output consistent with Bob-strategy B_y and the Alice-strategy A. Clearly the environment outputs 1 with probability 1 in the real protocol interaction. However, in the ideal world, the simulator must send some input x to $\mathcal{F}$ (chosen independently of the environment's choice of y) and Bob's output is $\mathcal{F}(x, y)$. But with probability at least $1/|Y|$, a constant, we have that y is one of the inputs for which $\mathcal{F}^*(A, B_y)$ and $\mathcal{F}^*(A_x, B_y)$ disagree, so the environment outputs 0.

This environment distinguishes the real protocol interaction from the ideal interaction. Thus we conclude that $\mathcal{F}$ is saturated. □

4.6 Impossibility Results for Concurrent Self-Composition

Recall that in the concurrent self-composition setting, the parties can initiate several concurrent instances of a *single* protocol, but with the same roles for each instance. Furthermore, the environment does not interact with the adversary throughout the execution of these protocol instances.

Lindell [70] showed that security under general (that is, with no limit on the number of protocol instances) concurrent self-composition is equivalent to full-fledged UC security, for almost all functionalities. His result was framed in the PPT setting, though the proof also holds in the unbounded setting that we consider in this chapter.

Lindell's proof heavily relies on the assumption that there is no *a priori* limit to the number of concurrent protocol instances. Indeed, in the case where there is such a fixed bound, chosen before the protocol is constructed (a setting called *bounded* concurrent self-composition), Lindell [69] and Pass and Rosen [80] constructed secure protocols for all SFE functionalities.

It remained open whether there is such a difference between bounded and unbounded concurrency, in the computationally unbounded setting as well. Backes et al. [3] gave an example protocol that was standalone-secure, and even had a perfect, efficient (rewinding) simulator, but was not secure against 2 concurrent instances. They left open the question of whether it is their choice of protocol which prevented security under concurrent self-composition (that is, the protocol requires a rewinding simulation), or an inherent property of the functionality. Using our new techniques, we are able to extend their concurrency attack and settle the question for the case of uniquely decomposable SSFE functionalities.

Theorem 4.24. *Let $\mathcal{F}$ be a 2-party SSFE functionality with unique decomposition. If $\mathcal{F}$ is not UC-realizable, then $\mathcal{F}$ has no protocol secure against even two concurrent instances.*

Proof. An SSFE functionality $\mathcal{F}$ as in the theorem statement must be uniquely decomposable with decomposition depth at least 2 (since SSFE functionalities with decomposition depth 1 are UC-realizable; Theorem 3.21). We show a concurrent attack against two instances of the *canonical protocol* for $\mathcal{F}$, which by Corollary 4.15 implies that $\mathcal{F}$ is not realizable under concurrent self-composition of two instances. Our attack is a simple generalization of the attack from [3].

By symmetry, suppose Alice speaks first in the canonical protocol, and let x, x' be two inputs which induce different messages in the first round of the protocol. Let y, y' be two inputs which induce different messages in the second round of the protocol when Alice has input x (thus $\mathcal{F}(x, y) \neq \mathcal{F}(x, y')$). We will construct an adversary that corrupts Bob. Consider an environment which runs two instances of $\mathcal{F}$, supplies inputs for Alice for the instances, and outputs 1 if Alice reports particular expected outputs for the two instances. The environment chooses randomly from one of the following three cases:

1. Supply inputs x and x', respectively. Expect outputs $\mathcal{F}(x, y')$ and $\mathcal{F}(x', y)$, respectively.
2. Supply inputs x' and x, respectively. Expect outputs $\mathcal{F}(x', y)$ and $\mathcal{F}(x, y')$, respectively.
3. Supply inputs x and x, respectively. Expect outputs $\mathcal{F}(x, y)$ and $\mathcal{F}(x, y)$, respectively.

A malicious Bob can cause the environment to output 1 with probability 1 in the real world. He waits to receive Alice's first message in *both* canonical protocol instances to determine whether she has x or x' in each instance. Then he can continue the protocols with inputs y or y', as appropriate.

In the ideal world, Bob must send an input to one of the instances of $\mathcal{F}$ before the other (i.e., before learning anything about how Alice's inputs have been chosen); suppose he first sends $\hat{y}$ to the first instance of $\mathcal{F}$. If $\mathcal{F}(x, \hat{y}) \neq \mathcal{F}(x, y)$, then with probability 1/3, he induces the wrong output in case (3). But if $\mathcal{F}(x, \hat{y}) = \mathcal{F}(x, y) \neq \mathcal{F}(x, y')$, then with probability 1/3, he induces the wrong output in case (1). Similarly, if he first sends an input to the second instance of $\mathcal{F}$, he violates either case (2) or (3) with probability 1/3. □

4.7 Cryptographic Complexity Reductions

In this section we develop a new technique for deriving separations among SSFE functionalities with respect to the $\sqsubseteq^{u}_{nt}$ relation. We apply the technique to specific functionalities to uncover interesting structures within the landscape of cryptographic complexity.

Theorem 4.25. *If $\mathcal{G}$ is a 2-party non-reactive functionality (not necessarily deterministic or uniquely decomposable SSFE) with a passive-secure, m-round protocol, and $\mathcal{F}$ is a 2-party SSFE functionality with unique decomposition of depth $n > m$, then $\mathcal{F} \not\sqsubseteq^{u}_{nt} \mathcal{G}$.*

Proof. Suppose for contradiction that π is a protocol for $\mathcal{F}$ in the $\mathcal{G}$-hybrid setting. Without loss of generality since $\mathcal{G}$ is non-reactive, assume that every instance of $\mathcal{G}$ invoked by π is invoked the following way: In the same step, both parties send an input to $\mathcal{G}$ and then wait for their outputs to be delivered before continuing.

By Lemma 4.3, π is also passive-secure in the same setting. Define $\widehat{\pi}$ to be the result of replacing each call to $\mathcal{G}$ in π with the m-round passive-secure protocol for $\mathcal{G}$. For simplicity, we assume that the first message in $\mathcal{G}$'s secure protocol is always prefixed with SUBPROT-START, the last message is always prefixed with SUBPROT-END, and all other messages are prefixed with SUBPROT. By our assumption about how instances of $\mathcal{G}$ are invoked (in particular, that both parties wait for $\mathcal{G}$ to give output before continuing), we have that each instance of $\mathcal{G}$'s protocol appears as a continuous subsequence of the $\widehat{\pi}$ protocol — that is, messages from different $\mathcal{G}$-protocol instances are not interleaved. By the composability of passive-secure protocols, we have that $\widehat{\pi}$ is a passive-secure protocol for $\mathcal{F}$ in the plain ($\mathcal{F}_{\text{PVT}}$) setting.

We will eventually apply Theorem 4.12 to derive an adversary attacking $\widehat{\pi}$. The proof of Theorem 4.12 constructs an adversary of the following form: it maintains a local variable for its input. At each step of the protocol, it possibly changes the contents of that variable, then runs the protocol honestly using the contents of the variable as the input to the protocol. Say that such an adversary *behaves consistently* for a span of protocol rounds if throughout that span, it does not change the contents of its input variable. We call a step of the $\widehat{\pi}$ protocol *off-limits* if the last message was prefixed with SUBPROT or SUBPROT-START (but not SUBPROT-END — the subprotocol is already complete if the last message begins with SUBPROT-END).

Now suppose that $\mathcal{S}$ is an adversary that behaves consistently at every off-limit step in $\widehat{\pi}$. Then there is an adversary $\mathcal{S}'$ which achieves the same effect in the π protocol (in the $\mathcal{G}$-hybrid setting) for all environments. $\mathcal{S}'$ simply simulates $\mathcal{S}$, except at each point in π when the next message is expected to begin with SUBPROT-START, $\mathcal{S}'$ instead sends the input to $\mathcal{G}$ that π prescribes in this step when using the current contents of $\mathcal{S}$'s input variable. Since $\mathcal{S}$ is guaranteed to behave consistently consistently until after the corresponding SUBPROT-END message has been received, it behaves passively throughout the entire subprotocol. Thus $\mathcal{S}'$ uses the passive security of the protocol for $\mathcal{G}$ to simulate the SUBPROT-messages of the $\widehat{\pi}$ protocol, given its input and output from $\mathcal{G}$.

Thus, it suffices to construct an adversary $\mathcal{S}$ that violates the security of $\mathcal{F}$, and that behaves consistently in every off-limits steps of $\widehat{\pi}$. This will contradict the assumption that π is a secure realization of $\mathcal{F}$ in the $\mathcal{G}$-hybrid setting. For this, we first construct an attack on the canonical protocol for $\mathcal{F}$ and use a modification of Theorem 4.12 to derive an adversary $\mathcal{S}$ attacking $\widehat{\pi}$ with the desired properties.

Let $x_0 \in X$, $y_0 \in Y$ be inputs that cause the canonical protocol for $\mathcal{F}$ to take $n \geq 2$ rounds, the maximum number of rounds. Let r and r' denote the indices of the first and last round, respectively, in which Alice speaks in the canonical protocol for $\mathcal{F}$ on inputs (x_0, y_0). Similarly, let s and s' denote the first and last rounds in which Bob speaks. Then $\{r, s\} = \{1, 2\}$, and $\{r', s'\} = \{n-1, n\}$.

When running $\widehat{\pi}$ honestly on inputs (x_0, y_0), we encounter each of the n transcript frontiers (corresponding to each round of the canonical protocol) in strict order, with overwhelming probability; that is, we do not encounter multiple frontiers in the same step of the protocol. As such, there are at least n steps of $\widehat{\pi}$ between encountering the first and the nth frontier, inclusively. Note that at most $m - 1 \leq n - 2$ consecutive steps of $\widehat{\pi}$ are off-limits, by our definition of off-limits. Then by the pigeonhole principle, at least one of the following probabilities must be at least 1/2:

- The probability of encountering a non-off-limits step between the s-th and the $(r' - 1)$-th frontiers of $\widehat{\pi}$ (inclusively), when executing $\widehat{\pi}$ honestly on inputs (x_0, y_0).
- The probability of encountering a non-off-limits step between the r-th and the $(s' - 1)$-th frontiers of $\widehat{\pi}$ (inclusively), when executing $\widehat{\pi}$ honestly on inputs (x_0, y_0).

By symmetry, suppose the second probability is at least 1/2. Let y_1 be such that the unique canonical-protocol transcripts for inputs (x_0, y_0) and (x_0, y_1) agree until Bob's last message; thus $\mathcal{F}(x_0, y_0) \neq \mathcal{F}(x_0, y_1)$. Let x_1 be an input for Alice such that x_0 and x_1 induce different message in Alice's first turn of the canonical protocol, when Bob uses input y_0. Now consider an environment which expects the adversary to corrupt Bob, and which does the following:

1. Choose random $b \leftarrow \{0, 1\}$, and supply input x_b to Alice.
2. Expect $b' \in \{0, 1\}$ from the adversary. If $b' \neq b$ or $b = 1$, then output 0.

3. If $b = 0$, then choose random $c \leftarrow \{0, 1\}$ and give c to the adversary.
4. Output 1 if Alice reports output $\mathcal{F}(x_b, y_c)$, and 0 otherwise.

The conditions enforced by this environment are similar to those of a "split adversary" considered by [22]. We claim that, when interacting with any adversary in the ideal world, the environment outputs 1 with probability at most 3/4. If the adversary gives an input y to $\mathcal{F}$ before reporting its guess of b to the environment, then its choice of y is independent of the environment's choice of c. With probability 1/4, the environment chooses $b = 0$ and chooses the c such that $\mathcal{F}(x_b, y_c)$ disagrees with $\mathcal{F}(x_b, y)$, so the environment outputs 0. If the adversary reports its guess of b to the environment before interacting with $\mathcal{F}$, then its guess is independent of the environment's choice of b, so the environment outputs 0 with probability 1/2.

Thus it suffices to design a suitable attack on the $\widehat{\pi}$ protocol that succeeds with probability noticeably greater than 3/4.

A straight-forward adversary $\mathcal{A}$ attacking the canonical protocol for $\mathcal{F}$ can make the environment output 1 with probability 1: It runs the canonical protocol on input y_0. After the first message from Alice (round r), it correctly determines whether Alice's input is x_0 or x_1, and reports accordingly. After receiving c from the environment, it thereafter runs the canonical protocol on input y_c. Note that the difference between y_0 and y_1 does not make a difference in the protocol until round s'.

Applying Theorem 4.12 to $\mathcal{A}$ results in a $\mathcal{S}$ that attacks the $\widehat{\pi}$ protocol. $\mathcal{S}$ only needs to change its input variable at most once: possibly from y_0 to y_1. By the choice of x_0 and x_1, $\mathcal{S}$ has determined whether it needs to swap as soon as it encounters the r-th frontier. Similarly, the resulting simulation is sound if $\mathcal{S}$ swaps its input any time except after leaving the $(s' - 1)$-th frontier (at the $(s' - 1)$th frontier, inputs y_0 and y_1 still induce statistically indistinguishable transcript distributions). We modify $\mathcal{S}$ so that whenever it wants to swap its input from y_0 to y_1, it waits for a non-off-limits step in this range to do so. If it does not encounter such a step, then $\mathcal{S}$ does not change its input and continues with input y_0. Thus $\mathcal{S}$ behaves consistently.

The probability that when interacting with $\mathcal{S}$, the environment outputs 0 is at most a negligible amount plus the probability that $b = 0$ and $c = 1$ and $\mathcal{S}$ is unable to swap its input. By our assumption, this event happens with probability at most 1/8, so the environment outputs 1 with probability negligibly close to 7/8. Thus, $\mathcal{S}$ violates the security of $\mathcal{F}$ in this environment. Since $\mathcal{S}$ behaves consistently, we can construct a $\mathcal{S}'$ that attacks π in the $\mathcal{G}$-hybrid setting and achieves the same effect. Finally, this attack $\mathcal{S}'$ violates the supposed security of π. □

From the relationship between a functionality's decomposition depth and the round complexity of the associated canonical protocol, we have the following combinatorial criterion:

Corollary 4.26. *If $\mathcal{F}$ is a 2-party SSFE functionality with unique decomposition depth m and $\mathcal{G}$ is a 2-party SSFE functionality with a (not necessarily unique) decomposition of depth $n < m$, then $\mathcal{F} \not\sqsubseteq^{u}_{nt} \mathcal{G}$.*

Put another way, decomposition depth (equivalently, round complexity of a passive-secure protocol; see Corollary 4.16) for uniquely decomposable SSFE functionalities is a *monotonic* quantity in the unbounded UC security setting, which cannot be increased via any protocol. For a certain class of 2-party SSFE functionalities $\mathcal{G}$, we are able to slightly strengthen Theorem 4.25, to also derive separations between pairs of SSFE functionalities that have the *same* decomposition depth:

Theorem 4.27. *If $\mathcal{F}$ is a 2-party SSFE that has unique decomposition depth at least 2, and $\mathcal{G}$ is (isomorphic to) a functionality of the form $\mathcal{G}(x, y) = (x, y)$,[3] then $\mathcal{F} \not\sqsubseteq^{u}_{nt} \mathcal{G}$.*

Proof sketch. The proof follows closely that of Theorem 4.25. We sketch the important differences.

First, observe that $\mathcal{G}$ has two distinct 2-round canonical protocols (and thus has decomposition depth 2): one in which Alice first announces x, then Bob announces y, and vice-versa. Now, let $\mathcal{A}$ be an adversary that attacks $\mathcal{F}$'s canonical protocol and violates the security of $\mathcal{F}$ in some environment (such an $\mathcal{A}$ must exist, since $\mathcal{F}$ has decomposition depth at least two and is therefore not UC-realizable; see Theorem 3.21). Suppose by symmetry that $\mathcal{A}$ corrupts Alice. We will translate $\mathcal{A}$ to an attack against any protocol for $\mathcal{F}$ in the $\mathcal{G}$-hybrid setting, using the approach of Theorem 4.25, except that we will replace instances of $\mathcal{G}$ with the canonical protocol for $\mathcal{G}$ in which Alice speaks first. (Similarly, if $\mathcal{A}$ corrupts Bob, we will use the canonical protocol for $\mathcal{G}$ in which Bob speaks first.) In this way, every *off-limits* step in the protocol is a turn in which Bob speaks. Thus every adversary corrupting Alice obtained by applying Theorem 4.12 to the $\widehat{\pi}$ protocol is an adversary which behaves consistently, and can be translated to an adversary attacking the π protocol that achieves the same effect. □

[3] Functionalities of this kind are called *symmetric exchange* functionalities, and are an important class in the results of Chapter 5. See Definition 5.22.

4.7.1 Infinite Hierarchy & Incomparable Complexities

Let $\mathcal{G}_n : \{0, 2, \ldots, 2n\} \times \{1, 3, \ldots, 2n+1\} \to \{0, 1, \ldots, 2n\}$ be defined as $\mathcal{G}_n(x, y) = \min\{x, y\}$. For example, $\mathcal{G}_1 = \boxed{\begin{smallmatrix} 0 & 0 \\ 1 & 2 \end{smallmatrix}}$ and $\mathcal{G}_2 = \boxed{\begin{smallmatrix} 0 & 0 & 0 \\ 1 & 2 & 2 \\ 1 & 3 & 4 \end{smallmatrix}}$. It can be seen by inspection that $\mathcal{G}_n$ has a unique decomposition of depth $2n$. The corresponding canonical protocol has at most $2n$ rounds, and is the one in which Alice first announces whether $x = 0$, then (if necessary) Bob announces whether $y = 1$, and so on. These are the "Dutch flower auction" protocols from [3].

Corollary 4.28. *The functionalities $\mathcal{G}_1, \mathcal{G}_2, \ldots$ form a strict, infinite hierarchy of increasing complexity under the $\sqsubseteq^u_{nt}$ relation. That is, $\mathcal{G}_i \sqsubseteq^u_{nt} \mathcal{G}_j$ if and only if $i \leq j$.*

Proof. By Theorem 4.25, $\mathcal{G}_i \not\sqsubseteq^u_{nt} \mathcal{G}_j$ when $i > j$. It suffices to show that $\mathcal{G}_n \sqsubseteq^u_{nt} \mathcal{G}_{n+1}$, since the $\sqsubseteq^u_{nt}$ relation is transitive. We claim that the following is a UC-secure protocol for $\mathcal{G}_n$ in the $\mathcal{G}_{n+1}$-hybrid world: both parties send their $\mathcal{G}_n$ inputs directly to $\mathcal{G}_{n+1}$, and abort if the response is out of bounds for the output of $\mathcal{G}_n$ (i.e., either $2n + 1$ or $2n + 2$).

The protocol is clearly correct when both parties are honest, thus it suffices to show a simulator for an adversary who corrupts a party. Suppose a real-world adversary attempts to send s to $\mathcal{G}_{n+1}$ in the protocol. If $s \leq 2n + 1$, then s is a valid input for $\mathcal{G}_n$, and the other party will accept, so the simulator simply sends s as the input in the ideal world, and delivers the output; this simulation is perfect. If the corrupt party is Bob, and $s = 2n + 3$, then since Alice is honest, her input will always be the minimum and she will never abort. The simulator can send $2n + 1$ to $\mathcal{G}_n$ in the ideal world (the maximum possible value) and deliver the output. If the corrupt party is Alice and $s = 2n + 2$, then if Bob's input to the protocol was $2n + 1$, he would abort in the real world interaction. So the simulator sends $2n$ to $\mathcal{G}_n$ and delivers the output unless the output is $2n$, which indicates that Bob's input must have been $2n + 1$. Note that we crucially use the fact that the simulator can block outputs. □

A consequence of Corollary 4.28 is that there is a UC-secure protocol for $\mathcal{G}_n$ in the $\mathcal{G}_m$-hybrid setting if and only if there is such a protocol that is deterministic, has perfect security, and invokes only a *single* instance of $\mathcal{G}_m$.

Corollary 4.29. *There is no functionality $\mathcal{F}$ which is complete (with respect to the $\sqsubseteq^u_{nt}$ relation) for the class of passively realizable 2-party SSFE functionalities, or for the class standalone-realizable 2-party SSFE functionalities.*

Proof. This follows from Theorem 4.25 and by observing that there exist functionalities of arbitrarily high decomposition depth in both classes in question. □

Corollary 4.30. *There exist functionalities $\mathcal{F}$ and $\mathcal{G}$ whose complexities are incomparable; that is, $\mathcal{F} \not\sqsubseteq^u_{nt} \mathcal{G}$ and $\mathcal{G} \not\sqsubseteq^u_{nt} \mathcal{F}$.*

Proof. If $\mathcal{F}$ is passively realizable but not standalone realizable, and $\mathcal{G}$ is standalone realizable, then $\mathcal{F} \not\sqsubseteq^u_{nt} \mathcal{G}$, since the class of standalone-realizable SFE functionalities is closed under composition. This is because without loss of generality (as in the normal form considered in the proof of Theorem 4.25), each call to a non-reactive functionality can be made synchronous in a single step of the calling protocol. Thus no instances of the functionality are concurrently active.

On the other hand, if $\mathcal{F}$ and $\mathcal{G}$ are 2-party decomposable SSFE functionalities, and $\mathcal{G}$ has unique decomposition depth larger than the decomposition depth of $\mathcal{F}$, then $\mathcal{G} \not\sqsubseteq^u_{nt} \mathcal{F}$, by Corollary 4.26. □

As a concrete example, the functionalities $\mathcal{F}$ and $\mathcal{G}$ are incomparable if we let $\mathcal{F}$ be the XOR function (or any function of the form $\mathcal{F}(x, y) = (x, y)$) and $\mathcal{G}$ be any of the $\mathcal{G}_i$ functionalities defined in Corollary 4.28.

4.8 Conclusion & Open Problems

We have developed a powerful new tool for analyzing the complexity of 2-party SSFE functionalities, which unifies all of our impossibility results in a wide variety of security settings. The natural extensions of our work involve extending these tools to apply to a larger class of functionalities.

Non-uniquely decomposable functionalities. Our main technical tool crucially relies on unique decomposability of the target functionality. This restriction prevents us from proving impossibility results for securely realizing functionalities like a symmetric exchange $\mathcal{F}(x,y) = (x,y)$, which may have many decompositions (this class of functionalities is important in Chapter 5). However, it seems likely that even for non-uniquely decomposable functionalities, the decomposition structure can be used to construct conceptually simple attacks on its protocols, in a manner similar to Theorem 4.12. For example, in any execution of any protocol for $\mathcal{F}$, the parties must (at least intuitively) reveal information about their inputs in the same order as *some* decomposition of $\mathcal{F}$.

We conjecture that among all decomposable SSFE functionalities $\mathcal{F}$ and $\mathcal{G}$, there is a simple combinatorial characterization (involving the decomposition structure) for when $\mathcal{F} \sqsubseteq^u_{nt} \mathcal{G}$. For instance, let $\mathcal{F}_n : \{1,\ldots,n\}^2 \to \{1,\ldots,n\}^2$ be the simultaneous exchange function defined by $\mathcal{F}_n(x,y) = (x,y)$. We conjecture that these functionalities form an infinite hierarchy of complexity, so that $\mathcal{F}_i \sqsubseteq^u_{nt} \mathcal{F}_j$ if and only if $i \leq j$. However, unlike the complexity hierarchy that is known in the uniquely decomposable case (Corollary 4.28), all of these functions have decompositions of depth 2. Our current techniques are only able to prove complexity separations based on decomposition depth (Corollary 4.26), so any complete combinatorial characterization of $\sqsubseteq^u_{nt}$ will likely have to take into account more details of the decomposition structure.

Nothing is known about the class of functionalities which are neither complete nor passively-realizable, other than the simple observation that this class is non-empty. One such 2-party SSFE is the "spiral" function $\begin{bmatrix} 0 & 0 & 1 \\ 3 & 4 & 1 \\ 3 & 2 & 2 \end{bmatrix}$. Clearly, completely new techniques are needed to show any results of the form $\mathcal{F} \not\sqsubseteq^u_{nt} \mathcal{G}$ among functionalities $\mathcal{F}$ and $\mathcal{G}$ in this class. However, since this class contains functionalities possibly more complex than any of the classes studied so far, any better understanding of the class will shed valuable insight on the nature of cryptographic complexity.

Non-SSFE functionalities. Our characterizations of passive- and standalone-realizability are known to generalize to the case of *multi-party* (non-symmetric) SFE functionalities [66], so it may be possible to extend our main technical tools in a similar way. However, our current proof techniques strongly rely on the fact that the SFE being computed has symmetric output; we use a normal form in which the outcome of the protocol is a public function of the transcript. Also, the communication channel (i.e., broadcast or point-to-point) makes a significant difference for secure realizability of multi-party functionalities.

There is no known characterization of passive- or standalone-realizability for *randomized* functionalities. Once randomized functionalities are better understood in the passive security setting, it is likely that our techniques can be updated, since we currently leverage decomposability results of the passive setting in all of our impossibility proofs.

In Chapter 5, we develop some tools for studying *reactive* functionalities, though it is not clear whether these automata-theoretic tools can be easily incorporated into our current approach.

Commitment compiler. We have constructed a general purpose "compiler" which converts a passive-secure protocol into a UC-secure protocol in the $\mathcal{F}_{\text{COM}}$-hybrid setting. We used this compiler to show that among 2-party SSFE functionalities, $\mathcal{F} \sqsubseteq^u_{nt} \mathcal{F}_{\text{COM}}$ if and only if $\mathcal{F}$ has a passive-secure protocol. We leave open the question of whether this characterization generalizes to larger classes of functionalities. In particular, our current approach seems severely limited to the case when $\mathcal{F}$ is non-reactive.

Furthermore, the compiler's overhead is exponential in the randomness complexity of a protocol. A more practical compiler would have polynomial overhead for all protocols, deterministic and randomized.

CHAPTER 5

Polynomial-Time Cryptographic Complexity

5.1 Overview

We have seen that the $\sqsubseteq^u_{nt}$ reduction can distinguish infinitely many levels of cryptographic complexity among MPC functionalities. Naturally, one might wonder whether this is also the case for the $\sqsubseteq^p_{nt}$ reduction, which is slightly weaker, yet still strong, being also based on UC security. In this section, we show that, perhaps surprisingly, there are only two levels of cryptographic complexity under the $\sqsubseteq^p_{nt}$ reduction.

Zero-one laws. Under any cryptographic reduction $\sqsubseteq$, there are always at least two distinct complexity levels:

- Trivial functionalities: Those which can be securely realized without access to any ideal functionality. Generally, the security model provides some kind of communication channel that protocols can use "for free", so the trivial functionalities are those which reduce to the communication channel. In our case, trivial functionalities are those which can be securely realized using $\mathcal{F}_{\text{PVT}}$.

- Complete functionalities: Those to which every functionality can be reduced. Technically, however, some functionalities can not have a secure protocol in *any* reasonable setting. For instance, consider the functionality which announces the list of corrupted parties — this is valid behavior, since the functionality is provided with this information. However, no protocol in any setting can securely realize this functionality, since if an adversary is passive (i.e., follows the protocol), the protocol must erroneously output that no one is corrupt.

  In this work, we will follow the convention of Canetti et al. [23], who call a functionality $\mathcal{F}$ complete if $\mathcal{G} \sqsubseteq \mathcal{F}$ for all *well-formed* functionalities $\mathcal{G}$. A functionality is well-formed if its behavior does not depend on its knowledge of which parties are corrupt. We directly use the completeness of $\mathcal{F}_{\text{COM}}$ proven by Canetti et el., and thus consider completeness only with respect to well-formed functionalities.

Within each of these two classes, all functionalities are equivalent under $\sqsubseteq$ reductions. In the language of computational complexity, these classes each constitute a *degree* of the reduction. We say that a class of functionalities has a *zero-one law* under a reduction $\sqsubseteq$ if every functionality in that class is either trivial or complete with respect to $\sqsubseteq$ reductions.

Zero-one laws are known to exist in several security settings for MPC. However, we note that none of them are applicable to the UC setting, and all of them are restricted to SFE functionalities:

- For 2-party SFE with one-sided output (one party's output is a constant function) against active, computationally unbounded adversaries [63].

- For 2-party *randomized* SFE with one-sided output against passive, computationally unbounded adversaries [63].

- For 2-party SFE with one-sided output against passive adversaries; both computationally bounded and unbounded cases, shown by Beimel, Malkin, and Micali [5]. They also consider a notion of active security for computationally bounded setting, and extends their dichotomy using a stronger notion of realizability and a weaker, non-black-box notion of completeness; this result draws the line between realizability and completeness differently from the above result of Kilian [63], and uses techniques not applicable in the UC setting.

- For 2-party SFE with one-sided output in the computationally bounded setting, where the size of the function can depend on the security parameter, shown by Harnik et al. [50]. They show characterizations for completeness and realizability which are only *nearly* complementary, reflecting the gap between functions which are efficiently invertible and one-way. Their approach uses techniques not applicable in the UC setting.

- For 2-party SFE with *symmetric, boolean* output against passive, computationally unbounded adversaries [64].

5.1.1 Our Results

In this chapter we prove a zero-one law for the $\sqsubseteq^p_{nt}$ reduction. We note that if P = NP, then all of our results for the $\sqsubseteq^u_{nt}$ reduction from Chapter 4 also apply to the $\sqsubseteq^p_{nt}$ reduction. Thus any result that shows such a significant difference between these two reductions must necessarily rely on some computational assumption. We consider the following assumption:

Assumption 5.1. *The oblivious transfer functionality $\mathcal{F}_{\text{OT}}$ has a passive-secure protocol (using $\mathcal{F}_{\text{PVT}}$) in the PPT setting.*

This assumption is well-studied, and known to be implied by many concrete cryptographic assumptions, such as the Decisional Diffie-Hellman assumption [76] and the existence of enhanced trapdoor permutations [40]. Our main theorem is that Assumption 5.1 is necessary and sufficient for a zero-one law under the $\sqsubseteq^p_{nt}$ reduction:

Theorem 5.2 (Main Theorem)**.** *There is zero-one law for 2-party deterministic finite functionalities (Definition 2.7) with respect to the $\sqsubseteq^p_{nt}$ reduction if and only if Assumption 5.1 is true.*

We emphasize that implicit in this result is the first characterization of secure realizability for a class of arbitrary *reactive* functionalities in the UC setting, and the first characterization of completeness for arbitrary reactive functionalities in *any* security model.

Our hardness assumption. It is relatively straight-forward to show that Assumption 5.1 is a necessary condition for the zero-one law. Thus to complete the proof of our main theorem, we must construct protocols which demonstrate the completeness of every non-trivial DFF, using the minimal hardness assumption Assumption 5.1.

Some of our protocol constructions rely on statistically binding commitment schemes, pseudorandom generators, and zero-knowledge and witness-indistinguishable proofs (of knowledge) for NP languages. All of these primitives exist assuming the existence of one-way functions [75, 51, 40]. One-way functions, in turn, are implied by Assumption 5.1 [48].

Targeting (extractable) commitment. We first observe that the commitment functionality $\mathcal{F}_{\text{COM}}$ is complete under the $\sqsubseteq^p_{nt}$ reduction (for well-formed functionalities), assuming Assumption 5.1, from the work of Canetti et al. [23] Thus, to show that $\mathcal{F}$ is complete, it suffices to show that $\mathcal{F}_{\text{COM}} \sqsubseteq^p_{nt} \mathcal{F}$.

The simulator for a UC-secure commitment protocol has two main tasks, which highlight the technical challenges involved: First, the simulator for a corrupt sender must extract the adversary's input during the commit phase, so that it can commit to the appropriate bit in the ideal interaction. Second, the simulator for a corrupt receiver must give an equivocable commitment in the reveal phase, which it must be able to later open to any value in the reveal phase.

The main challenge in our work is achieving extractability using very simple (but still non-trivial) functionalities, and without relying on trapdoor permutations or any other assumptions not known to be implied by Assumption 5.1. Previous UC-secure protocols for commitment have used stronger computational assumptions (trapdoor permutations and dense cryptosystems in [23] and dual-mode cryptosystems in [82]) to achieve extractability; we use new protocol techniques to achieve extractability from a non-trapdoor-based assumption.

To focus on the extractability requirement, we define an intermediate ideal "extractable commitment" functionality $\mathcal{F}_{\text{EXT-COM}}$. This functionality formalizes the requirement that a protocol has a standalone-secure hiding guarantee, but admits a straight-line extraction in the UC framework. Then we show that $\mathcal{F}_{\text{COM}} \sqsubseteq^p_{nt} \mathcal{F}_{\text{EXT-COM}}$ (Lemma 5.26); thus to show that a functionality is complete, it suffices to use that functionality to construct an "extractable commitment" protocol (i.e., to show $\mathcal{F}_{\text{EXT-COM}} \sqsubseteq^p_{nt} \mathcal{F}$). In our $\mathcal{F}_{\text{COM}}$ protocol in the $\mathcal{F}_{\text{EXT-COM}}$-hybrid setting, we use witness-indistinguishable proofs and apply an idea similar in spirit to $\binom{2}{1}$-commitments [78].

Classifying reactive functionalities. Since ours is the first work to consider completeness for arbitrary *reactive* functionalities, one of our important contributions is developing new technical tools for reasoning about them. We model reactive functionalities as finite-state automata, and we develop an automata-theoretic characterization of important cryptographic properties.

Our characterization allows us to identify exactly what behaviors make a functionality non-trivial. First, a reactive functionality can be non-trivial if in some state its input/output behavior defines non-trivial SFE behavior. Second, a reactive functionality can be non-trivial if it uses its memory to store hidden information about the parties' inputs, which later influences the functionality's behavior. These intuitive properties, which are challenging to define formally and combinatorially, completely characterize non-triviality, as demonstrated in the following result:

Theorem. *Let $\mathcal{F}$ be a DFF. Then $\mathcal{F}$ is non-trivial if and only if $\mathcal{F}_{\text{COM}} \sqsubseteq_{nt} \mathcal{F}$ or $\mathcal{G} \sqsubseteq_{nt} \mathcal{F}$ for some non-trivial SFE functionality $\mathcal{G}$.*

In other words, a functionality is non-trivial if and only if it uses its memory in a non-trivial way (which is the essence of the commitment functionality $\mathcal{F}_{\text{COM}}$) or it gives input/output in a non-trivial way. We note that our combinatorial definitions of these properties also provide a completely automata-theoretic characterization of realizability for arbitrary DFFs. Indeed, the protocols constructed in this proof are unconditionally secure, so the characterization of realizability also holds in the unbounded setting.

Classifying SFE functionalities. Theorem 5.14 effectively reduces the question of a zero-one law for DFFs to the simpler question of a zero-one law for SFE functionalities.

We already have a combinatorial characterization of triviality for SFE functionalities from Chapter 3, as well as a combinatorial characterization for completeness in the unbounded setting from [65] (relying on the completeness of oblivious transfer [61, 56]). We give the following purely combinatorial classification of non-trivial SFE functionalities:

Lemma. *Let $\mathcal{F}$ be a non-trivial 2-party SFE functionality. Then either $\mathcal{F}_{\text{OT}} \sqsubseteq_{nt} \mathcal{F}$, or $\mathcal{F}_{\text{CC}} \sqsubseteq_{nt} \mathcal{F}$, or $\mathcal{F}_{\text{COIN}} \sqsubseteq_{nt} \mathcal{F}$.*

The protocol for $\mathcal{F}_{\text{OT}}$ is due to Kraschewski and Müller-Quade [65]. $\mathcal{F}_{\text{COIN}}$ is the coin-tossing functionality, and $\mathcal{F}_{\text{CC}}$ is the symmetric-output SFE functionality whose function table is $\left[\begin{smallmatrix} 0 & 0 \\ 1 & 2 \end{smallmatrix}\right]$. We interpret $\mathcal{F}_{\text{CC}}$ as a kind of cut-and-choose functionality — Alice can choose to learn Bob's input by choosing the bottom row input, or choose not to learn it by choosing the top row input; Bob learns whether or not Alice has learned his input.

Obtaining extractable commitment. To complete the picture, we must finally show extractable commitment protocols in the $\mathcal{F}_{\text{CC}}$-hybrid and $\mathcal{F}_{\text{COIN}}$-hybrid settings.[1]

Our extractable commitment protocol in the $\mathcal{F}_{\text{CC}}$-hybrid setting (Lemma 5.27) achieves extractability by performing a cut-and-choose on an error-correcting encoding of a random string chosen by the sender, so that the receiver can (adaptively) choose to see a small fraction of the bits of the codeword. The commitment will use a deterministically extracted bit that the receiver has no information about, as a random mask. But the simulator obtains the entire (possibly noisy, if Alice is corrupt) codeword, which it can use to extract the committed bit.

Our extractable commitment protocol in the $\mathcal{F}_{\text{COIN}}$-hybrid setting directly uses a passive-secure OT protocol and "compiles" it using (standalone-secure) zero-knowledge and witness-indistinguishable proofs in a variant of the GMW paradigm [41]. The sender gives a standalone commitment to some random bits, half of which the receiver picks up using the compiled OT protocol. Then the parties use $\mathcal{F}_{\text{COIN}}$ to publicly agree on a random half of the bits to actually use as the random mask in the commitment (Lemma 5.28). The simulator can arrange for these choices to coincide, so that it learns the entire random mask and can extract.

Implications of a zero-one law. Composable security is a very desirable cryptographic property. However, the existing models for composable security have proven to be prohibitively strong. Very few functionalities admit composably secure protocols in these models, and thus much work has focused on tweaking the models to maintain a protocol composition theorem while allowing more functionalities to be securely realizable. The most common augmentation of the model is to assume the existence of a "setup" functionality like a common random string. A zero-one law implies that all such augmentations of the model are either trivial (they do not add any expressivity) or else all-powerful (they admit secure protocols for every task).

Indeed, many previously considered setup functionalities were known to be consistent with a zero-one characterization. For example, starting with Canetti et al. [23], many setup functionalities are known to be complete, while Kidron and Lindell [60] showed that the known impossibility results in the plain UC framework also apply in the presences of several setup functionalities.

We view a zero-one law as strong evidence that *the notion of composable security against PPT adversaries is robust*. Indeed, to the best of our knowledge, all existing modifications to the UC framework (not just those involving simply adding a setup functionality) follow a zero-one characterization, giving more evidence for the robustness of composable security. For example, relaxing the composability requirement to *self-composability* yields an equivalent model to general composability [70]. Bounding network latency [58] or allowing superpolynomial-time simulation [87] allows secure protocols for all functionalities.

[1] Canetti et al. [23] gave a secure commitment protocol in the $\mathcal{F}_{\text{COIN}}$-hybrid model, which is secure even against adaptive corruption, but relied on enhanced trapdoor permutations and dense cryptosystems; we develop a secure commitment protocol in the $\mathcal{F}_{\text{COIN}}$-hybrid setting (against static corruption) using the minimal hardness assumption.

5.2 Necessity of the Cryptographic Assumption

We first show that Assumption 5.1 is necessary for our desired zero-one to hold.

Lemma 5.3. *Let $\mathcal{F}$ be any 2-party SFE functionality that is passively realizable but not UC-realizable (note that such functionalities exist and that these characterizations are unconditional; see Chapter 4). If $\mathcal{F}$ is complete for the $\sqsubseteq^p_{nt}$ reduction, then Assumption 5.1 is true.*

Proof. If $\mathcal{F}$ is complete, then $\mathcal{F}_{\text{OT}} \sqsubseteq^p_{nt} \mathcal{F}$ via some protocol π. We claim that π is also a *passive-secure* protocol. Supposing this is true, we can compose π (a protocol in the $\mathcal{F}$-hybrid setting) with the passive-secure protocol for $\mathcal{F}$ (in the $\mathcal{F}_{\text{PVT}}$-hybrid setting) to obtain a passive-secure protocol for $\mathcal{F}_{\text{OT}}$ to establish Assumption 5.1.

Thus it suffices to show that π is passive-secure. We will use an argument similar to Lemma 4.3 to show that the simulator for π without loss of generality maps passive real-world adversaries to passive ideal-world adversaries. Consider an environment that chooses random inputs for all parties (x_0, x_1 for the sender and b for the receiver) and outputs 1 if the receiver reports output x_b. This environment outputs 1 with probability negligibly close to 1 in the real world by the correctness of the protocol.

Suppose $\mathcal{A}$ is a passive adversary that corrupts the sender. Whenever the corresponding simulator $\mathcal{S}$ does not send (x_0, x_1) to $\mathcal{F}_{\text{OT}}$ in the ideal world, the environment outputs 0 with probability 1/2. Thus with overwhelming probability by the protocol's security guarantee, $\mathcal{S}$ acts passively by sending the correct inputs. Without loss of generality, $\mathcal{S}$ can be made passive with probability 1.

In the other case, suppose $\mathcal{A}$ is a passive adversary that corrupts the receiver. Whenever the corresponding simulator $\mathcal{S}$ does not send b to $\mathcal{F}_{\text{OT}}$ in the ideal world, its view is independent of the correct output x_b, so the environment outputs 0 with probability 1/2. Thus with overwhelming probability, $\mathcal{S}$ acts passively by sending the correct inputs and reporting the output from $\mathcal{F}_{\text{OT}}$. Again, $\mathcal{S}$ can be made passive with probability 1 without loss of generality. □

5.3 Classifying Reactive Functionalities

Our first step is to develop an "automata-theoretic" language to reason about arbitrary reactive functionalities.

5.3.1 Dominating Inputs

In arguing security, it is often convenient for Alice to assume that Bob will supply an input that is the "worst possible" for Alice, among all inputs that achieve the same effect. Towards that end, we develop the notion of *dominating inputs* to formally define when one input x "achieves the same effect" as another input x', in the context of a reactive functionality. Intuitively, this happens when every behavior that can be induced by sending x at a certain point can also be induced by sending x' instead, and thereafter appropriately translating subsequent inputs and outputs. More formally:

Definition 5.4. *Let $\mathcal{F}$ be a finite functionality, and let $x, x' \in X$ be inputs for Alice. We say that x* dominates *x' in the first round of $\mathcal{F}$, and write $x \geq_A x'$, if there is a secure protocol for $\mathcal{F}$ in the $\mathcal{F}$-hybrid setting, where the protocol for Bob is to run the dummy protocol (as Bob), and the protocol for Alice has the property that whenever the environment provides input x' for Alice in the first round, the protocol instead sends x to the functionality in the first round.*

We define domination for Bob inputs analogously, with the roles of Alice and Bob reversed. Without loss of generality, the secure protocol from the definition above may be just the dummy protocol, except possibly when the environment provides x' as Alice's first input. In this case, the definition requires that any behavior of $\mathcal{F}$ that is possible when Alice uses x' as her first input can also be induced *in an online fashion* by using x as her first input (and subsequently translating inputs/outputs according to some strategy).

Note that domination is trivially reflexive, and due to the universal composition theorem, it is also transitive. Also note that if x dominates x', then both x and x' must induce the same output for Bob in the first round, regardless of Bob's input.

Combinatorial characterization. We now show that there is also an alternative characterization of dominating inputs that is purely combinatorial. The previous definition in terms of secure protocols is more intuitive, but the combinatorial criteria will be useful in proving Lemma 5.8, which is crucially used in Theorem 5.14.

Definition 5.5. *Let $\mathcal{F}$ be a DFF, $S \subseteq Q^2$, let $x, x' \in X$, and let z be a possible output of f_A. We define:*

$$\text{next}(S, x, x', z) = \left\{ \big(\delta(q, x, y), \delta(q', x', y)\big) \;\middle|\; \exists (q, q') \in S, y \in Y : f_A(q, x, y) = z \right\}.$$

The intuition behind this definition is as follows. Suppose that in some protocol that uses $\mathcal{F}$, Alice has received inputs $x'_1, x'_2, \ldots$ from the environment but has instead sent $x_1, x_2, \ldots$ to $\mathcal{F}$. Suppose Alice is keeping track of S, the set of pairs (q, q'), such that:

- There is a sequence of inputs for Bob, $y_1, y_2, \ldots$, such that Alice's view of $\mathcal{F}$ is consistent with $\mathcal{F}$'s behavior on input sequence $(x'_1, y_1), (x'_2, y_2), \ldots$
- The input sequence $(x_1, y_1), (x_2, y_2), \ldots$ would put $\mathcal{F}$ in state q.
- The input sequence $(x'_1, y_1), (x'_2, y_2), \ldots$ would put $\mathcal{F}$ in state q'.

Then $\text{next}(S, x, x', r)$ defines the subsequent value of S if the environment then provides input x' but the protocol instead sends x to $\mathcal{F}$ and receives output z.

Definition 5.6. *Let $\mathcal{F}$ be a DFF, $S \subseteq Q^2$, and $x, x' \in X$. We say that (x, x')* is good for *S if the following are true:*

1. *For all $(q, q') \in S$, we have $f_B(q, x, \cdot) \equiv f_B(q', x', \cdot)$,*
2. *For all outputs z, we have $\big|\{f_A(q', x', y) \mid \exists (q, q') \in S, y \in Y$ such that $f_A(q, x, y) = z\}\big| = 1$,*
3. *For all outputs z and all $\overline{x}' \in X$, there exists $\overline{x} \in X$ such that $(\overline{x}, \overline{x}')$ is good for* $\text{next}(S, x, x', z)$.

Intuitively, suppose Alice has been sending different inputs to $\mathcal{F}$ than requested by the environment, but is trying to make the behavior of $\mathcal{F}$ reflect the environment's requests. If S represents Alice's knowledge about $\mathcal{F}$'s state so far (as defined above), and S is not good for x, x', then Alice has a chance of being caught in the future if in the next round the environment asks her to send x but she sends x' instead.

In case (1), there is a chance (depending on Bob's sequence of inputs) that Alice may induce the wrong output for Bob in this round. In case (2), Alice might send x to $\mathcal{F}$ and get response z as the response, but this new view might be consistent with at least 2 states which would require Alice to send conflicting outputs to the environment. In case (3), Alice may be able to induce correct outputs in this round, but she has a chance of being caught in the next round if the environment happens to provide input $\overline{x}'$.

Lemma 5.7. *$x \geq_A x'$ if and only if (x, x') is good for $\{(q_0, q_0)\}$.*

Proof. ($\Leftarrow$) Suppose (x, x') is good for $\{(q_0, q_0)\}$. We must describe a strategy for Alice to send x in the first round, but make it appear as if she had sent x' and is running the dummy protocol. Without loss of generality, suppose the environment internally simulates an instance of $\mathcal{F}$, with the inputs of its choice, and compares the parties' outputs with the expected outputs from this simulated instance of $\mathcal{F}$.

Then the protocol for Alice is to maintain a state of knowledge S according to her view, as above, starting with $S = \{(q_0, q_0)\}$. She maintains the following invariants:

- For all $\overline{x}' \in X$ that the environment might supply in the next round, there is some $\overline{x} \in X$ such that $(\overline{x}, \overline{x}')$ is good for S.
- If the external instance of $\mathcal{F}$ is in state q and the environment's internally simulated instance of $\mathcal{F}$ is in state q', then $(q, q') \in S$.

The claim is true for the base case of $S = \{(q_0, q_0)\}$, since the environment will send x' in the first round, and (x, x') is good for S.

The protocol proceeds as follows: If the environment provides input $\overline{x}'$ for Alice, then Alice sends input $\overline{x}$ to $\mathcal{F}$ such that $(\overline{x}, \overline{x}')$ is good for S. Such an $\overline{x}$ must exist by the inductive hypothesis. Then we have:

- Bob reports the correct output in this round, since his output is $f_B(q, \overline{x}, y)$, and the environment is expecting $f_B(q', \overline{x}', y)$, and $f_B(q, \overline{x}, \cdot) \equiv f_B(q', \overline{x}', \cdot)$ from case (1) of Definition 5.6.
- Alice receives input $z = f_A(q, \overline{x}, y)$, and the environment is expecting $z' = f_A(q', \overline{x}', y)$. By case (2) of Definition 5.6, given S, $\overline{x}$, $\overline{x}'$, and z, Alice can compute a singleton set which contains z', so she reports this output to the environment.
- Alice updates $S \leftarrow \text{next}(S, \overline{x}, \overline{x}', z)$. By case (3) of Definition 5.6 and the definition of next$(\cdot)$, the inductive invariants are maintained for the next round.

($\Rightarrow$) Assume that (x, x') is not good for $\{(q_0, q_0)\}$, and consider any Alice protocol that replaces x' by x in the first round. It suffices to construct an environment that successfully distinguishes this interaction from an interaction in which Alice uses the dummy protocol.

Let n be the minimum number of times that case (3) of Definition 5.6 needs to be applied to show that (x, x') is not good for $\{(q_0, q_0)\}$. We note that n is always at most $m = 2^{|Q|^2}|X|^2$, a constant.

We will construct $\mathcal{Z}_0$, which sends x' to Alice in the first round, and otherwise sends randomly chosen inputs, for a total of m rounds. As usual, it also internally simulates an instance of $\mathcal{F}$, to which it sends the inputs that it has chosen for Alice and Bob. $\mathcal{Z}_0$ outputs 1 if Alice and Bob's outputs always match that of its simulated instance of $\mathcal{F}$, and 0 otherwise.

Clearly $\mathcal{Z}_0$ outputs 1 with probability 1 when both parties run the dummy protocol. It suffices to show that when Alice runs a protocol which in the first round sends x instead of x', the environment $\mathcal{Z}_0$ outputs 0 with at least some constant probability. We will prove via induction that $\mathcal{Z}_0$ outputs 0 with probability at least $2(|Y||X|)^{-n}$, where n is defined as above. Let q_k be the state of the external instance of $\mathcal{F}$ after k rounds, and q'_k be the state of the internally simulated instance of $\mathcal{F}$ after k rounds. As before, we let $S_k \subseteq Q^2$ denote the set of pairs (q, q') that are consistent with Alice's view after k rounds.

We first claim that $\Pr[(q_k, q'_k) = (q, q') \mid (q, q') \in S_k] \geq |Y|^{-k}$. In other words, after k rounds of interacting with $\mathcal{F}$, every $(q, q') \in S_k$ has some constant probability of being the "correct" pair, from Alice's point of view. The claim is trivially true for $k = 0$. For the inductive step, observe that by the definition of next$(\cdot)$, every $(p, p') \in S_{k+1}$ is in the set owing to at least one particular predecessor $(q, q') \in S_k$ and Bob input $y \in Y$. Thus the probability that (p, p') is correct is at least the probability that the predecessor (q, q') is correct, and the appropriate $y \in Y$ is chosen, which is $|Y|^{-k-1}$ as desired.

We will prove the claim about $\mathcal{Z}_0$'s distinguishing probability inductively in n. We will maintain the invariant that (x_{k+1}, x'_{k+1}) is not good for S_k, which is true in the base case.

Suppose (x_{k+1}, x'_{k+1}) is not good for S_k due to case (1) of Definition 5.6. Then with probability at least $|Y|^{-k}$, the two instances of $\mathcal{F}$ are in the "bad" states (q, q') from the negation of case (1). Conditioned on this event, then with probability $1/|Y|$, the environment chooses input y_k such that Bob's output and expected output disagree. The environment outputs 0 with probability at least $|Y|^{-k-1}$.

Suppose (x_{k+1}, x'_{k+1}) is not good for S_k due to case (2) of Definition 5.6. Then there are two triples (q, q', y) such that if the two instances of $\mathcal{F}$ are in states q and q' respectively, and Bob's input is chosen as y, then Alice's output is the same, but her expected output is different. The correct value of (q_k, q'_k, y_k) is indeed one of these triples (q, q', y) with probability at least $2/|Y|^{k+1}$, and conditioned on this being the case, Alice's reported output is incorrect with probability 1/2. Overall, the environment outputs 0 with probability at least $|Y|^{-k-1}$.

Suppose (x_{k+1}, x'_{k+1}) is not good for S_k due to case (3) of Definition 5.6. Then with probability at least $1/|X||Y|$, the environment chooses x'_{k+2} and y_{k+2} to be among the "bad" ones so that Alice receives output z and for all x_{k+2}, (x_{k+2}, x'_{k+2}) is not good for next$(S_k, x_{k+1}, x'_{k+1}, z)$. We may condition on this event and apply the inductive hypothesis.

We see that the total probability that $\mathcal{Z}_0$ outputs 0 is at least $|X|^{-n}|Y|^{-2n}$. □

The first half of the above proof also immediately implies the following useful lemma:

Lemma 5.8. *Let $\mathcal{F}$ be a DFF. Then there is an environment $\mathcal{Z}_0$ with the following properties:*

- *$\mathcal{Z}_0$ sends a constant number of inputs to $\mathcal{F}$,*
- *$\mathcal{Z}_0$ always outputs 1 when interacting with two parties running the dummy protocol on an instance of $\mathcal{F}$,*
- *For every $x, x' \in X$, if $x \not\geq_A x'$, then $\mathcal{Z}_0$ has a constant probability of outputting 0 when interacting with an Alice protocol that sends x instead of x' in the first round.*

The $\mathcal{Z}_0$ in question is the environment that simply chooses random inputs, and compares the responses to the known, deterministic behavior of $\mathcal{F}$. From the proof of Lemma 5.7, we see that $\mathcal{Z}_0$ needs to execute for only m rounds, where m is a constant that depends only on the size of $\mathcal{F}$. The distinguishing probability of $\mathcal{Z}_0$ is at least $|X|^{-m}|Y|^{-2m}$, a constant.

5.3.2 Simple States & Safe Transitions

Definition 5.9. *Let $\mathcal{F}$ be a DFF, and let q be one of its states. We define $\mathcal{F}[q]$ as the functionality obtained by modifying $\mathcal{F}$ so that its start state is q.*

Definition 5.10. *Let $\mathcal{F}$ be a DFF, and let q be one of its states. We say that q is a* simple state *if:*

- *The input/output behavior of $\mathcal{F}$ at state q — $(f_A(q,\cdot,\cdot), f_B(q,\cdot,\cdot))$ — is a trivial SFE; and*
- *For all Alice inputs $x, x' \in X$ such that $f_B(q,x,\cdot) \equiv f_B(q,x',\cdot)$, there exists an Alice input $x^* \in X$ such that $x^* \geq_A x$ and $x^* \geq_A x'$ in $\mathcal{F}[q]$; and*
- *For all Bob inputs $y, y' \in Y$ such that $f_A(q,\cdot,y) \equiv f_A(q,\cdot,y')$, there exists a Bob input $y^* \in Y$ such that $y^* \geq_B y$ and $y^* \geq_B y'$ in $\mathcal{F}[q]$.*

Suppose q is a simple state. We write $x \overset{q}{\sim} x'$ if $f_B(q,x,\cdot) \equiv f_B(q,x',\cdot)$. The relation $\overset{q}{\sim}$ induces equivalence classes over X. When q is a simple state, then within each such equivalence class, there exists at least one input x^* which dominates all other members of its class. For each equivalence class, we arbitrarily pick a single such input x^* and call it a *master* input for state q. Similarly we define master inputs for Bob by exchanging the roles of Alice and Bob.

Definition 5.11. *Let $\mathcal{F}$ be a DFF, We say that a transition is* safe *if it leaves a simple state q on inputs (x,y), where x and y are both master inputs for state q.*

We define $r(\mathcal{F})$ to be the functionality which runs $\mathcal{F}$, except that in the first round only, it allows only safe *transitions to be taken. $r(\mathcal{F})$ can be written as a copy of $\mathcal{F}$ plus a new start state. The new start state of $r(\mathcal{F})$ duplicates all the safe transitions of $\mathcal{F}$'s start state.*

Observation 5.12. *If a safe transition was just taken in $\mathcal{F}$, then Alice (resp. Bob) can uniquely determine Bob's (resp. Alice's) input in the previous round and the current state of $\mathcal{F}$, given only the previous state of $\mathcal{F}$ and Alice's (resp. Bob's) input and output in the previous round.*

Proof. We will show that Alice has no uncertainty about which master input Bob used, thus no uncertainty about the resulting state of $\mathcal{F}$. If a safe transition was just taken from q, then q was a simple state and its associated SFE $(f_A(q,\cdot,\cdot), f_B(q,\cdot,\cdot))$ is trivial. Thus either $f_A(q,\cdot,\cdot)$ is insensitive to Bob's input, or $f_B(q,\cdot,\cdot)$ is insensitive to Alice's input.

If $f_A(q,\cdot,\cdot)$ is insensitive to Bob's input, then Bob has a single master input y for q (all of his inputs are in a single equivalence class under $\overset{q}{\sim}$). There is no uncertainty for Alice regarding which master input Bob used.

If $f_B(q,\cdot,\cdot)$ is insensitive to Alice's input, then let x^* be Alice's unique master input. If y, y' are distinct master inputs for Bob, then $f_A(q,\cdot,y) \not\equiv f_A(q,\cdot,y')$. In other words, $f_A(q,x,y) \neq f_A(q,x,y')$ for some s. Since $x^* \geq_A x$, we must have $f_A(q,x^*,y) \neq f_A(q,x^*,y')$, so Alice (who must have used input x^*) has no uncertainty about which master input Bob used. □

Lemma 5.13. *If the start state of $\mathcal{F}$ is simple, then $r(\mathcal{F}) \sqsubseteq_{nt} \mathcal{F} \sqsubseteq_{nt} r(\mathcal{F})$. Furthermore, if q is reachable from the start state of $\mathcal{F}$ via a safe transition, then $\mathcal{F}[q] \sqsubseteq_{nt} \mathcal{F}$.*

Proof. The protocol for $r(\mathcal{F}) \sqsubseteq_{nt} \mathcal{F}$ is the dummy protocol, since $r(\mathcal{F})$ implements simply a subset of the behavior of $\mathcal{F}$. Simulation is trivial unless in the first round, the corrupt party (say, Alice) sends an input x to $\mathcal{F}$ which is not a master input for q_0. The simulator must send the corresponding master input x^* (from the $\overset{q_0}{\sim}$ equivalence class of x) in the ideal world, and then it uses the translation protocol guaranteed by the definition of $x^* \geq_A x$ to provide a consistent view to Alice and induce correct outputs for Bob.

Similarly, the protocol for $\mathcal{F} \sqsubseteq_{nt} r(\mathcal{F})$ is simply the dual of the above protocol. On input x in the first round, Alice sends x^* to $r(\mathcal{F})$, where x^* is the master input from the $\overset{q_0}{\sim}$-equivalence class of x. Thereafter, Alice runs the protocol guaranteed by the fact that $x^* \geq_A x$. Bob's protocol is analogous. Simulation is a trivial dummy simulation, since any valid sequence of inputs to $r(\mathcal{F})$ in the real world also produces the same outcome in the $\mathcal{F}$-ideal world ($r(\mathcal{F})$ implements a subset of the behavior of $\mathcal{F}$).

Note that in $r(\mathcal{F})$, the added start state has no incoming transitions; thus $(r(\mathcal{F}))[q] = \mathcal{F}[q]$ if q is a state in $\mathcal{F}$. So to show $\mathcal{F}[q] \sqsubseteq_{nt} \mathcal{F}$, it suffices to show that $(r(\mathcal{F}))[q] \sqsubseteq_{nt} r(\mathcal{F})$. Suppose q is reachable in $\mathcal{F}$ from the start state via safe transition on master inputs x^*, y^*. The protocol for $\mathcal{F}[q]$ is for Alice and Bob to send x^* and y^* to $r(\mathcal{F})$, respectively, as a "preamble". Each party can determine with certainty, given their input and output in this preamble, whether $r(\mathcal{F})$ is in state q (since only safe transitions can be taken from the start state of $r(\mathcal{F})$). If the functionality is not in q, then the parties abort. Otherwise, the functionality is $r(\mathcal{F})$ in state q as desired, so the parties thereafter run the dummy protocol. Simulation is trivial – the simulator aborts if the corrupt party does not send its specified input (x^* or y^*) in the preamble; otherwise it runs a dummy simulation. □

5.3.3 Complete Characterization of DFFs

We now prove our main classification regarding reactive functionalities, which is a useful alternative characterization of secure realizability for DFFs. Interestingly, though this chapter focuses exclusively on the PPT setting, our characterization of DFFs in this section also applies in the unbounded setting. Our characterization is as follows:

Theorem 5.14. *Let $\mathcal{F}$ be a DFF. Then the following are equivalent:*

1. $\mathcal{F} \sqsubseteq_{nt} \mathcal{F}_{\text{PVT}}$
2. $\mathcal{F}_{\text{COM}} \not\sqsubseteq_{nt} \mathcal{F}$ *and* $\mathcal{G} \not\sqsubseteq_{nt} \mathcal{F}$ *for all non-trivial SFE functionalities* $\mathcal{G}$
3. *No non-simple state in* $\mathcal{F}$ *is reachable via a sequence of safe transitions from* $\mathcal{F}$*'s start state.*

First, this lemma implies that if $\mathcal{F}_{\text{COM}}$ is $\sqsubseteq^p_{nt}$-complete, and if there is a zero-one law for SFE functionalities, then there is a zero-one law for all DFFs. Thus we have reduced the proof of the zero-one law to the much simpler case of SFE. To prove Theorem 5.14, we construct two protocols in the following lemmas, both of which are unconditionally secure. Also, the definition of triviality for SFE functionalities is the same in both the PPT and unbounded settings. Thus, the lemma provides a complete characterization of secure realizability for DFFs in both settings. Finally, note that condition (3) of Theorem 5.14 can be expressed completely combinatorially (automata-theoretically) using Lemma 5.7, giving the first such alternate characterization of realizability for any large class of arbitrary reactive functionalities.

We have that (1) $\Rightarrow$ (2) of Theorem 5.14, by the fact that $\mathcal{F}_{\text{COM}}$ is non-trivial (in the unbounded and PPT settings). We prove (2) $\Rightarrow$ (3) and (3) $\Rightarrow$ (1) in the following two lemmas:

Lemma 5.15. *If a non-simple state in $\mathcal{F}$ is reachable via a sequence of safe transitions from $\mathcal{F}$'s start state, then either $\mathcal{F}_{\text{COM}} \sqsubseteq_{nt} \mathcal{F}$ or $\mathcal{G} \sqsubseteq_{nt} \mathcal{F}$ for some non-trivial SFE functionality $\mathcal{G}$.*

Proof. Without loss of generality (by Lemma 5.13) we assume that the start state of $\mathcal{F}$ is non-simple.

First, suppose the start state q_0 of $\mathcal{F}$ is non-simple because its input/output behavior in the first round is non-trivial. Then in the $\mathcal{F}$-hybrid setting we can easily securely realize the SFE functionality $\mathcal{G} = (f_A(q_0, \cdot, \cdot), f_B(q_0, \cdot, \cdot))$, by the simple dummy protocol. Even though $\mathcal{F}$ may keep in its memory arbitrary information about the first-round inputs, the information can never be accessed since honest parties never send inputs to $\mathcal{F}$ after its first round, and $\mathcal{F}$ waits for inputs from both parties before giving any output. Thus $\mathcal{G} \sqsubseteq_{nt} \mathcal{F}$.

Otherwise, assume that the input/output behavior in the first round is a trivial SFE, and that q_0 is non-simple for one of the other reasons in Definition 5.10. The two cases are symmetric, and we present the case where Alice can commit to Bob. Suppose there are Alice inputs $x^*_0, x^*_1 \in X$ such that $f_B(q_0, x^*_0, \cdot) \equiv f_B(q_0, x^*_1, \cdot)$, but for all $x \in X$, either $x \not\geq_A x^*_0$ or $x \not\geq_A x^*_1$. Intuitively, this means that $\mathcal{F}$ *binds* Alice to her choice between inputs x^*_0 and x^*_1 — there are behaviors of $\mathcal{F}$ possible when her first input is x^*_b, which are not possible when her first input is x^*_{1-b}. We formalize this intuition by using the first input round of $\mathcal{F}$ to let Alice commit a bit to Bob.

Recall the "complete" environment $\mathcal{Z}_0$ from Lemma 5.8, and suppose it runs for m rounds and has a distinguishing probability $p > 0$. Our protocol for $\mathcal{F}_{\text{COM}}$ is to instantiate $N = 2\lceil \log_{1-p} 0.5 \rceil \kappa = \Theta(\kappa)$ independent instances of $\mathcal{F}$, where κ is the security parameter. We will write $\mathcal{F}_i$ to refer to the ith instance of $\mathcal{F}$. The protocol is as follows:

1. (Commit phase, on Alice input (COMMIT, b), where $b \in \{0, 1\}$) Alice sends x^*_b to each $\mathcal{F}_i$. For each i, Bob sends a random $y_{i1} \in Y$ to $\mathcal{F}_i$ and waits for output $f_B(q_0, y_{i1}, x^*_0) = f_B(q_0, y_{i1}, x^*_1)$. If he receives a different input, he aborts. Otherwise, he outputs COMMITTED.
2. (Reveal phase, on Alice input REVEAL) Alice sends b to Bob. For each i, Alice sends her input/output view of $\mathcal{F}_i$ to Bob (x^*_b and the first-round response from $\mathcal{F}_i$). If any of these reported views involve Alice sending something other than x^*_b to $\mathcal{F}_i$, then Bob aborts. Otherwise, Bob sets $x_{i1} = x^*_b$ for all i.
3. For $j = 2$ to m:
   (a) Bob sends Alice a randomly chosen $x_{ij} \in X$. Alice sends x_{ij} to $\mathcal{F}_i$.
   (b) Bob sends a randomly chosen input $y_{ij} \in Y$ to $\mathcal{F}_i$.
   (c) For each i, Alice reports to Bob her output from $\mathcal{F}_i$ in this round.
4. If for any i, Alice's reported view or Bob's outputs from $\mathcal{F}_i$ does not match the (deterministic) behavior of $\mathcal{F}$ on input sequence $(x_{i1}, y_{i1}), (x_{i2}, y_{i2}), \ldots$, then Bob aborts. Otherwise, he outputs (REVEAL, b).

When Bob is corrupt, the simulation is to do the following for each i: When Bob sends y_{i1} to $\mathcal{F}$ in the commit phase, simulate $\mathcal{F}_i$'s response as $f_B(q_0, x_0^*, y_{i1}) = f_B(q_0, x_1^*, y_{i1})$. In the reveal phase, to open to a bit b, simulate that Alice sent Bob x_b^* and the view that is consistent with that input: $f_A(q_0, x_b^*, y_{i1})$. Maintain the corresponding state q_i of $\mathcal{F}_i$ after seeing inputs (x_b^*, y_{i1}). Then when Bob sends x_{ij} to Alice and y_{ij} to $\mathcal{F}_i$, simulate that $\mathcal{F}_i$ gave the correct output to Bob and that Alice reported back the correct output from $\mathcal{F}_i$ that is consistent with $\mathcal{F}$ receiving inputs x_{ij}, y_{ij} in state q_i. Each time, also update the state q_i according to those inputs. It is clear that the simulation is perfect.

When Alice is corrupt, the simulation is as follows: The simulator faithfully simulates each instance of $\mathcal{F}$ and the behavior of an honest Bob. If at any point, the simulated Bob aborts, then the simulation aborts. Suppose Alice sends $\tilde{x}_{i1}$ to each $\mathcal{F}_i$ in the commit phase, and that the simulation has not aborted at the end of the commit phase. If the majority of $\tilde{x}_{i1}$ values satisfy $\tilde{x}_{i1} \geq_A x_0^*$, then the simulator sends $(\text{COMMIT}, 0)$ to $\mathcal{F}_{\text{COM}}$; otherwise it sends $(\text{COMMIT}, 1)$. Note that by the properties of $\mathcal{F}$, each $\tilde{x}_{i1}$ cannot dominate both x_0^* and x_1^*. Let b be the bit that the simulator sent to $\mathcal{F}_{\text{COM}}$.

If the simulated Bob ever outputs (REVEAL, b), then the simulator sends REVEAL to $\mathcal{F}_{\text{COM}}$. The simulation is perfect except for the case where the simulated Bob outputs $(\text{REVEAL}, 1-b)$ (in this case, the real world interaction ends with Bob outputting $(\text{REVEAL}, 1-b)$, while the ideal world interaction aborts). We show that this event happens with negligible probability, and thus our overall simulation is statistically sound.

Suppose Alice sends $b' = 1 - b$ at the beginning of the reveal phase. Say that an instance $\mathcal{F}_i$ is *bad* if $\tilde{x}_{i1} \not\geq_A x_{1-b}^*$. Note that at least half of the instances of $\mathcal{F}_i$ are bad. When an instance $\mathcal{F}_i$ is bad, $\mathcal{Z}_0$ can distinguish with probability at least p between the cases of $\mathcal{F}$ receiving first input $\tilde{x}_{i1}$ and x_{1-b}^* from Alice. However, in each instance of $\mathcal{F}_i$, Bob is sending random inputs to Alice (who sent $\tilde{x}_{i1}$ as the first input to $\mathcal{F}_i$), sending random inputs himself to $\mathcal{F}_i$, obtaining his own output and Alice's reported output from $\mathcal{F}_i$ in an on-line fashion, and comparing the result to the known behavior of $\mathcal{F}$ (when x_{1-b}^* is the first input of Alice). This is exactly what $\mathcal{Z}_0$ does in the definition of $\tilde{x}_{i1} \geq_A x_{1-b}^*$, so Bob will detect an error with probability p in each bad instance. In the real world, Bob would accept in this reveal phase with probability at most $(1-p)^{-N/2} \leq 2^{-\kappa}$, which is negligible as desired. □

Lemma 5.16. *If no non-simple state in $\mathcal{F}$ is reachable via a sequence of safe transitions from $\mathcal{F}$'s start state, then $\mathcal{F}$ is trivial ($\mathcal{F} \sqsubseteq_{nt} \mathcal{F}_{\text{PVT}}$).*

Proof. We first define an intermediate functionality $R(\mathcal{F})$, which is $\mathcal{F}$ with all its non-safe transitions removed. We first observe that $R(\mathcal{F})$ is trivial (in fact, $R(\mathcal{F})$ is trivial for all $\mathcal{F}$). Only safe transitions may be taken in $R(\mathcal{F})$, thus both parties' views uniquely determine the state of $R(\mathcal{F})$. If the current state q was non-simple in $\mathcal{F}$, then q is a dead state in $R(\mathcal{F})$ and the protocol is trivial. Otherwise, note that restricting a simple state's transition function to its safe transitions preserves the triviality of the SFE round function. Thus the protocol's behavior when in state q is to simply evaluate a trivial SFE.

Next, to prove the main claim it suffices to show that $\mathcal{F} \sqsubseteq_{nt} R(\mathcal{F})$, since $R(\mathcal{F})$ is trivial. We prove a stronger claim; namely that if q is safely reachable (i.e., reachable from the start state by a sequence of safe transitions) in $\mathcal{F}$, then $\mathcal{F}[q] \sqsubseteq_{nt} (R(\mathcal{F}))[q]$. To prove this stronger claim, we construct a family of protocols $\hat{\pi}_q$, for every such q.

First, let π_q denote the protocol guaranteed by $\mathcal{F}[q] \sqsubseteq_{nt} r(\mathcal{F}[q])$ (Lemma 5.13). Then the protocol $\hat{\pi}_q$ is as follows:

1. Run π_q to interact with the functionality.
2. After the first round, we will have sent an input to the functionality and received an output. Assuming that the functionality was $(R(\mathcal{F}))[q]$, use the first round's input/output to determine the next state q' (Observation 5.12)
3. Continue running π_q, but hereafter, instead of letting it interact directly with the functionality, we recursively instantiate $\hat{\pi}_{q'}$. We let our π_q instance interface with $\hat{\pi}_{q'}$, which we let interact directly with the functionality.

The protocol is recursive, and after k rounds, must maintain a stack depth of size k. We prove by induction on k that $\hat{\pi}_q$ is a secure protocol for $\mathcal{F}[q]$ using $(R(\mathcal{F}))[q]$, against environments that run the protocol for $k \geq 0$ steps. The claim is trivially true for $k = 0$.

Note that simulation is trivial if either party is corrupt. Such an adversary is running the protocol interacting with $(R(\mathcal{F}))[q]$, which is a subset of the functionality $\mathcal{F}[q]$. Thus the simulator is a dummy simulator. It suffices to show that the output of the protocol is correct (indistinguishable from the ideal interaction) when both parties are honest.

In the first round, both parties are running π_q, interacting with $(R(\mathcal{F}))[q]$. Although π_q is designed to interact with $r(\mathcal{F}[q])$, the behavior of both these functionalities is identical in the first round (including the next-state function). Thus the first round of outputs is correct, by the security of π_q. For the same reason, step 2 of $\hat{\pi}_q$ correctly identifies the next state q' of $(R(\mathcal{F}))[q]$. Clearly $(R(\mathcal{F}))[q][q'] = (R(\mathcal{F}))[q']$, so after step 1 of the protocol, the functionality is identical to a fresh instantiation of $(R(\mathcal{F}))[q']$. At the same time, we also instantiate a fresh instance of $\hat{\pi}_{q'}$ to interact with this functionality. By the inductive hypothesis, hereafter π_q is interacting with an interface that is indistinguishable from an ideal interaction with $\mathcal{F}[q']$. However, an external functionality which behaves like

$R(\mathcal{F})[q]$ in the first round, then after transitioning to state q' behaves like $\mathcal{F}[q']$, is simply the functionality $r(\mathcal{F}[q])$. In other words, the entire protocol $\hat{\pi}_q$ is indistinguishable from running π_q on $r(\mathcal{F}[q])$. By definition of π_q, this is indistinguishable from an ideal interaction with $\mathcal{F}[q]$ itself. □

5.4 Classifying SFE Functionalities

2×2 minors. Our classification of SFE functionalities is combinatorial, and relies on identifying crucial 2×2 minors in the function table of the SFE.

Definition 5.17. *Let $\mathcal{F} = (f_A, f_B)$ be a 2-party SFE. We say that $\mathcal{F}$ has a* generalized CC-minor *at $\{x, x'\} \times \{y, y'\}$ if $\mathcal{F}$ has the following form:*

f_A	y	y'
x	a	a
x'	b	c

f_B	y	y'
x	h	j
x'	i	k

where $b \neq c$ and $h \neq i$ and $j \neq k$

or the symmetric condition with the roles of Alice and Bob exchanged.

In a generalized CC-minor, Alice chooses input x if she wants no information about Bob's input (y or y'), and chooses input x' if she wants to receive Bob's input. Bob learns which option Alice chose.

We call the following (symmetric-output) function the *symmetric cut-and-choose* function $\mathcal{F}_{\text{CC}}$:

$f_A = f_B$	0	1
0	0	0
1	1	2

It is the canonical function that contains a generalized CC-minor, and it plays an important role in our results.

Definition 5.18 ([65])**.** *Let $\mathcal{F} = (f_A, f_B)$ be a 2-party SFE. We say that $\mathcal{F}$ has a* generalized OR-minor *at $\{x, x'\} \times \{y, y'\}$ if $\mathcal{F}$ has the following form:*

f_A	y	y'
x	a	a
x'	b	c

f_B	y	y'
x	h	j
x'	h	k

where $b \neq c$ or $j \neq k$

Lemma 5.19 ([65])**.** *If $\mathcal{F}$ is a 2-party SFE functionality that contains a generalized* OR*-minor after removing all redundant inputs, then $\mathcal{F}$ is complete under $\sqsubseteq_{nt}$ reductions.*

In fact, the lemma proven by Kraschewski and Müller-Quade [65] is stronger, giving a complete characterization of completeness against computationally unbounded adversaries. That is, they prove that $\mathcal{F}$ is $\sqsubseteq_{nt}$-complete if and only if it contains a generalized OR-minor. We note that the protocol and simulator in their reduction are both efficient when $\mathcal{F}$ has constant size, but the security holds against computationally unbounded adversaries. Thus the completeness of these functions holds with respect to the $\sqsubseteq_{nt}^{u}$ and $\sqsubseteq_{nt}^{p}$ reductions. Of course, we will show that many other SFE functions are also complete under the $\sqsubseteq_{nt}^{p}$ reduction.

Definition 5.20. *$\mathcal{F} = (f_A, f_B)$ is a* symmetric exchange function *if $\mathcal{F}$ is isomorphic to the symmetric function $\mathcal{F}(x, y) = (x, y)$ for some input domain $X \times Y$.*

In a symmetric exchange function, each party learns the other party's input (the function's output also includes that party's own input, which it already knows). The cryptographic non-triviality of symmetric exchange functions is due to the fact that each party's input is chosen independently of the other party's input.

Combinatorially classifying non-reactive functionalities. We now prove the first characterization of SFE functionalities, using two technical lemmas:

Lemma 5.21. *Let $\mathcal{F} = (f_A, f_B)$ be an SSFE functionality. If $\mathcal{F}$ has a generalized* CC*-minor and no generalized* OR*-minor, then $\mathcal{F}_{\text{CC}} \sqsubseteq_{nt} \mathcal{F}$.*

Proof. Suppose $\mathcal{F}$ has a generalized CC-minor at $\{x, x'\} \times \{y, y'\}$. We will show that the protocol in which parties simply restrict their inputs to this 2×2 minor is a UC-secure protocol for computing that minor.[2] If the output from $\mathcal{F}$ is not consistent with the party's input and one of the two allowed inputs for the other party, then we abort. Since every generalized CC-minor is isomorphic to symmetric CC, the claim will be established.

Suppose the function table of $\mathcal{F}$ is as follows for the CC-minor:

f_A	y	y'
x	a	a
x'	b	c

f_B	y	y'
x	h	j
x'	i	k

where $b \neq c$, $h \neq i$, and $j \neq k$

We first consider the case where Alice is corrupt. If Alice provides input x or x', then the simulator also gives the same input in the ideal world, and returns the output to Alice. Otherwise, suppose Alice sends some other input x'':

f_A	y	y'
x	a	a
x'	b	c
x''	p	q

f_B	y	y'
x	h	j
x'	i	k
x''	r	s

where $b \neq c$, $h \neq i$, and $j \neq k$

We consider several cases, depending on the values of p, q, r, s:

- If $r \notin \{h, i\}$ and $s \notin \{j, k\}$, then honest Bob will always abort in the real world. The simulator sends input x' in the ideal world. The simulator can determine from its output whether Bob's input was y or y', and simulate either output p or q to Alice accordingly, and finally abort.
- If [$r = h$ and $s = j$ and $p \neq q$] or [$r = h$ and $s \neq j$] or [$r \neq h$ and $s = j$]. then $\{x, x''\} \times \{y, y'\}$ is a generalized OR-minor in $\mathcal{F}$. This is not possible in $\mathcal{F}$.
- If $r = h$ and $s = j$ and $p = q$, then the simulator sends input x in the ideal world. Bob receives the same output as in the real world, and the simulator gives Alice output $p = q$.
- If $r = i$ and $s = k$, then the simulator sends input x' in the ideal world. Bob receives the same output as in the real world, the simulator can determine from its output whether Bob's input was y or y', and simulate either output p or q to Bob, accordingly.
- If $r = i$ and $s \notin \{j, k\}$, then Bob will abort in the real world if his input was y'. The simulator sends input x' in the ideal world. If Bob's input is y, then Bob receives the same output as in the real world. The simulator can determine from its output whether Bob's input was y or y', and simulate either output p or q to Bob, accordingly. The simulator aborts if Bob's input is y'.

The definition of generalized CC-minor is symmetric with respect to y and y', so all other cases are obtained by symmetry.

The other case to consider is when Bob is corrupt. Similarly, the simulation is trivial when Bob uses either y or y'. Otherwise, suppose Bob uses some other input y'':

f_A	y	y'	y''
x	a	a	p
x'	b	c	q

f_B	y	y'	y''
x	h	j	r
x'	i	k	s

where $b \neq c$, $h \neq i$, and $j \neq k$

We again consider several cases:

- If $p \neq a$ and $q \notin \{b, c\}$, then similar to above, Alice will always abort in the real world. The simulator sends input y in the ideal world. It can determine from its output whether Alice's input was x or x', and simulate either output r or s to Bob accordingly, and finally abort.
- If $p = a$ and $q = b$, then the simulator sends input y in the ideal world. Alice receives the same output as in the real world. The simulator can determine from its output whether Alice's input was x or x', and simulate either output r or s to Bob, accordingly.
- If $p = a$ and $q = c$, the simulation is identical to the previous case, except the simulator sends input y' in the ideal world.

[2]Note that for an arbitrary $\mathcal{F}$, it does not necessarily follow that restricting inputs is a secure protocol for evaluating that minor, since adversaries may carefully choose other inputs to send to $\mathcal{F}$.

- If $p = a$ and $q \notin \{b, c\}$, then Alice aborts in the real world if her input was x'. The simulator sends input y in the ideal world. Alice receives the same output as in the real world if her input was x. The simulator can determine from its output whether Alice's input was x or x', and simulate either output r or s to Bob, accordingly. The simulator aborts if Alice's input was x'.
- If $p \neq a$ and $q = b$, then Alice aborts in the real world if her input was x. Similar to above, the simulator sends input y in the ideal world, simulates the appropriate output for Bob, and aborts if Alice's input was x.
- If $p \neq a$ and $q = c$, then the simulation is identical to the previous case, except the simulator sends input y' in the ideal world. □

Lemma 5.22. *If $\mathcal{F} = (f_A, f_B)$ has no generalized* CC*-minor and no generalized* OR*-minor, then $\mathcal{F}$ is a symmetric exchange function.*

Proof. If $\mathcal{F}$ is not a symmetric exchange function, then different inputs to $\mathcal{F}$ for (without loss of generality) Alice let her learn different distinctions among Bob's inputs. That is, for some x, x', y, y', we have: $f_A(x, y) = f_A(x, y')$ and $f_A(x', y) \neq f_A(x', y')$. Now no matter how f_B behaves on inputs $\{x, x'\} \times \{y, y'\}$, that 2×2 minor is either a generalized CC-minor or generalized OR-minor. □

Finally, we prove our main classification of SFE functionalities:

Lemma 5.23. *Let $\mathcal{F}$ be a non-trivial 2-party SFE functionality. Then either $\mathcal{F}_{\text{OT}} \sqsubseteq_{nt} \mathcal{F}$, or $\mathcal{F}_{\text{CC}} \sqsubseteq_{nt} \mathcal{F}$, or $\mathcal{F}_{\text{COIN}} \sqsubseteq_{nt} \mathcal{F}$.*

Proof. If $\mathcal{F}$ contains no generalized OR-minor, but contains a generalized CC-minor, then $\mathcal{F}_{\text{CC}} \sqsubseteq_{nt} \mathcal{F}$ (Lemma 5.21). Otherwise, if $\mathcal{F}$ contains neither a generalized CC-minor nor a generalized OR-minor, then $\mathcal{F}$ is a symmetric exchange function (Lemma 5.22). Suppose $\mathcal{F}$ is (isomorphic to) the symmetric function $\mathcal{F}(x, y) = (x, y)$, where $\mathcal{F}$ has input domain $X \times Y$. If $|X| < 2$ or $|Y| < 2$, then $\mathcal{F}$ is trivial, by Theorem 3.21.

Otherwise, suppose $|X| \geq 2$ and $|Y| \geq 2$. Then let $x_0 \neq x_1 \in X$ and $y_0 \neq y_1 \in Y$. A secure protocol for $\mathcal{F}_{\text{COIN}}$ in the $\mathcal{F}$-hybrid setting is for Alice to choose a random bit $a \leftarrow \{0, 1\}$ and Bob to choose a random bit $b \leftarrow \{0, 1\}$. They compute $\mathcal{F}(x_a, y_b) = (x_a, y_b)$ and use $a \oplus b$ as the output of the coin toss. Alice aborts if she observes that Bob did not use y_0 or y_1 as his input, and likewise Bob aborts if he observes that Alice did not use x_0 or x_1 as her input. It is straight-forward to see that this protocol is UC-secure. □

5.5 Extractable Commitment

In this section we show how full-fledged commitment can be realized using only a weak intermediate variant. We start off by introducing some convenient terminology.

Definition 5.24. *A protocol is a* syntactic commitment protocol *if:*

- *It is a two phase protocol between a sender and a receiver (using only plain communication channels).*
- *At the end of the first phase (commitment phase), the sender and the receiver output a transcript τ. Further the sender receives an output γ (which will be used for opening the commitment).*
- *In the reveal phase the sender sends a message γ to the receiver, who extracts an output value* $\mathsf{opening}(\tau, \gamma) \in \{0, 1\}^\kappa \cup \{\perp\}$.

In the above description, as is implicit in all our protocol specifications, the parties may choose to abort at any point in the protocol.

Definition 5.25. *We say that two syntactic commitment protocols (ω_L, ω_R) form* a pair of complementary statistically binding commitment protocols *if the following hold:*

- *ω_R is a statistically binding commitment scheme (with standalone security).*
- *In ω_L, at the end of the commitment phase the receiver outputs a string $z \in \{0, 1\}^\kappa$. If the the receiver is honest, it is only with negligible probability that there exists γ such that* $\mathsf{opening}(\tau, \gamma) \neq \perp$ *and* $\mathsf{opening}(\tau, \gamma) \neq z$.

We name the two parties Sender and Receiver. The functionality's behavior depends on who is corrupt.

If both Sender and Receiver are honest, the functionality behaves as follows:

1. (Commitment phase.) It accepts (COMMIT, x) from Sender. Then it internally simulates a session of ω_R (simulating both the sender and the receiver in ω_R), with the sender's input being x. It gives $(\text{TRANSCRIPT}, \tau, \gamma)$ to Sender and $(\text{COMMITTED}, \tau)$ to Receiver.
2. (Reveal phase.) On receiving the message REVEAL from Sender, it sends (REVEAL, x) to Receiver.

If Sender is corrupt and Receiver is honest, the functionality does the following:

1. (Commitment phase.) It runs the commitment phase of ω_L with Sender, playing the part of the receiver in ω_L, to obtain (τ, z). It sends $(\text{COMMITTED}, \tau)$ to Receiver and internally records z.
2. (Reveal phase.) It receives (REVEAL, γ) from Sender. If $\mathsf{opening}(\tau, \gamma) = z$, it sends (REVEAL, z) to Receiver.

If Sender is honest and Receiver is corrupt, the functionality does the following:

1. (Commitment phase.) It accepts (COMMIT, x) from Sender. Then it runs the commitment phase of ω_R with Receiver, playing the sender's role in ω_R, with x as input. It obtains the output (τ, γ) at the end of this phase, and sends $(\text{TRANSCRIPT}, \tau, \gamma)$ to Sender.
2. (Reveal phase.) it sends (γ, x) to Receiver.

(We do not define the behavior of the functionality when both Sender and Receiver are corrupt.)

Figure 5.1: Functionality $\mathcal{F}_{\text{EXT-COM}}^{(\omega_L,\omega_R)}$: Extractable commitment, parameterized by two syntactic commitment protocols ω_L and ω_R.

Note that ω_L by itself is not an interesting cryptographic goal, as the sender can simply send the committed string in the clear during the commitment phase; however, in defining $\mathcal{F}_{\text{EXT-COM}}^{(\omega_L,\omega_R)}$ below, we will require a single protocol to satisfy both the security guarantees.

We define the extractable commitment functionality $\mathcal{F}_{\text{EXT-COM}}^{(\omega_L,\omega_R)}$ in Figure 5.1. The functionality is parameterized by a pair of complementary statistically binding commitment protocols.

Our main result in this section is that extractable commitment can be used to securely realize full-fledged commitment:

Lemma 5.26. *If (ω_L, ω_R) form a pair of complementary statistically binding commitment protocols, then $\mathcal{F}_{\text{COM}} \sqsubseteq_{nt}^{p} \mathcal{F}_{\text{EXT-COM}}^{(\omega_L,\omega_R)}$.*

Proof. Our protocol uses additional witness-indistinguishable proofs, which are guaranteed to exist if standalone-secure commitment schemes exist. The protocol uses a "1-out-of-2 binding commitment" scheme, similar to the notion introduced by Nguyen and Vadhan [78].

Our protocol for $\mathcal{F}_{\text{COM}}$ is as follows, with security parameter κ. It uses ideal access to 3 independent instances of $\mathcal{F}_{\text{EXT-COM}}^{(\omega_L,\omega_R)}$, which for clarity we will name $\mathcal{F}_0, \mathcal{F}_1, \mathcal{F}_2$. Bob is identified as the sender in $\mathcal{F}_0$, and the receiver in $\mathcal{F}_1$ and $\mathcal{F}_2$.

1. (Commit phase, on Alice input (COMMIT, x)) Bob chooses a random string $r \leftarrow \{0,1\}^\kappa$ and sends (COMMIT, r) to $\mathcal{F}_0$. Alice receives output $(\text{COMMITTED}, \tau_0)$ and Bob receives output $(\text{TRANSCRIPT}, \tau_0, \gamma_0)$.
2. Alice sends (COMMIT, x) to $\mathcal{F}_1$. Alice receives output $(\text{TRANSCRIPT}, \tau_1, \gamma_1)$ and Bob receives output $(\text{COMMITTED}, \tau_1)$.
3. Alice sends $(\text{COMMIT}, 0^\kappa)$ to $\mathcal{F}_2$. Alice receives output $(\text{TRANSCRIPT}, \tau_2, \gamma_2)$ and Bob receives output $(\text{COMMITTED}, \tau_2)$.
4. Bob sends REVEAL to $\mathcal{F}_0$, and Alice receives output $(\text{REVEAL}, \gamma_0, r)$. Bob outputs COMMITTED.
5. (Reveal phase, on Alice input REVEAL) Alice sends x to Bob, then uses a WI proof to prove the following statement $\mathcal{S}(x, r, \tau_1, \tau_2)$:

   There exists γ such that either $\mathsf{opening}(\tau_1, \gamma) = x$ or $\mathsf{opening}(\tau_2, \gamma) = r$.

   Bob outputs (REVEAL, x) if the proof verifies.

It is straight-forward to see that the protocol is correct; i.e., simulation is trivial when both parties are honest. Simulation is trivial when both parties are corrupt, too. We consider the other two cases.

Simulation when Alice is corrupt: Since Bob has no private inputs to $\mathcal{F}_{\text{COM}}$, the simulator faithfully simulates the honest Bob protocol and honest functionalities $\mathcal{F}_0, \mathcal{F}_1, \mathcal{F}_2$. If the simulated Bob ever aborts, then the simulation also aborts. In step (2), the simulator obtains the value z_1 the value recorded by $\mathcal{F}_1$. When the commit phase finishes, the simulator sends (COMMIT, z_1) to $\mathcal{F}_{\text{COM}}$. Later, in the reveal phase, if the simulated ever Bob outputs (REVEAL, z_1), then the simulator sends REVEAL to $\mathcal{F}_{\text{COM}}$; if the simulated Bob outputs (REVEAL, x) for some $x \neq z_1$, then the simulation aborts. The simulation is clearly perfect except in the case where the simulated Bob outputs (REVEAL, x) for some $x \neq z_1$. We will show that this event happens with only negligible probability.

First, we argue that at the end of the commitment phase, the probability that there exists γ such that $\mathsf{opening}(\tau_2, \gamma) = r$ is negligible. By the security property of ω_L, the functionality $\mathcal{F}_2$ records a value z_2 such that (except with negligible probability) there does not exist γ such that $\mathsf{opening}(\tau_2, \gamma) \neq z_2$. Hence, it suffices to show that $\mathcal{F}_2$ records $z_2 = r$ with at most negligible probability. However, consider an adversary $\mathcal{A}$ attacking the standalone hiding property of ω_R. Adversary $\mathcal{A}$ internally simulates Alice and the functionality instances $\mathcal{F}_0$ and $\mathcal{F}_2$. It simulates Alice's interaction with $\mathcal{F}_1$ by interacting in a challenge commit phase of ω_R, to a random value r. After the commit phase, $\mathcal{A}$ outputs the value z_2 output by its internally simulated $\mathcal{F}_2$. By the standalone hiding property of ω_R, this output can equal r with only negligible probability.

Given this, with overwhelming probability, the second clause of the WI proof statement is false. By the security property of ω_L, the first clause is only true when Alice is attempting to reveal to $x = z_1$, except with negligible probability. Thus by the soundness of WI proof, if the simulated Bob outputs (REVEAL, x), then $x = z_1$ except for negligible probability, as desired.

Simulation when Bob is corrupt: Here, the simulator will simulate $\mathcal{F}_0$, $\mathcal{F}_1$ and $\mathcal{F}_2$ during the commitment phase, as follows:

1. First, it carries out an honest simulation of step (1), where it faithfully runs $\mathcal{F}_0$ and the receiver's protocol with $\mathcal{F}_0$. At the end of this it obtains a value z_0 as the value recorded by $\mathcal{F}_0$.
2. Then it simulates step (2) by internally simulating $\mathcal{F}_1$ and the honest sender, but with the sender's input as 0^κ (instead of Alice's input x, which it does not know yet).
3. It simulates step (3) similarly, but this time using z_0 as the sender's input (instead of 0^κ); note that this yields (τ_2, γ_2) such that $\mathsf{opening}(\tau_2, \gamma_2) = z_0$.
4. Then it simulates step (4), the reveal phase of $\mathcal{F}_2$. If the simulated $\mathcal{F}_2$ outputs (REVEAL, r) to the simulated sender, then the simulator ensures that $r = z_0$. If this is not the case, then the simulator fails.

The reveal phase is simulated as follows:

1. First the simulator obtains (REVEAL, x) from $\mathcal{F}_{\text{COM}}$. It simulates the protocol execution by sending x and then gives a WI proof for the statement $\mathcal{S}(x, r, \tau_1, \tau_2)$, by using the witness γ_2 and the fact that $\mathsf{opening}(\tau_2, \gamma) = r$.

First, we observe that the probability of the simulator failing (in step (4)) of commitment is negligible (by the security of ω_L). To show that the simulation is indistinguishable from the real protocol execution (conditioned on the simulator not failing), we shall rely on the hiding property of ω_R and the witness indistinguishability of the WI proof. In more detail, we employ the following hybrid simulators:

Hybrid 1: Same as the simulation, except that in step (2) the simulator uses Alice's true input x rather than 0^κ as the input to the (simulated) sender in its interaction with (simulated) $\mathcal{F}_1$. The entire interaction can be carried out by an adversary in the standalone hiding experiment for ω_R: the adversary receives either a commitment to x or to 0^κ, and it can simulate the rest of the interaction without receiving the opening of that commitment (the opening is not used as a witness to the WI proof later). Thus these two interactions are indistinguishable.

Hybrid 2: Same as above, except that in the reveal phase, the simulator uses the witness γ_1 in the WI proof, since $\mathsf{opening}(\tau_1, \gamma_1) = x$. This interaction is indistinguishable from the previous by a straightforward application of the witness-indistinguishability property in the WI proof.

Real world: Same as above, except that the simulator sends 0^κ to $\mathcal{F}_2$ instead of z_0, in step (3). This interaction is indistinguishable from the previous hybrid, by an identical argument as was used to show that Hybrid 1 and the simulation are indistinguishable. □

5.6 Obtaining Extractable Commitment from $\mathcal{F}_{\text{CC}}$

In this section, we show how $\mathcal{F}_{\text{EXT-COM}}$ can be securely realized using $\mathcal{F}_{\text{CC}}$. We first show a protocol π in the $\mathcal{F}_{\text{CC}}$-hybrid world, and then define appropriate (π_L, π_R) protocols such that π is a secure realization of $\mathcal{F}_{\text{EXT-COM}}^{(\pi_L,\pi_R)}$.

Parameters. Let Com be a statistically binding, standalone-secure commitment scheme with a non-interactive reveal phase (for instance, Naor's commitment scheme [75], which relies only on the existence of one-way functions). Let $\mathcal{C}_1, \ldots$ be a family of error-correcting codes, with the following properties:

- $\mathcal{C}_i$ is a linear (n_i, k_i) code over $GF(2)$, with generator matrix M_i.
- $k_i, n_i \in \Theta(i)$.
- It is possible to efficiently (polynomial time in i) correct $\Theta(n_i)$ errors in $\mathcal{C}_i$.

These parameters can be easily achieved, for instance, by a Justesen code [72].

The protocol. We define the following interactive protocol π in the $\mathcal{F}_{\text{CC}}$-hybrid model. The security parameter is κ.

1. (Commit phase.) On input (COMMIT, b) (for $b \in \{0,1\}$), Alice chooses random string $s \in \{0,1\}^{k_\kappa}$ and computes the associated codeword $t = (M_\kappa)s$. She commits to t using Com.
2. Bob chooses a string $y \in \{0,1\}^{n_\kappa}$ by setting each $y_i = 0$ with probability $k_\kappa/2n_\kappa = \Theta(1)$.
3. For $i \in \{1, \ldots, n_\kappa\}$, do:
   (a) Alice and Bob invoke a session of $\mathcal{F}_{\text{CC}}$ with Alice as sender. Alice sends t_i, the ith bit of t, as her input to $\mathcal{F}_{\text{CC}}$, and Bob sends input y_i.
   Recall that in $\mathcal{F}_{\text{CC}}$, Alice learns y_i; Bob learns t_i whenever $y_i = 0$.
   (b) If Alice sees that Bob has set $y_i = 0$ as many as k_κ times so far, then Alice aborts the protocol.
4. The bits of t that Bob has picked up are a linear function of s (a subset of the rows of M_κ), but are insufficient to completely determine s. Let g be a vector in $\{0,1\}^{k_\kappa}$ linearly independent of all the rows of M_κ for which Alice has revealed the corresponding bits of t. g can be computed by both parties in some canonical way. Then Alice sends $c = b \oplus \langle g, s \rangle$ to Bob.
5. Both parties locally output τ to consist of y, $y \wedge t$, g, c, and the transcript of the commitment to t in step (1).
6. Alice locally outputs γ to consist of s, t and the non-interactive opening to the commitment t.
7. (Reveal phase) Alice sends γ to Bob. We define $\text{opening}(\tau, \gamma) = \perp$ if it does not contain a valid opening of the commitment of t to a valid codeword $M_\kappa s$, or if the bits of t are not consistent with y and $y \wedge t$ computed in step (3). Otherwise, $\text{opening}(\tau, \gamma) = c \oplus \langle g, s \rangle$.

We define two protocols π_L and π_R (in the plain model, without access to $\mathcal{F}_{\text{CC}}$), as follows.

- π_L is identical to π, except that Bob honestly plays the role of $\mathcal{F}_{\text{CC}}$. Thus in step (3), Alice sends every bit t_i to Bob, and Bob responds by sending y_i to Alice.
  After the commit phase, Bob uses the error-decoding algorithm of $\mathcal{C}_\kappa$ to decode the sequence of bits $t = t_1 t_2 \cdots$ to its maximum likelihood dataword $\tilde{s}$, and locally outputs the extracted value $z = c \oplus \langle g, \tilde{s} \rangle$.
- π_R is identical to π, except that Alice honestly plays the role of $\mathcal{F}_{\text{CC}}$. Thus in step (3), Bob sends each y_i to Alice and Alice responds appropriately according to t_i.

Lemma 5.27.

1. *If* Com *is a statistically binding commitment scheme with non-interactive reveal, then* (π_L, π_R) *are a pair of complementary statistically binding commitment protocols.*
2. *The protocol* π *securely realizes* $\mathcal{F}_{\text{EXT-COM}}^{(\pi_L,\pi_R)}$ *in the* $\mathcal{F}_{\text{CC}}$*-hybrid model.*

Proof. Given that part (1) is true, part (2) is easily demonstrated via a trivial simulation, since (π_L, π_R) are simply π with $\mathcal{F}_{\text{CC}}$ honestly "collapsed" into the responsibilities of one party. The non-trivial step is to show that part (1) is true.

First, we claim that π_R is a statistically binding standalone commitment scheme. This is straight-forward by the security of the component Com commitment scheme. We remark that, by applying a standard Chernoff bound, we see that an honest Bob will request to see more than k_κ bits of t only with exponentially low probability κ.

Next, we must show that π_L is extractable. As in Definition 5.25, we consider an interaction between a corrupt Alice and honest Bob. Let $\tilde{t}$ be the sequence of inputs sent by Alice in step (3). After step (4), Bob decodes $\tilde{t}$ to obtain maximum likelihood dataword $\tilde{s}$, and locally outputs $z = c \oplus \langle g, \tilde{s} \rangle$.

We now argue that the extracted value z is correct. In step (1) of π_R, Alice gives a statistically binding commitment, so with overwhelming probability there is a well-defined unique value t^* such that the commitment can be successfully opened only to t^*. We condition on this overwhelming-probability event. If t^* is not a codeword of $\mathcal{C}_\kappa$, then Bob will never accept in the reveal phase of $\hat{\pi}$, and our extraction is trivially correct. Otherwise, assume t^* is a codeword, $t^* = (M_\kappa)s^*$. If s^* is equal to $\tilde{s}$ computed by Bob, then the extraction is also correct.

However, if $\tilde{s} \neq s^*$, then the Hamming distance between $\tilde{t}$ and codeword t^* is at least the minimum distance of $\mathcal{C}_\kappa$, which is $d = \Theta(n_\kappa)$. With overwhelming probability $1 - (k_\kappa/2n_\kappa)^d = 1 - O(1)^{\Theta(\kappa)}$, one of these d positions would have appeared in τ as a result of Bob choosing $y_i = 0$. When this happens, then Bob will never accept in the reveal phase and our extraction is correct. □

5.7 Obtaining Extractable Commitment from $\mathcal{F}_{\text{COIN}}$

In this section, we show how $\mathcal{F}_{\text{EXT-COM}}$ can be securely realized using $\mathcal{F}_{\text{COIN}}$. We first show a protocol ρ in the $\mathcal{F}_{\text{CC}}$-hybrid world, and then define appropriate (ρ_L, ρ_R) protocols such that ρ is a secure realization of $\mathcal{F}_{\text{EXT-COM}}^{(\rho_L, \rho_R)}$. Protocol ρ uses as a component the following protocol $\hat{\psi}$:

Compiled OT protocol $\hat{\psi}$. Suppose ψ_{psv} is a passive-secure OT protocol (as in Assumption 5.1) with security parameter κ, that uses $R(\kappa)$ bits of randomness. Let Com denote a statistically binding, standalone-secure commitment scheme with non-interactive reveal phase (for instance, Naor's commitment scheme [75], which relies only on the existence of one-way functions). Finally, let $G_n : \{0,1\}^n \to \{0,1\}^{2n}$ be a PRG family.

We define the following OT protocol $\hat{\psi}$ as follows, with security parameter κ. We assume Alice is the sender, with input bits x_0, x_1, and Bob the receiver with input bit b.

1. Alice and Bob use $\mathcal{F}_{\text{COIN}}$ to generate a random $\sigma \leftarrow \{0,1\}^{2\kappa}$.
2. Alice chooses a random string $r_A \leftarrow \{0,1\}^{R(\kappa)}$ and commits to x_0, x_1, and r_A using Com.
3. Alice uses a zero-knowledge proof of knowledge scheme to prove that she knows a valid opening for the commitment in the previous step.
4. Bob sends a random string $r'_A \leftarrow \{0,1\}^{R(\kappa)}$ to Alice.
5. Bob chooses a random string $r_B \leftarrow \{0,1\}^{R(\kappa)}$ and commits to (b, r_B) using Com.
6. Bob uses a zero-knowledge proof of knowledge scheme to prove that he knows a valid opening for the commitment in the previous step.
7. Alice sends a random string $r'_B \leftarrow \{0,1\}^{R(\kappa)}$ to Bob.
8. Both parties start running the ψ_{psv} protocol, with Alice using input (x_0, x_1) and random tape $r_A \oplus r'_A$, and Bob using input b and random tape $r_B \oplus r'_B$. At each step, the parties do the following: Suppose the transcript so far in the ψ_{psv} protocol is τ:
   (a) If it is Alice's turn in the ψ_{psv} protocol, then she sends m according to the ψ_{psv} protocol. Then she uses a zero-knowledge proof scheme to prove that there exists $(\tilde{x}_0, \tilde{x}_1, \tilde{r}_A)$ such that her commitments in step (2) can be successfully opened to $\tilde{x}_0$, $\tilde{x}_1$, and $\tilde{r}_A$ respectively, and the ψ_{psv} protocol instructs Alice to send m when her input is $(\tilde{x}_0, \tilde{x}_1)$, her random tape is $\tilde{r}_A \oplus r'_A$, and the transcript so far is τ.
   (b) If it is Bob's turn in the ψ_{psv} protocol, then he sends m according to the ψ_{psv} protocol. Then he uses a witness-indistinguishable proof scheme to prove that there exists $(\tilde{b}, \tilde{r}_B, s)$ such that *either* $G_\kappa(s) = \sigma$, or his commitment in step (5) can be successfully opened to $(\tilde{b}, \tilde{r}_B)$, and the ψ_{psv} protocol instructs Bob to send m when his input is $\tilde{b}$, his random tape is $\tilde{r}_B \oplus r'_B$, and the transcript so far is τ.

9. If at any point, a zero-knowledge or witness-indistinguishable proof by the other party fails to verify, the parties immediately abort. Otherwise, they both output the values prescribed by the ψ_{psv} protocol when it terminates.

Some explanation of $\hat{\psi}$ is in order. We have essentially applied the GMW compiler [41] to ψ_{psv}, with the following important differences:

- In steps (3) and (6), the parties prove knowledge of the commitments to their inputs and random-tape shares. This is important because we eventually reduce an adversary running $\hat{\psi}$ to a passive adversary running ψ_{psv}. For technical reasons, we need to extract inputs and the random tape share in this step, but rewinding extraction is sufficient.
- Bob does not prove the standard GMW-style statement. Instead, he gives a witness-indistinguishable proof of a statement containing a trapdoor clause related to the public randomness σ. This extra trapdoor allows a *straight-line* simulator for a corrupt Bob to prove false statements (by choosing σ from the range of G and using the trapdoor witness), which is crucial in our subsequent security reductions.
- Note that $\hat{\psi}$ involves Alice committing to her inputs x_0, x_1 separately, but never opening these commitments. We use this fact and eventually open these commitments in our overall protocol.

Interactive commitment protocol ρ. We now define a statistically binding standalone-secure commitment protocol $\hat{\rho}$. The protocol $\hat{\rho}$ has security parameter κ, and is as follows:

1. (Commit phase, on Alice input (COMMIT, b), with $b \in \{0,1\}$) For $i \in \{1, \ldots, \kappa\}$, do:
   (a) Alice chooses random bits $x_i^{(0)}$ and $x_i^{(1)}$, and Bob chooses a random bit a_i.
   (b) The parties run the $\hat{\psi}$ OT protocol, so that Bob obtains $x_i^{(a_i)}$. This interaction also results in commitments to $x_i^{(0)}$ and $x_i^{(1)}$ under the Com commitment scheme.
2. Alice and Bob obtain $r \in \{0,1\}^\kappa$ using $\mathcal{F}_{\text{COIN}}$.
3. Alice sends $c = b \oplus \left(\bigoplus_i x_i^{(r_i)} \right)$ to Bob.
4. Both parties output τ consisting of all of the Com-commitments to values $x_i^{(r_i)}$.
5. Alice locally outputs γ to consist of the non-interactive openings for the commitments to each $x_i^{(r_i)}$.
6. (Reveal phase, on Alice input REVEAL) Alice sends γ. We define $\mathsf{opening}(\tau, \gamma) = \perp$ if the decommitments to $x_i^{(r_i)}$ bits are not valid. Otherwise, $\mathsf{opening}(\tau, \gamma) = c \oplus \left(\bigoplus_i x_i^{(r_i)} \right)$.

We then define the two related protocols, in the plain model:

- ρ_L is identical to π, except that Bob honestly plays the role of $\mathcal{F}_{\text{COIN}}$. However, he chooses the coins r for step (2) at the beginning of the protocol, and in step (1c) he runs the $\hat{\psi}$ protocol obtaining $x_i^{(r_i)}$ instead of $x_i^{(a_i)}$. After the commit phase, Bob locally outputs the extracted value $z = c \oplus \left(\bigoplus_i x_i^{(r_i)} \right)$.
- ρ_R is identical to ρ, except that Alice honestly plays the role of $\mathcal{F}_{\text{COIN}}$.

Lemma 5.28.

1. *If ψ_{psv} is a passive-secure OT protocol, and* Com *is a statistically binding commitment scheme with non-interactive reveal, then (ρ_L, ρ_R) are a pair of complementary statistically binding commitment protocols.*
2. *The protocol ρ securely realizes $\mathcal{F}_{\text{EXT-COM}}^{(\rho_L, \rho_R)}$ in the $\mathcal{F}_{\text{COIN}}$-hybrid model.*

Proof. (1) We first show that ρ_R is a statistically binding standalone-secure commitment protocol. Statistical binding follows directly from the fact that the opening value of the commitment is a fixed function of the openings of n statistically binding Com-commitments.

To show that ρ_R is computationally hiding, we consider an interaction between an honest Alice and a malicious Bob, in the commit phase of ρ_R. To establish the computational hiding property, we construct a (rewinding) simulator that simulates the commit phase against Bob without knowledge of Alice's bit. We construct the simulator via the following sequence of hybrids:

Real world: The simulator simply runs the honest Alice protocol, on Alice's input. This is exactly what happens in the real interaction between Alice and Bob.

Hybrid 1: Same as above, but each time in the $\hat{\psi}$ subprotocol, the simulator extracts Bob's input (b, r_B), possibly by rewinding, from the proof of knowledge in step (6). The simulator flips fair coins R and sends $r'_B = R \oplus r_B$ in step (7) of $\hat{\psi}$. Intuitively, the simulator will force Bob to run ψ_{psv} on the coins R of its choice. This hybrid is distributed exactly as the previous.

Hybrid 2: Same as above, but each time in the $\hat{\psi}$ protocol, the simulator also computes the next-message function of ψ_{psv} on input b and random tape R. If the honest next message disagrees with the message Bob gives in step (8b), then the simulator aborts. By the soundness of the extraction of (b, r_B), the statistical binding of the commitment to (b, r_B), and the soundness of the WI proof that Bob gives in step (8b),[3] this hybrid is indistinguishable from the previous. However, we are guaranteed that in this hybrid (and subsequent hybrids), Bob is running the ψ_{psv} sub-protocol honestly on some input on a randomly chosen random tape (note that it is important that the simulator honestly sampled the random tape — it not enough for the random tape to be distributed uniformly, since Bob may be able to generate correctly distributed random tapes with some trapdoor).

Hybrid 3: Same as above, but each time in steps (3) and (8a) of the $\hat{\psi}$ protocol, the simulator gives simulated zero-knowledge proofs (perhaps by rewinding Bob). This difference is indistinguishable by the security of the zero-knowledge proof schemes.

Hybrid 4: Same as above, but each time in the $\hat{\psi}$ protocol, the simulator chooses two sets of independent random bits: x'_0, x'_1 for the commitment in step (2), and x_0, x_1 to use as input to the ψ_{psv} protocol in step (8a). This difference is indistinguishable by the computational hiding guarantee of the Com commitments, since the openings for these commitments are never needed (they are no longer used as witnesses in the ZK proof protocols).

Hybrid 5: Same as above, but the simulator chooses the string r from step (2) of ρ_R uniformly from the set $\{0,1\}^\kappa \setminus \{a_1 a_2 \cdots a_\kappa\}$, where a_i is Bob's input to the ith instance of the $\hat{\psi}$ subprotocol, extracted above (instead of randomly from $\{0,1\}^\kappa$). This difference is statistically indistinguishable from the previous hybrid.

Simulation: Same as above, but the simulator chooses the message c from step (3) of ρ_R randomly. This hybrid defines our final simulation, since it does not need to know Alice's bit, as desired. To see that these last two hybrids are indistinguishable, choose an i such that $a_i \neq r_i$ (such an i must exist by our assumption in hybrid 5). Then in hybrid 5, the value of $x_i^{(r_i)}$ influences Bob's view only in the ith instance of ψ_{psv}, and in the value c. However, since Bob is running ψ_{psv} honestly with input $a_i \neq r_i$, the transcript of this ψ_{psv} execution is computationally indistinguishable from a transcript that is independent of $x_i^{(r_i)}$. Thus the value c is pseudorandom, and our final simulation is computationally indistinguishable from the previous hybrid.

We next show that ρ_L has the desired extraction property. For this, we consider an interaction between a malicious Alice and honest Bob. We show that with overwhelming probability, Bob's outputs from step (1) of ρ_L are the *only* values to which Alice can successfully open the commitments of $x_i^{(r_i)}$ in the reveal phase. We omit the details, which follow very closely to the approach taken above. Briefly, we can argue that Alice runs the ψ_{psv} protocol honestly on some inputs, using an honestly sampled random tape (again by extracting from the proof of knowledge). As such, the correctness of the ψ_{psv} protocol implies that Bob will always learn the correct value of $x_i^{(r_i)}$ that underlies Alice's relevant Com-commitments. Thus, by the statistical binding property of these commitments, Bob has learned (with overwhelming probability) the only values to which Alice can successfully open.

(2) Finally, we show that ρ is a secure protocol for $\mathcal{F}_{\text{EXT-COM}}^{(\rho_L, \rho_R)}$ in the $\mathcal{F}_{\text{COIN}}$-hybrid model. The simulation is trivial, in that the simulator honestly engages in ρ_L against a corrupt Alice, and engages in ρ_R against a corrupt Bob. What remains is to argue that this simulation is indistinguishable from the real-world interaction. This is straight-forward for ρ_R, since there the honest party faithfully simulates the behavior of $\mathcal{F}_{\text{COIN}}$.

In the case of ρ_L, the simulation is sufficiently different from the real-world interaction, since in the simulation Bob runs $\hat{\psi}$ with inputs r_i instead of a_i. We carefully establish the soundness of the simulation through the following sequence of hybrids:

[3] Only with negligible probability is σ in the range of the PRG, so that clause of the WI proof is false.

Hybrid 1: Same as the real world, except that each time in step (1) of ψ, the simulator picks σ from the pseudorandom distribution G. This difference is indistinguishable by the pseudorandomness of the PRG G.

Hybrid 2: Same as before, except that each time in step (8b) of $\hat{\psi}$, the simulator uses the "trapdoor" witness corresponding to the preimage of σ. This difference is indistinguishable by the witness-indistinguishability of the WI proof scheme.

Hybrid 3: Same as before, except that each time in step (8) of $\hat{\psi}$, the simulator uses r_i as its inputs to the ψ_{psv} protocol, instead of a_i (recall that these have been chosen at the beginning of the simulation, and that the actual inputs to ψ_{psv} are no longer being used in the witness for the WI proof in this hybrid). The difference is indistinguishable by the passive security guarantee of the receiver in the ψ_{psv} protocol. To show this formally requires some care, but it follows the same approach as the previous reductions to passive security properties used earlier in this proof.

Hybrid 4: Same as before, except that each time in step (5) of $\hat{\psi}$, the simulator commits to r_i instead of a_i. Since the contents of these commitments are never used later (either by revealing them later, or as a witness in the WI proofs), this difference is indistinguishable by the hiding property of Com.

Hybrid 5: Same as before, except that each time in step (8b) of $\hat{\psi}$, the simulator uses the "legitimate" witness to the WI proof, corresponding to the opening of the commitment to r_i. This difference is indistinguishable by the witness-indistinguishability of the WI proof scheme.

Simulation: Same as before, except that each time in step (1) of ψ, the simulator faithfully chooses random coins σ instead of pseudorandom coins. Since the previous hybrid did not use the preimage to the pseudorandom coins σ, these two hybrids are indistinguishable by the pseudorandomness of G. We observe that this final hybrid represents exactly what our simulator does: runs ρ_L honestly. Thus the simulation is indistinguishable from the real-world interaction. □

5.8 Extensions & Open Problems

We now identify some possible extensions to our zero-one law, and some directions for future work suggested by our results.

5.8.1 Strengthening the Reduction

We have shown a zero-one law under the $\sqsubseteq^p_{nt}$ reduction. Since there is no zero-one law for the significantly stronger $\sqsubseteq^u_{nt}$ reduction (Chapter 4), a natural question is how much the $\sqsubseteq^p_{nt}$ reduction can be strengthened to still admit a zero-one law?

Fixed roles. Our definition $\mathcal{F} \sqsubseteq \mathcal{G}$ allows parties to access many instances of the functionality $\mathcal{G}$, and accessing $\mathcal{G}$ as either of the two parties that it expects to interact with. One may consider a stronger reduction called a *fixed-role reduction*, in which the protocol for $\mathcal{F}$ is required to interact with *all* instances of $\mathcal{G}$ in the same role. Some functionalities like $\mathcal{F}_{\text{COIN}}$ are symmetric with respect to the two parties, but others like $\mathcal{F}_{\text{CC}}$ are not.

Theorem 5.29. *There is no fixed-role reduction from $\mathcal{F}_{\text{COM}}$ to $\mathcal{F}_{\text{CC}}$.*

Proof. Suppose there is a protocol that securely realizes $\mathcal{F}_{\text{COM}}$ in the $\mathcal{F}_{\text{CC}}$-hybrid setting where Alice, the committer for $\mathcal{F}_{\text{COM}}$, is always the "sender" in $\mathcal{F}_{\text{CC}}$. Then Alice can equivocate in such a protocol, as follows: she internally runs the simulator for when Bob is corrupt, playing the role of corrupt Bob, and relaying the messages from the simulator to Bob. When the protocol requires Alice and Bob to access $\mathcal{F}_{\text{CC}}$, Alice obtains an input to be sent to $\mathcal{F}_{\text{CC}}$ from the simulator by telling it that Bob's input to $\mathcal{F}_{\text{CC}}$ is 1 (i.e., Bob chooses to see Alice's input); then Alice sends this input to $\mathcal{F}_{\text{CC}}$. If it turns out that Bob indeed chooses to see Alice's input, the simulation continues normally. However, if Bob chooses to not see Alice's input, Alice *rewinds* the simulator, tells it that Bob's input to $\mathcal{F}_{\text{CC}}$ is 0; she obtains an input to $\mathcal{F}_{\text{CC}}$ from the simulator, but does not forward it to $\mathcal{F}_{\text{CC}}$ (because she has already sent an input). Note that it does not matter if the input to $\mathcal{F}_{\text{CC}}$ by the simulator changes after rewinding, as this input is not revealed to Bob. Thus a corrupt Alice can faithfully run the simulator, and in particular open the commitment to any bit specified at the beginning of the opening phase, and the protocol is not secure.

On the other hand, suppose there is a protocol for $\mathcal{F}_{\text{COM}}$ in the $\mathcal{F}_{\text{CC}}$-hybrid model, in which Alice, the committer for $\mathcal{F}_{\text{COM}}$ is always the "receiver" in $\mathcal{F}_{\text{CC}}$. In this case, we show that Bob can learn Alice's input after the commit phase and before the reveal phase. For this Bob will internally run the simulator for when Alice is corrupt, relaying the messages from Alice in the actual protocol to this simulator. Now again, when Alice and Bob are required to access $\mathcal{F}_{\text{CC}}$, Bob (who is the sender in $\mathcal{F}_{\text{CC}}$) will generate an input for $\mathcal{F}_{\text{CC}}$ by telling the simulator that Alice's input to $\mathcal{F}_{\text{CC}}$ is 1. Subsequently, if her input turns out to be 0, Bob will rewind the simulator, give it 0 as Alice's input, as above. In this case, Bob obtains Alice's input as the bit extracted by the simulator at the end of the commitment phase. □

Since $\mathcal{F}_{\text{CC}}$ is unconditionally non-trivial, this theorem demonstrates that there is no zero-one law under this fixed-role reduction. This impossibility highlights the fact that $\mathcal{F}_{\text{CC}}$ is indeed a functionality of rather low complexity, and justifies our somewhat complicated protocol used to realize $\mathcal{F}_{\text{COM}}$ using $\mathcal{F}_{\text{CC}}$.

Weakening the communication channel. We formulated our results in the private channel model, where the two parties can communicate with each other privately via an ideal communication channel. (However, the adversary is allowed learn the number of bits communicated.) This is perhaps the natural model for capturing the cryptographic complexity of 2-party computation. Nevertheless, our main result readily extends to a model where the parties use a public channel completely controlled by the adversary. This follows from the fact that under Assumption 5.1, private channels can be securely realized using public channels [38]. (If the public channels are not authenticated channels, then digital signatures are used to achieve authentication, with identities of the parties being their signature verification keys. Note that digital signatures also follow from one-way functions [90], in turn implied by Assumption 5.1.)

Even if the reduction is strengthened so that no additional channel is given (i.e., the protocol is allowed to access only instances of $\mathcal{F}$), our zero-one results still hold, since $\mathcal{F}_{\text{PVT}} \sqsubseteq_{nt} \mathcal{F}$ for all non-trivial $\mathcal{F}$ (in the terminology of [70], non-trivial functionalities $\mathcal{F}$ *enable bit transmission*). Suppose $\mathcal{F}$ is some DFF, and suppose for all sequences of inputs $\vec{x}_0, \vec{x}_1, \vec{y}$, the output for Bob is the same when $\mathcal{F}$ is given input sequence $(\vec{x}_0, \vec{y})$ or $(\vec{x}_1, \vec{y})$, and vice-versa with the roles of Alice and Bob reversed. Then $\mathcal{F}$ is trivial, since either party's output does not depend on the other's inputs. So if $\mathcal{F}$ is non-trivial, then the above condition (or its symmetric variant) holds. Then $\mathcal{F}_{\text{PVT}} \sqsubseteq_{nt} \mathcal{F}$ as follows: To send a bit b to Bob, Alice sends input sequence $\vec{x}_b$ to $\mathcal{F}$, and to receive it Bob sends sequence $\vec{y}$. It is straight-forward to see that this protocol is secure.

Hardness assumptions. Our results rely on the existence of passive-secure protocols for $\mathcal{F}_{\text{OT}}$. From this assumption, we obtain one-way functions and other useful primitives. However, in our protocol constructions we use most of these primitives in a non-black-box way (in particular, we use witness-indistinguishable and zero-knowledge proofs of statements relating to these primitives). It is not known whether the zero-one law holds with respect to protocols that use Assumption 5.1 in a *black-box way*. In particular, it is not obvious how witness-indistinguishable/zero-knowledge proofs can be avoided in our constructions (or how our proofs can be made to avoid statements involving the evaluation of one-way functions or commitment scheme algorithms).

While we have shown that Assumption 5.1 is necessary for the zero-one law to hold, we do not know the state of affairs if only a weaker assumption (like the existence one-way functions) is true. Intuitively, the weaker the cryptographic assumptions considered, the closer the $\sqsubseteq^p_{nt}$ reduction acts to the $\sqsubseteq^u_{nt}$ reduction. In Lemma 5.3, we showed that if the $\sqsubseteq^p_{nt}$-completeness class expands (in comparison to the $\sqsubseteq^u_{nt}$-completeness class) to intersect the class of passively realizable functionalities, then Assumption 5.1 is true, and in fact the completeness class expands to contain all non-trivial functionalities. However, Lemma 5.3 does not rule out the possibility that the $\sqsubseteq^p_{nt}$-completeness class is larger than the $\sqsubseteq^u_{nt}$-completeness class, but is disjoint from the class of passively realizable functionalities. In this case, perhaps only a much weaker hardness assumption is needed.

As a concrete example, consider the SSFE functionality whose function table is $\begin{bmatrix} 0 & 0 & 1 \\ 3 & 4 & 1 \\ 3 & 2 & 2 \end{bmatrix}$. This functionality is neither passively realizable nor $\sqsubseteq^u_{nt}$-complete. Is it $\sqsubseteq^p_{nt}$-complete under a cryptographic assumption weaker than Assumption 5.1; say, the existence of one-way functions? Alternatively, is $\sqsubseteq^p_{nt}$-completeness equivalent to $\sqsubseteq^u_{nt}$-completeness, assuming only one-way functions?

Adaptive security. We consider protocols which are secure only against static adversaries, and we do not know whether our results extend to the case of adaptive corruption. It is likely that a completely different approach, and also a much stronger hardness assumption will be needed to achieve adaptive security. The current state of the art is the result of Canetti et al. [23], who showed that $\mathcal{F}_{\text{COIN}}$ is complete for adaptive security, under the assumption of dense cryptosystems. Thus it suffices to identify a hardness assumption under which $\mathcal{F}_{\text{CC}}$ is complete for adaptive

security. Our current approach of obtaining full-fledged commitment from the intermediate extractable commitment $\mathcal{F}_{\text{EXT-COM}}$ is not adaptively secure.

Of independent interest would be to identify the minimal hardness assumption necessary for a zero-one law for adaptive security. In particular, it is not clear how an adaptively secure protocol for, say, $\mathcal{F}_{\text{OT}}$ in the $\mathcal{F}_{\text{COIN}}$-hybrid setting would imply a dense cryptosystem. However, this must be the case if the currently known hardness assumption is optimal.

5.8.2 Larger Classes of Functionalities

Our zero-one result is for the class of deterministic, finite functionalities. Some of the restrictions of this class are necessary for our result, while we conjecture that the result extends if some of the restrictions are relaxed.

Unbounded memory. In this work we strictly considered functionalities with finite memory. This restriction was crucial for our techniques, and we show that some memory restriction is crucial for the zero-one law to hold:

Theorem 5.30. *Let $g : \{0,1\}^* \to \{0,1\}^*$ be a one-way function, and define*

$$f(x) = \begin{cases} g(x) & \text{if } |x| \text{ is of the form } 2^{2^n} \text{ for some } n \\ x & \text{otherwise.} \end{cases}$$

Let $\mathcal{F}$ be the functionality that takes input x from Alice and delivers $f(x)$ to Bob. Then $\mathcal{F}$ is neither complete nor trivial under the $\sqsubseteq^p_{nt}$ reduction.

Proof. First, $\mathcal{F}$ is not trivial. This can be seen by an appeal to the splittability characterization of Theorem 3.20. $\mathcal{F}$ is not completely invertible, in particular it is non-invertible on security parameters of the form 2^{2^n}.

On the other hand, $\mathcal{F}$ is not complete. Consider any purported protocol for $\mathcal{F}_{\text{OT}}$ in the $\mathcal{F}$-hybrid setting. For most values of the security parameter k, access to $\mathcal{F}$ is equivalent to $\mathcal{F}_{\text{PVT}}$, since messages of length 2^{2^n} are superpolynomially long in k (and thus can never be sent to $\mathcal{F}$), or are sub-logarithmically short in k (and thus f can be efficiently inverted to a canonical pre-image). As such, for these values of k, a real-world adversary has exactly as much power as a simulator. Thus, a corrupt receiver can run the simulator algorithm for a corrupt sender, to extract both of the honest sender's inputs. This violates the security of $\mathcal{F}_{\text{OT}}$ for these values of the security parameter, so the purported protocol is not secure. □

The functionality $\mathcal{F}$ is admittedly contrived. While this impossibility result still does indicate the necessity of making *some* restriction on the class of functionalities, we leave open the problem of identifying the largest "natural" class of unbounded-memory functionalities that does satisfy the zero-one law.

For instance, in this paper we model functionalities as finite state automata; a natural model which allows unbounded memory but avoids letting the functionality evaluate one-way functions is to consider automata equipped with a counter or stack. We leave this as an interesting open question.

Randomized functionalities. Our restriction to deterministic functionalities is crucial for several parts of our proof. We conjecture that the zero-one law holds also for randomized (but still finite-memory) functionalities. A particular challenge is to consider that the *transition function* of a finite-memory functionality might be randomized. However, even some simpler questions about randomized functionalities are unresolved. For instance, which randomized *SFE* functionalities admit unconditional protocols for $\mathcal{F}_{\text{OT}}$?

Non-well-formed functionalities. Two other restrictions we require are that functionalities take inputs from both parties and give outputs to each party in a series of rounds, and that functionalities interact with the adversary only to model an asynchronous adversarially controlled communication network. We leave open the problem of relaxing these requirements, both for realization results (i.e., considering a functionality complete if a larger class of functionalities can be reduced to it) and for extending the scope of the zero-one result (i.e., developing techniques to analyze a larger class of functionalities).

Indeed, it is often very convenient to define intermediate functionalities which crucially use more sophisticated communication with the adversary, and which do not have a round-based input/output structure. Our splittability characterization (Chapter 3) does indeed completely characterize triviality for arbitrary functionalities like these, so it may be possible to develop tools for classifying them in the context of completeness.

Of secondary interest would be to have a convincing, tighter definition of when a functionality "abuses" its knowledge of which parties are corrupt. The current definition of well-formedness is motivated by ruling out

pathological cases, like a functionality which announces the list of corrupt parties. Such a functionality cannot be securely realized by any protocol in any realistic setting. However, since many realizable functionalities do indeed require knowledge of the corrupt parties, the current definition of well-formedness (which rules out *all* such dependence on the identities of corrupt parties) seems overkill. To extend the zero-one law to functionalities whose behavior depends on the identities of the corrupt parties, it will likely be necessary to develop a more refined notion of well-formedness.

CHAPTER 6

Reconciling Non-Malleability & Homomorphic Encryption

6.1 Overview

A recurring theme in cryptography is the tension between achieving powerful functionality and making strong security guarantees. In the case of encryption, a recent trend is towards achieving various kinds of computations on encrypted data, e.g., rerandomizability [43, 45], proxy re-encryption [11, 20], searchability [94, 26], predicate encryption [59], and various homomorphism properties [36, 79]. Encryption schemes with computational features are interesting and useful in their own right, but also are convenient in designing simple, intuitive protocols for many tasks that combine data privacy and computation (like mix-nets and voting schemes).

On the other hand, non-malleability is usually required for an encryption scheme to be directly useful in providing guarantees against malicious adversaries, in a composable security setting. In a very general sense, non-malleability means that an adversary cannot manipulate ciphertexts in any "unexpected" way. Unfortunately, existing non-malleability security definitions are incompatible with encryption schemes with computational features. CCA security (security against chosen ciphertext attack)[1] rules out the possibility of manipulating encrypted data in any way, CPA security (security against chosen plaintext attack) permits every possible kind of manipulation, and the very few security notions between these two extremes are not expressive enough to cover the many desired uses of encryption with computational features.

An ideal security definition for encryption with computational features would be one that realizes a sharp tradeoff; for example, "this encryption scheme allow operations x, y, and z, but *no other* operations are possible."

In this work we address this problem in the context of *homomorphic* public-key encryption schemes — those which allow (as a feature) anyone to obtain an encryption of $T(m_1, \ldots, m_k)$ given only the encryptions of unknown messages $m_1, \ldots, m_k$, for some allowed set of functions T. Homomorphic encryption schemes hide not only the underlying plaintext, but also the "history" of a ciphertext — i.e., whether it was derived by encrypting a known plaintext, or by applying a homomorphic operation applied to some other ciphertext(s). Such schemes have been extensively studied for a long time and have a wide variety of applications (cf. [10, 91, 29, 92, 52, 57, 33, 43, 54, 31]).

Challenges and Related Work. The first challenge is formally defining (in a convincing way) the intuitive requirement that a scheme "allow particular features but forbid all others." Security notions for regular encryption developed and matured over many years [42, 77, 89, 8, 35, 16], while arguably security definitions for homomorphic encryptions have lagged behind — to date, homomorphic encryptions are almost exclusively held to the weak standard of CPA security. In some applications (e.g., [31]) CPA security is indeed sufficient, but for others (e.g. [34]) it is not. In practical terms, this means that protocols that use homomorphic encryption usually have to employ the heavy machinery of zero-knowledge proofs or verifiable secret sharing to extend to security against malicious adversaries.

Very little work has addressed the possibility of homomorphic encryption schemes having malleability beyond the desired operations; one exception is Wikström [95], who addresses this question in a simpler setting for El Gamal encryption. Benignly-malleable (also called gCCA) security [93, 1] was proposed as a relaxation of CCA security, and was further relaxed in the definition of Replayable-CCA (RCCA) security [21]. RCCA security allows a scheme to be malleable, but only in ways which are guaranteed to preserve the underlying plaintext. However, relaxing CCA security in the same way does not yield an acceptable level of security when applied to more expressive homomorphic operations (see Section 6.3.1); a new approach to defining security is needed.

The second challenge is achieving the desired security with a construction based on standard assumptions — i.e., an encryption scheme that has a particular set of expressive homomorphic operations, but is non-malleable with respect to all other operations. Note that even if the set of allowed operations is very simple, supporting it can be very involved. Indeed, the problem of *rerandomizable RCCA encryption* considered in a recent series of works [21, 45, 83] corresponds to the simplest special case of our definitions.

[1] We will use the term CCA to refer to *adaptive* chosen-ciphertext security, also known as CCA2. When referring to CCA1 (also known as static CCA or "lunchtime attack") security, we will be explicit.

6.1.1 Our Results

We give several new security definitions to precisely capture the desired requirements in the case of *unary* homomorphic operations (those which transform a single encryption of m to an encryption of $T(m)$, for a particular set of functions T). We provide two new indistinguishability-based security definitions. The first definition, called *Homomorphic-CCA (HCCA) security*, formalizes the intuition of "non-malleability except for certain prescribed operations," and the second definition, called *unlinkability*, formalizes the intuition that ciphertexts not leak their "history." To justify our non-malleability definition, we show that it subsumes the standard CCA, gCCA, and RCCA security definitions (Theorem 6.5). We further show that our two new security requirements imply a natural definition of security in the Universal Composition framework (Theorem 6.9). Using the UC framework to define security of encryption schemes was already considered in [16, 21, 81, 19].

Our main result is to describe a parameterized encryption scheme which achieves our definitions for a wide range of allowed (unary) homomorphism operations. The construction is secure under the standard DDH assumption, and supports homomorphic operations related to a cyclic group operation as its homomorphic feature. We also describe a very efficient construction that supports homomorphic operations in *arbitrary* groups, but which only achieves a weak level of unlinkability.

To demonstrate the practical utility of our definitions, we show a simple, intuitive protocol for an anonymous opinion polling functionality, which uses HCCA-secure encryption as a key component. Even though the component encryption scheme supports only unary operations, we are able to perform non-trivial computations on a set of independently encrypted inputs, crucially using the scheme's homomorphic features. Furthermore, because of the strong non-malleability guarantee, this simple protocol achieves UC security against malicious adversaries without resorting to the overhead of zero-knowledge proofs or general-purpose multi-party computation. Our protocol can be instantiated with either of our two new constructions, resulting in a very efficient protocol.

Finally, we consider extending our definitions to the case of *binary* homomorphic operations (those which combine pairs of ciphertexts). We show that the natural generalization of our UC security definition to this scenario is unachievable for a large class of useful homomorphic operations (Theorem 6.16). However, we also give a positive result when one of our requirements is slightly relaxed. In particular, if we allow a ciphertext to leak only the *number* of homomorphic operations that were applied to obtain the ciphertext, then it is possible to construct a homomorphic scheme that supports the binary group operation (that is, it is possible to obtain $\mathsf{Enc}(\alpha * \beta)$ from $\mathsf{Enc}(\alpha)$ and $\mathsf{Enc}(\beta)$, but no other operations are possible).

6.2 Preliminaries

6.2.1 Homomorphic Encryption Syntax

Let $\mathcal{M}$ be a space of plaintext messages, let $\perp$ be a special error indicator symbol not in $\mathcal{M}$, and let $\mathcal{T}$ be a "transformation space" — i.e., a set of polynomial-time computable functions from $(\mathcal{M} \cup \{\perp\})^k$ to $\mathcal{M} \cup \{\perp\}$. We call the elements of $\mathcal{T}$ the *allowable transformations*.

An encryption scheme consists of three probabilistic polynomial-time (polynomial in the implicit security parameter) algorithms: KeyGen, Enc and Dec.

A *$\mathcal{T}$-homomorphic encryption scheme* comes with an additional probabilistic polynomial-time algorithm CTrans, the homomorphic operation feature, which takes k ciphertexts and (the description of) a *transformation* from $\mathcal{T}$, and outputs another ciphertext.[2] For much of this work, we focus on the case where $k = 1$; i.e., when the homomorphic operation is *unary*.

Syntax of unary homomorphic encryption. Below we give the correctness properties for unary homomorphic encryption. These requirements can be slightly relaxed (e.g., to hold only with overwhelming probability over the randomness of they various procedures), but for simplicity we present the stronger formulations.

For all key pairs (PK, SK) in the support of KeyGen, we require the following:

1. For every plaintext $\mathsf{msg} \in \mathcal{M}$, we require $\mathsf{Dec}_{SK}(\mathsf{Enc}_{PK}(\mathsf{msg})) = \mathsf{msg}$, with probability 1 over the randomness of Enc.

2. For every purported ciphertext ζ and every $T \in \mathcal{T}$, we require $\mathsf{Dec}_{SK}(\mathsf{CTrans}(\zeta, T)) = T(\mathsf{Dec}_{SK}(\zeta))$, with probability 1 over the randomness of CTrans.

[2] Allowing CTrans to take the public key as additional input would also be a meaningful relaxation, but may not be suitable in some applications. See Section 6.8.

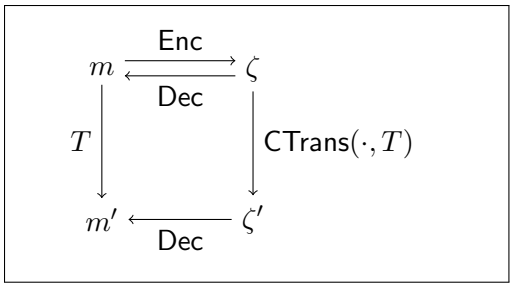

Figure 6.1: Illustration of the syntax and correctness of a homomorphic encryption scheme.

6.2.2 Decisional Diffie-Hellman (DDH) Assumption

Let $\mathbb{G}$ be a (multiplicative) cyclic group of prime order p. The *Decisional Diffie-Hellman (DDH) assumption in* $\mathbb{G}$ is that the following two distributions are computationally indistinguishable:

$$\{(g, g^a, g^b, g^{ab})\}_{g \leftarrow \mathbb{G}; a,b \leftarrow \mathbb{Z}_p} \quad \text{and} \quad \{(g, g^a, g^b, g^c)\}_{g \leftarrow \mathbb{G}; a,b,c \leftarrow \mathbb{Z}_p}.$$

Here, $x \leftarrow X$ denotes that x is drawn uniformly at random from a set X.

Cunningham chains. Our construction requires two (multiplicative) cyclic groups with a specific relationship: $\mathbb{G}$ of prime order p, and $\widehat{\mathbb{G}}$ of prime order q, where $\widehat{\mathbb{G}}$ is a subgroup of $\mathbb{Z}_p^*$. We require the DDH assumption to hold in both groups (with respect to the same security parameter).

As a concrete example, the DDH assumption is believed to hold in $\mathbb{QR}_p^*$, the group of quadratic residues modulo p, where p and $\frac{p-1}{2}$ are prime (i.e, p is a *safe prime*). Given a sequence of primes $(q, 2q+1, 4q+3)$, the two groups $\widehat{\mathbb{G}} = \mathbb{QR}_{2q+1}^*$ and $\mathbb{G} = \mathbb{QR}_{4q+3}^*$ satisfy the needs of our construction. A sequence of primes of this form is called a *Cunningham chain* (of the first kind) of length 3 (see [2]). Such Cunningham chains are known to exist with q as large as $2^{20,000}$. It is conjectured that infinitely many such chains exist, and with sufficient density.

6.2.3 Existing Security Definitions for Encryption

Several existing security definitions for encryption — CCA security, benignly-malleable (gCCA) security [93, 1], and Replayable-CCA (RCCA) security [21] — follow a similar paradigm: The adversary has access to a decryption oracle, and receives an encryption of one of two messages of his choice. His task is to determine which of the two messages has been encrypted, and we say that security holds if no adversary can succeed with probability significantly better than chance.

Since the adversary can simply submit the challenge ciphertext to the decryption oracle, it is necessary to restrict the oracle in some way. The difference among these three security definitions is in how the decryption oracle is restricted. This restriction corresponds intuitively to identifying when a ciphertext has been (potentially) derived from the challenge ciphertext. In Figure 6.2, we give a unified overview of these three security notions.

6.3 New Security Definitions for Homomorphic Encryption

In this section we present our formal security definitions. The first two are traditional definitions based on indistinguishability security games, while the third is a definition in the Universal Composition framework.

6.3.1 Homomorphic-CCA (HCCA) Security

Our first indistinguishability-based security definition formalizes the intuitive notions of message privacy and "non-malleability other than certain operations."

A natural idea for formalizing our desired notion of non-malleability is to start with the standard CCA security experiment and sufficiently relax it. Indeed, this is the approach taken in the definitions of benignly-malleable (gCCA) security [93, 1] and Replayable CCA (RCCA) security [21], which allow for a scheme to be only "mostly" non-malleable. In these security experiments (see Figure 6.2), the decryption oracle is guarded so as not to decrypt ciphertexts that may be legitimate "derivatives" of the challenge ciphertext. In CCA security, the only derivative is the

Setup: Pick $(PK, SK) \leftarrow \mathsf{KeyGen}$ and give PK to $\mathcal{A}$.

Phase I: $\mathcal{A}$ is given access to the decryption oracle $\mathsf{Dec}_{SK}(\cdot)$.

Challenge: Flip a coin $b \leftarrow \{0,1\}$. Receive from $\mathcal{A}$ two plaintexts $\mathsf{msg}_0, \mathsf{msg}_1$. Compute $\zeta^* \leftarrow \mathsf{Enc}_{PK}(\mathsf{msg}_b)$, and give ζ^* to $\mathcal{A}$.

Phase II: $\mathcal{A}$ is given access to the following "guarded" decryption oracle:

$$\mathsf{GDec}_{SK}(\zeta) = \begin{cases} \text{“GUARDED”} & \text{if } \mathcal{G}(SK, \zeta, \zeta^*, \mathsf{msg}_0, \mathsf{msg}_1) = 1 \\ \mathsf{Dec}_{SK}(\zeta) & \text{otherwise} \end{cases}.$$

Output: $\mathcal{A}$ outputs a bit b'. The *advantage* of $\mathcal{A}$ in this experiment is $\Pr[b' = b] - \frac{1}{2}$.

By using different predicates for $\mathcal{G}$, different levels of security are obtained:

Security:	$\mathcal{G}(SK, \zeta, \zeta^*, \mathsf{msg}_0, \mathsf{msg}_1)$:
CCA	$\zeta \stackrel{?}{=} \zeta^*$
gCCA	$R(\zeta, \zeta^*) \stackrel{?}{=} 1$, for arbitrary predicate R, provided that R satisfies $R(\zeta, \zeta^*) = 1 \Rightarrow \mathsf{Dec}_{SK}(\zeta) = \mathsf{Dec}_{SK}(\zeta^*)$
RCCA	$\mathsf{Dec}_{SK}(\zeta) \stackrel{?}{\in} \{\mathsf{msg}_0, \mathsf{msg}_1\}$

Figure 6.2: Security experiment for CCA security and its relaxations gCCA and RCCA

challenge ciphertext itself; in gCCA, derivatives are those which satisfy a particular binary relation with the challenge ciphertext; in RCCA, derivatives are those which decrypt to either of the two adversarially-chosen plaintexts.

However, the same approach of guarding the decryption oracle fails in the case of more general homomorphic encryption. For some sets of allowed transformations, it may be legal (i.e., possible via a feature of the scheme) to change the underlying plaintext of a ciphertext to any other possible plaintext. Indeed, in some instantiations of our construction, *every ciphertext* in the support of the Enc operation is a possible derivative of every other such ciphertext. Letting the decryption oracle refuse to decrypt possible derivatives in such a scenario would essentially weaken the security requirement to IND-CCA1 (i.e., "lunchtime attack") security, which is unsatisfactory.

Our approach to identifying "derivative" ciphertexts is completely different, and as a result our definition initially appears incomparable to these other standard definitions. However, Theorem 6.5 demonstrates that our new definition gives a generic notion of non-malleability which subsumes these existing definitions.

Overview and intuition. The formal definition, which we call *Homomorphic-CCA (HCCA) security*, appears below. Informally, in the security experiment we identify derivative ciphertexts not for normal encryptions, but for special "rigged" ciphertexts.

When $b = 0$ in the experiment, the adversary simply receives an encryption of his chosen plaintext msg^*, and gets access to an unrestricted decryption oracle. However, when $b = 1$ in the experiment, instead of an encryption of msg^*, the adversary receives a "rigged" ciphertext, generated by RigEnc without knowledge of msg^*. Such a rigged ciphertext need not encode any actual message, so if the adversary asks for it (or any of its derivatives) to be decrypted, we must compensate for the decryption oracle's response in some way, or else it would be easy to distinguish the $b = 0$ and $b = 1$ cases. For this purpose, the RigEnc procedure also produces some (secret) extra state information, which makes it possible to identify (via the RigExtract procedure) all ciphertexts derived from that particular rigged ciphertext, as well as *how* they were derived. So in the $b = 1$ scenario, the decryption oracle first uses RigExtract to check whether the given ciphertext was derived via a homomorphic operation of the scheme, and if so, compensates in its response. For example, if the query ciphertext was derived by applying the T transformation, then the decryption oracle should respond with $T(\mathsf{msg}^*)$, to mimic the $b = 0$ case.

It is easily seen that if it is feasible for an adversary to modify an encryption of $\mathsf{Enc}(\mathsf{msg})$ into a related encryption $\mathsf{Enc}(T(\mathsf{msg}))$, but RigExtract never outputs T, then there is a way for an adversary to distinguish between $b = 0$ and $b = 1$ in the experiment. Conversely, if RigExtract never outputs T, and yet no adversary has nonnegligible advantage in the IND-HCCA experiment, then the scheme is intuitively non-malleable with respect to T. Thus by restricting the range of the RigExtract procedure in the security definition, we enforce a limit on the malleability of the scheme.

Finally, because RigExtract uses the private key, as well as secret auxiliary information from RigEnc, we provide an oracle for these procedures. We do so in a "guarded" way that keeps the auxiliary shared information hidden from

the adversary in the experiment. These oracles are useful in security reductions where we replace several encryptions with rigged encryptions.

Definition 6.1. *A unary homomorphic encryption scheme is* Homomorphic-CCA (HCCA) secure *with respect to $\mathcal{T}$ if there are PPT algorithms* RigEnc *and* RigExtract*, where the range of* RigExtract *is $\mathcal{T} \cup \{\perp\}$, and such that for all PPT adversaries $\mathcal{A}$, the advantage of $\mathcal{A}$ in the IND-HCCA experiment (Figure 6.3) is negligible.*

Setup: Pick $(PK, SK) \leftarrow \mathsf{KeyGen}$ and give PK to $\mathcal{A}$.

Phase I: $\mathcal{A}$ is given access to the $\mathsf{Dec}_{SK}(\cdot)$ oracle and the following two "guarded" RigEnc and RigExtract oracles, which keep shared state to keep RigEnc's auxiliary information hidden:

$\mathsf{GRigEnc}_{PK}$**:** Takes no input. Run $(\zeta, S) \leftarrow \mathsf{RigEnc}_{PK}$, store (ζ, S), and return ζ_S.

$\mathsf{GRigExtract}_{SK}$**:** On input (ζ, ζ'), if (ζ', S) is stored for some S, then return $\mathsf{RigExtract}_{SK}(\zeta, S)$.

Challenge: Flip a coin $b \leftarrow \{0,1\}$. Receive from $\mathcal{A}$ a plaintext msg^*. If $b = 0$, compute $\zeta^* \leftarrow \mathsf{Enc}_{PK}(\mathsf{msg}^*)$; if $b = 1$, compute $(\zeta^*, S^*) \leftarrow \mathsf{RigEnc}_{PK}$. Give ζ^* to $\mathcal{A}$.

Phase II: $\mathcal{A}$ is given access to the same GRigEnc and GRigExtract oracles as in Phase I, as well as a "rigged" version of the decryption oracle $\mathsf{RigDec}_{SK}(\cdot)$. When $b = 0$, $\mathsf{RigDec}_{SK}(\cdot)$ is simply implemented as the normal decryption oracle $\mathsf{Dec}_{SK}(\cdot)$. When $b = 1$, $\mathsf{RigDec}_{SK}(\cdot)$ is implemented as follows:

$$\mathsf{RigDec}_{SK}(\zeta) = \begin{cases} T(\mathsf{msg}^*) & \text{if } \perp \neq T \leftarrow \mathsf{RigExtract}_{SK}(\zeta, S^*) \\ \mathsf{Dec}_{SK}(\zeta) & \text{otherwise} \end{cases}.$$

Output: $\mathcal{A}$ outputs a bit b'. The *advantage* of $\mathcal{A}$ in this experiment is $\Pr[b' = b] - \frac{1}{2}$.

Figure 6.3: IND-HCCA experiment, parameterized by set of allowed transformations $\mathcal{T}$

Simpler formulation. Some of constructions do not need the full power of the IND-HCCA experiment. For instance, consider the case where RigExtract may evaluate Dec_{SK} as a black-box, but otherwise does not use SK. Then the guarded oracles GRigEnc and GRigExtract are superfluous in the IND-HCCA experiment. An adversary can simulate the behavior and shared state of these two oracles using the public key PK and access to its own decryption oracle Dec/GDec.

Without these extra oracles, the IND-HCCA experiment collapses to a much simpler experiment, which we call IND-HCCA-simp (Figure 6.4). Several of our results involve RigExtract procedures of this required special form, and our corresponding proofs are made simpler by considering the IND-HCCA-simp experiment. However, our main construction (Section 6.5.2) does require the full power of the original experiment.

Setup: Pick $(PK, SK) \leftarrow \mathsf{KeyGen}$ and give PK to $\mathcal{A}$.

Phase I: $\mathcal{A}$ is given access to the $\mathsf{Dec}_{SK}(\cdot)$ oracle.

Challenge: Flip a coin $b \leftarrow \{0,1\}$. Receive from $\mathcal{A}$ a plaintext msg^*. If $b = 0$, compute $\zeta^* \leftarrow \mathsf{Enc}_{PK}(\mathsf{msg}^*)$; if $b = 1$, compute $(\zeta^*, S^*) \leftarrow \mathsf{RigEnc}_{PK}$. Give ζ^* to $\mathcal{A}$.

Phase II: $\mathcal{A}$ is given access to a "rigged" version of the decryption oracle $\mathsf{RigDec}_{SK}(\cdot)$, which is implemented as follows. When $b = 0$, $\mathsf{RigDec}_{SK}(\cdot)$ is simply implemented as the normal decryption oracle $\mathsf{Dec}_{SK}(\cdot)$. When $b = 1$, $\mathsf{RigDec}_{SK}(\cdot)$ is implemented as follows:

$$\mathsf{RigDec}_{SK}(\zeta) = \begin{cases} T(\mathsf{msg}^*) & \text{if } \perp \neq T \leftarrow \mathsf{RigExtract}_{SK}(\zeta, S^*) \\ \mathsf{Dec}_{SK}(\zeta) & \text{otherwise} \end{cases}.$$

Output: $\mathcal{A}$ outputs a bit b'. The *advantage* of $\mathcal{A}$ in this experiment is $\Pr[b' = b] - \frac{1}{2}$.

Figure 6.4: IND-HCCA-simp experiment, parameterized by set of allowed transformations $\mathcal{T}$

6.3.2 Unlinkability

There indeed is some tension between the HCCA definition given above and the intuitive notion of unlinkability that we desire. HCCA security implies that it is possible to track transformations applied to rigged ciphertexts, while unlinkability demands that ciphertexts not leak whether they were generated via a transformation. To reconcile this, we require unlinkability to apply only to *ciphertexts that successfully decrypt under a private key chosen by the challenger*. This excludes linkability via the RigEnc and RigExtract procedures, since tracking ciphertexts using RigExtract in general requires the tracking party to know the private key.

Our formal definition of unlinkability is given below. We note that the definition is more than just a correctness property, as it involves the behavior of the scheme's algorithms on maliciously-crafted ciphertexts. The security experiment also includes a decryption oracle, making it applicable even to adversaries with chosen-ciphertext attack capabilities.

Definition 6.2. *A unary homomorphic encryption scheme is* unlinkably homomorphic with respect to $\mathcal{T}$ ($\mathcal{T}$-unlinkable) *if for all PPT adversaries $\mathcal{A}$, the advantage of $\mathcal{A}$ in the IND-UH experiment (Figure 6.5) is negligible.*

Setup: Pick $(PK, SK) \leftarrow \mathsf{KeyGen}$ and give PK to $\mathcal{A}$.

Phase I: $\mathcal{A}$ is given access to the decryption oracle $\mathsf{Dec}_{SK}(\cdot)$.

Challenge: Flip a coin $b \leftarrow \{0,1\}$. Receive from $\mathcal{A}$ a ciphertext ζ and a transformation $T \in \mathcal{T}$. If $\mathsf{Dec}_{SK}(\zeta) = \perp$, do nothing; otherwise:

- If $b = 0$, give the adversary $\zeta^* \leftarrow \mathsf{Enc}_{PK}(T(\mathsf{Dec}_{SK}(\zeta))$.
- If $b = 1$, give the adversary $\zeta^* \leftarrow \mathsf{CTrans}(\zeta, T)$.

Phase II: Same as Phase I.

Output: $\mathcal{A}$ outputs a bit b'. The *advantage* of $\mathcal{A}$ in this experiment is $\Pr[b' = b] - \frac{1}{2}$.

Figure 6.5: IND-UH experiment, parameterized by set of allowed transformations $\mathcal{T}$

Relaxations. Unlinkability is a strong security guarantee that considers even maliciously crafted ciphertexts. We also consider relaxations of unlinkability which are implied by simple indistinguishability properties of the scheme's CTrans procedure, but are nonetheless useful.

Definition 6.3. *A unary homomorphic encryption scheme is* weakly unlinkably homomorphic with respect to $\mathcal{T}$ ($\mathcal{T}$-weakly-unlinkable) *if for all PPT adversaries $\mathcal{A}$, the advantage of $\mathcal{A}$ in the IND-WUH experiment (Figure 6.6) is negligible.*

Setup: Pick $(PK, SK) \leftarrow \mathsf{KeyGen}$ and give PK to $\mathcal{A}$.

Phase I: $\mathcal{A}$ is given access to the decryption oracle $\mathsf{Dec}_{SK}(\cdot)$.

Challenge: Flip a coin $b \leftarrow \{0,1\}$. Receive from $\mathcal{A}$ a plaintext msg and transformation $T \in \mathcal{T}$.

- If $b = 0$, give the adversary $\zeta_0^* \leftarrow \mathsf{Enc}_{PK}(\mathsf{msg})$ and $\zeta_1^* \leftarrow \mathsf{Enc}_{PK}(T(\mathsf{msg}))$.
- If $b = 1$, give the adversary $\zeta_0^* \leftarrow \mathsf{Enc}_{PK}(\mathsf{msg})$ and $\zeta_1^* \leftarrow \mathsf{CTrans}(\zeta_0^*, T)$.

Phase II: Same as Phase I.

Output: $\mathcal{A}$ outputs a bit b'. The *advantage* of $\mathcal{A}$ in this experiment is $\Pr[b' = b] - \frac{1}{2}$.

Figure 6.6: IND-WUH experiment, parameterized by set of allowed transformations $\mathcal{T}$

Unlike the stronger variant, weak unlinkability is implied by a simple indistinguishability condition: namely, that for all transformations $T \in \mathcal{T}$, plaintexts msg, and ciphertexts $\zeta \leftarrow \mathsf{Enc}_{PK}(\mathsf{msg})$, the distributions of $\mathsf{Enc}_{PK}(T(\mathsf{msg}))$ and $\mathsf{CTrans}(\zeta, T)$ are computationally indistinguishable given SK.

Definition 6.4. *A unary homomorphic encryption scheme is* transformation hiding with respect to $\mathcal{T}$ ($\mathcal{T}$-transformation-hiding) *if for all PPT adversaries $\mathcal{A}$, the advantage of $\mathcal{A}$ in the IND-TH experiment (Figure 6.7) is negligible.*

Setup: Pick $(PK, SK) \leftarrow \mathsf{KeyGen}$ and give PK to $\mathcal{A}$.

Phase I: $\mathcal{A}$ is given access to the decryption oracle $\mathsf{Dec}_{SK}(\cdot)$.

Challenge: Flip a coin $b \leftarrow \{0,1\}$. Receive from $\mathcal{A}$ a plaintext msg and transformation $T \in \mathcal{T}$.

- If $b = 0$, give the adversary $\zeta^* \leftarrow \mathsf{Enc}_{PK}(T(\mathsf{msg}))$.
- If $b = 1$, generate $\zeta \leftarrow \mathsf{Enc}_{PK}(\mathsf{msg})$ and give the adversary $\zeta^* \leftarrow \mathsf{CTrans}(\zeta, T)$.

Phase II: Same as Phase I.

Output: $\mathcal{A}$ outputs a bit b'. The *advantage* of $\mathcal{A}$ in this experiment is $\Pr[b' = b] - \frac{1}{2}$.

Figure 6.7: IND-TH experiment, parameterized by set of allowed transformations $\mathcal{T}$

Note that the IND-TH experiment is identical to the IND-WUH experiment, except that the adversary does not receive ζ_0^* in the IND-TH experiment. Thus transformation hiding is a strictly weaker requirement.

Transformation hiding is also implied by a simple indistinguishability criterion: namely, that for all transformations T and plaintexts msg, the distributions $\mathsf{Enc}_{PK}(T(\mathsf{msg}))$ and $\mathsf{CTrans}(\mathsf{Enc}_{PK}(\mathsf{msg}), T)$ are computationally indistinguishable given SK, over the randomness of both Enc and CTrans in the second expression. In particular, CTrans need not be randomized for a scheme to be transformation hiding.

6.3.3 Defining Security Using an Ideal Functionality

We also define the "Homomorphic Message Posting" functionality $\mathcal{F}_{\text{HMP}}^{\mathcal{T}}$ in the framework of Universally Composable security [16] as a natural security definition encompassing both unlinkability and our desired notion of non-malleability. The complete definition appears in Figure 6.8.

Setup: On receiving a command SETUP from a party P: If a previous SETUP command has been processed, abort. Else, send (ID-REQ, P) to the adversary, and expect in response a string id. Broadcast (ID-ANNOUNCE, P, id) to all other parties.

Message posting: On receiving a command (POST, msg) from a party sender: If $\mathsf{msg} \notin \mathcal{M}$, ignore the request. If P is corrupt, send (HANDLE-REQ, sender, msg) to the adversary; otherwise send (HANDLE-REQ, sender) to the adversary. In both cases expect a string handle in return. If handle has been previously used, abort; else internally record (handle, msg) and broadcast (HANDLE-ANNOUNCE, handle) to all parties.

Dummy handles: On receiving a command (DUMMY, handle) from a *corrupt* party, internally record (handle, $\perp$) and broadcast (HANDLE-ANNOUNCE, handle) to all parties.

Homomorphic reposting: On receiving a command (REPOST, handle, T) from a party sender: If handle is not recorded internally or $T \notin \mathcal{T}$, ignore the request. Otherwise, suppose (handle, msg) is recorded internally. If $\mathsf{msg} \neq \perp$, then do the same as if (POST, $T(\mathsf{msg})$) were received. Otherwise, send (HANDLE-REQ, sender, handle, T) to the adversary and expect a string handle' in return. If handle' has been previously used, abort; else record (handle', $T(\mathsf{msg})$) internally and send (HANDLE-ANNOUNCE, handle') to all parties.

Message reading: On receiving a command (GET, handle) from party P (and only party P): If a record (handle, msg) is recorded internally, give msg to P; otherwise ignore this request.

Figure 6.8: UC ideal functionality $\mathcal{F}_{\text{HMP}}^{\mathcal{T}}$.

$\mathcal{F}_{\text{HMP}}^{\mathcal{T}}$ allows parties to post private messages for other parties, as on a bulletin board, represented by abstract *handles* which reveal no information about the message (they are generated by the adversary without knowledge of the message). Only the designated receiver is allowed to obtain the corresponding message for a handle. To model the homomorphic features, the functionality allows parties to post messages derived from other handles. The functionality is parameterized by the set of allowed transformations $\mathcal{T}$. When a party provides a previously posted handle and a transformation $T \in \mathcal{T}$, the functionality retrieves the message m corresponding to the handle and then acts as if the party had actually requested $T(m)$ to be posted. The sender does not need to know, nor is it told, the underlying message m of the existing handle.

$\mathcal{F}_{\text{HMP}}^{\mathcal{T}}$ models the non-malleability we require, since the *only* way a posted message can influence a subsequent message is via an allowed transformation.

The functionality also models unlinkability by internally behaving identically (in particular, in its interaction with the adversary) for the two different kinds of posts. The only exception is that corrupt parties may generate "dummy" handles which look like normal handles but do not contain any message. When a party derives a new handle from such a dummy handle, the adversary learns the transformation. This apparent slight weakness is natural[3] and it mirrors the tradeoff between our indistinguishability definitions. In our security proofs, this additional dummy handle feature is crucial.

Homomorphic encryption schemes and protocols for $\mathcal{F}_{\text{HMP}}^{\mathcal{T}}$. The UC framework defines when a protocol is said to *securely realize* the functionality $\mathcal{F}_{\text{HMP}}^{\mathcal{T}}$: for every PPT adversary in the real world interaction (using the protocol), there exists a PPT simulator in the ideal world interaction with $\mathcal{F}_{\text{HMP}}^{\mathcal{T}}$, such that no PPT environment can distinguish between the two interactions.

We associate homomorphic encryption schemes with candidate protocols for $\mathcal{F}_{\text{HMP}}^{\mathcal{T}}$ in the following natural way (for simplicity assume all communication is on an authenticated broadcast channel). To setup an instance of $\mathcal{F}_{\text{HMP}}^{\mathcal{T}}$, a party generates a key pair and broadcasts the public key. To post a message, a party encrypts it under the public key and broadcasts the resulting ciphertext. The "derived post" feature is implemented via the CTrans procedure. To retrieve a message from a handle, the receiver decrypts it using the private key.

6.4 Relationships Among Security Definitions

To justify our new security definitions, we prove some relationships among them and among the more established definitions of CCA, gCCA [1, 93], and RCCA [21] security. We first give some simple observations.

6.4.1 Simple Observations

In order to achieve HCCA security, $\mathcal{T}$ must be closed under composition, and must also contain the identity function, since the adversary can simply submit the challenge ciphertext ζ^* to the RigDec oracle.

Simultaneously achieving HCCA security and unlinkability. We have defined unlinkability and HCCA security with the goal that a scheme can be both $\mathcal{T}$-unlinkably homomorphic, and $\mathcal{T}$-HCCA-secure (for the same $\mathcal{T}$). Indeed, it is easy to see that if a scheme is $\mathcal{T}$-unlinkably homomorphic and $\mathcal{T}'$-HCCA-secure, then $\mathcal{T} \subseteq \mathcal{T}'$. For simplicity, and to highlight the compatibility and sharp tradeoff between these two definitions, we only focus on schemes which satisfy them both with respect to the same transformation space $\mathcal{T}$.

Triviality of HCCA without (weak) unlinkability or transformation-hiding. Without a requirement of (weak) unlinkability or transformation-hiding, it is relatively trivial to construct a scheme that is HCCA-secure with respect to *any* space of message transformations $\mathcal{T}$. Consider modifying any CCA-secure encryption scheme by considering an additional kind of ciphertext of the form (ζ, T), where ζ is a ciphertext in the original scheme and T is a description of a transformation in $\mathcal{T}$. To decrypt a ciphertext of this new form, first decrypt ζ and then if $T \in \mathcal{T}$, apply T to the result. The scheme has a homomorphic transformation procedure: $\mathsf{CTrans}(\zeta, T) = (\zeta, T)$, and $\mathsf{CTrans}((\zeta, T), T') = (\zeta, T' \circ T)$.

It is not hard to see that such a scheme achieves HCCA security with respect to $\mathcal{T}$. RigEnc should encrypt some fixed message and use the ciphertext itself as the auxiliary information S. Then on input $(\zeta, T), S$, the RigExtract procedure should return T if $T \in \mathcal{T}$ and $\zeta = S$, and return $\perp$ otherwise. Since T is included explicitly in the ciphertext itself, the scheme is clearly not even transformation-hiding.

6.4.2 HCCA Generalizes Existing Non-Malleability Definitions

Theorem 6.5. *CCA, gCCA, and RCCA security can be obtained as special cases of the HCCA definition, by appropriately restricting* RigEnc *and* RigExtract.

[3] For example, an adversary may broadcast a single encryption under a public key that he keeps hidden. The ciphertext will be meaningless to the recipient, but if the adversary later encounters another ciphertext that decrypts under this same key, he can deduce that the it was derived from his previous ciphertext.

Proof. The restrictions on RigExtract are progressively relaxed as we go from CCA to gCCA to RCCA, making it explicit that the non-malleability requirements get weaker in that order.

First, consider a variant of the traditional CCA security experiment (Figure 6.2), in which the adversary provides only one of the two challenge plaintexts msg_1, while the other challenge plaintext msg_0 is fixed and publicly known. This variant realizes the same level of security as the original CCA definition (in which the adversary chooses both plaintexts).[4] We can further modify the experiment cosmetically so that when the adversary submits the challenge ciphertext to the decryption oracle, the response is msg_1 (regardless of whether msg_0 or msg_1 was chosen), instead of "GUARDED".

This modified CCA experiment can be directly obtained as a special case of IND-HCCA as follows: RigEnc generates an encryption of msg_0, and uses the ciphertext itself as the auxiliary information. RigExtract simply checks if an input ciphertext is identical to the auxiliary information; if so, it outputs the identity transformation (indicating that the given ciphertext encodes the same plaintext as the output of RigEnc); otherwise, it outputs $\perp$. This RigExtract does not use the private key at all, thus the corresponding IND-HCCA experiment can be simplified to the IND-HCCA-simp experiment without loss of generality. Then it is easy to see that the resulting IND-HCCA-simp experiment is exactly equivalent to the modified CCA experiment described above, with RigDec acting as the decryption oracle.

Similarly, benignly non-malleable (or gCCA) security is obtained if RigExtract is allowed to compute an arbitrary equivalence relation among the two ciphertexts (but still without being given the private key). RCCA security is obtained if RigEnc encrypts a random plaintext (using that plaintext as the auxiliary information), and RigExtract simply decrypts the given ciphertext and checks whether the result equals the auxiliary information. In this case, RigExtract only uses the private key to implement Dec as a black box, so again the corresponding IND-HCCA experiment collapses to the IND-HCCA-simp experiment.

Note that in all these cases RigExtract is allowed to output only $\perp$ or the identity transformation. This highlights the fact that schemes satisfying these security definitions are not malleable in ways which alter the message. □

We note that all of these special cases of HCCA involve a RigEnc procedure which simply encrypt a plaintext as normal. In our construction (Section 6.5.2), we exploit the flexibility of the full HCCA definition to achieve larger classes of transformations, by letting RigEnc generate a "ciphertext" that is not in the range of $\mathsf{Enc}(\cdot)$.

Rerandomizable RCCA. The notion of *rerandomizable* RCCA encryption was introduced in [21] and later considered by [45, 83]. Briefly, rerandomizable RCCA security demands that given any ciphertext in the support of $\mathsf{Enc}(m)$, anyone should be able to freshly sample the distribution of $\mathsf{Enc}(m)$ (rerandomizability), but the scheme is non-malleable in ways which alter the plaintext (RCCA security).

Rerandomizable RCCA security corresponds to the special case of unlinkable HCCA security, where the only allowed transformation is the identity transformation,[5] and we will use the term "rerandomizable RCCA" security when appropriate. However, for the general case, we prefer the term "unlinkable", as it emphasizes a concrete security end goal, and not the technique used to achieve it.

6.4.3 CCA From HCCA

Given a (not necessarily unlinkable) HCCA-secure scheme satisfying a reasonable condition, we show a black-box construction of a CCA-secure scheme. The reduction is a simple modification of the Canetti-Halevi-Katz (CHK) transformation to construct a CCA-secure scheme from any identity-based encryption (IBE) scheme [18]. A similar black-box transformation appeared independently in [71]. The CHK transformation relies on the fact that IBE ciphertexts are non-malleable with respect to the identity string. Using the verification key of a strong one-time signature scheme as the identity, the CHK transformation thwarts any potential malleability of the IBE ciphertext's message. Similarly, if the homomorphic transformations of an HCCA-secure scheme preserve *any* part of the plaintext, then that part of the plaintext is non-malleable and can be used to store a signature verification key, as in the CHK transformation, to thwart the possible malleability of the remainder of the plaintext.

Let $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{CTrans})$ be a $\mathcal{T}$-HCCA-secure scheme with the following properties:

- The message space $\mathcal{M}$ is isomorphic to $A \times B$. That is, there are efficiently computable maps between $\mathcal{M}$ and $A \times B$. Without loss of generality, we assume $\mathcal{M} = A \times B$.
- For all $T \in \mathcal{T}$, there exists a function $t : B \to B$ such that $T(a, b) = (a, t(b))$. That is, each transformation preserves the A-component of the plaintext.

[4] If an adversary can successfully distinguish between encryptions of msg versus msg' in the CCA experiment, then a related adversary can successfully distinguish between either $(\mathsf{msg}, \mathsf{msg}_0)$ or $(\mathsf{msg}', \mathsf{msg}_0)$.

[5] Technically, our unlinkability requirement is slightly stronger than the rerandomizability requirement as stated above.

Intuitively, these are the minimal requirements so that $\mathcal{E}$ enforces non-malleability on *some* part of the message. These requirements are sufficient to adapt the CHK transformation.

Let $\Sigma = (\mathsf{SigGen}, \mathsf{Sign}, \mathsf{Ver})$ be a strong one-time signature scheme[6] with A as its space of verification keys. We define the new CCA-secure scheme $\mathcal{E}^{\text{CHK}}$, with message space B, as follows:

- $\mathsf{KeyGen}^{\text{CHK}}$: same as KeyGen.
- $\mathsf{Enc}^{\text{CHK}}_{PK}(\mathsf{msg})$: Run $(vk, ssk) \leftarrow \mathsf{SigGen}$. Compute $\zeta \leftarrow \mathsf{Enc}_{PK}(vk, \mathsf{msg})$ and $\sigma \leftarrow \mathsf{Sign}_{ssk}(\zeta)$, then output (vk, ζ, σ).
- $\mathsf{Dec}^{\text{CHK}}_{SK}(vk, \zeta, \sigma)$: If $\mathsf{Ver}_{vk}(\zeta, \sigma) \neq 1$, then output $\perp$. Else, compute $(vk', \mathsf{msg}) \leftarrow \mathsf{Dec}_{SK}(\zeta)$. If the decryption fails, or if $vk \neq vk'$, then output $\perp$. Otherwise, output msg.

Theorem 6.6. *If $\mathcal{E}$ and Σ are as above, then $\mathcal{E}^{\text{CHK}}$ is CCA-secure.*

Proof. Consider the standard CCA experiment (Figure 6.2) against $\mathcal{E}^{\text{CHK}}$, in which the adversary receives an encryption of $\mathsf{msg}^* = \mathsf{msg}_b$. Now consider a hybrid experiment in which the challenge ciphertext is generated as follows:

- $\mathsf{Enc}^*_{PK}(\mathsf{msg}^*)$: Run $(vk^*, ssk) \leftarrow \mathsf{SigGen}$. Compute $(\zeta^*, S^*) \leftarrow \mathsf{RigEnc}_{PK}$ and $\sigma^* \leftarrow \mathsf{Sign}_{ssk}(\zeta^*)$, then output $(vk^*, \zeta^*, \sigma^*)$.

and the decryption oracle is replaced by:

- $\mathsf{Dec}^*_{SK}(vk, \zeta, \sigma)$: If $\mathsf{Ver}_{vk}(\zeta, \sigma) \neq 1$, then output $\perp$. If $\perp \neq T \leftarrow \mathsf{RigExtract}_{SK}(\zeta, S^*)$, then set $(vk', \mathsf{msg}) = T(vk^*, \mathsf{msg}^*)$. By the properties of $\mathcal{E}$, $vk' = vk^*$ in this case. Otherwise set $(vk', \mathsf{msg}) \leftarrow \mathsf{Dec}_{SK}(\zeta)$. If the decryption fails, or if $vk \neq vk'$, then output $\perp$. Otherwise, output msg.

By the HCCA security of $\mathcal{E}$, this hybrid experiment is indistinguishable from the original experiment (to simulate the CCA experiment and this hybrid variant in the IND-HCCA experiment requires no access to the GRigEnc or GRigExtract oracles). Note that in the hybrid experiment, the challenge ciphertext is generated independently of the message msg^*, but the hybrid decryption oracle still depends on msg^*.

However, in the case that $\mathsf{RigExtract}_{SK}(\zeta, S^*) \neq \perp$, we have that vk' will be set to vk^*. But since the decryption oracle is only called when $(vk, \zeta, \sigma) \neq (vk^*, \zeta^*, \sigma^*)$, we must have either $vk' \neq vk$, or $\mathsf{Ver}_{vk'}(\zeta, \sigma) = 0$ with overwhelming probability by the strong unforgeability of Σ. Thus the experiment is indistinguishable from one in which the decryption oracle rejects whenever $\mathsf{RigExtract}_{SK}(\zeta, S^*) \neq \perp$. That is:

- $\mathsf{Dec}^*_{SK}(vk, \zeta, \sigma)$: If $\mathsf{Ver}_{vk}(\zeta, \sigma) \neq 1$, or if $\mathsf{RigExtract}_{SK}(\zeta; S^*) \neq \perp$, then output $\perp$. Set $(vk', \mathsf{msg}) \leftarrow \mathsf{Dec}_{SK}(\zeta)$. If the decryption fails, or if $vk \neq vk'$, then output $\perp$. Otherwise, output msg.

Now in this final hybrid, the challenge ciphertext and decryption oracle responses are computed independently of msg^*, so the adversary's advantage is 1/2. By the indistinguishability of our sequence of hybrids, the scheme is CCA-secure. □

We also note that if Σ is not *strongly* unforgeable, then the above transformation results in an RCCA-secure scheme instead of a CCA-secure scheme.

Given an HCCA-secure scheme $\mathcal{E}$ with message space isomorphic to $A \times B$, and all allowed transformations preserving the A-component, the above construction gives a CCA-secure scheme with message space B. We also observe that one can also construct an RCCA-secure [21] scheme with message space A using $\mathcal{E}$: simply add an arbitrary B-component when encrypting and ignore the B-component when decrypting.

Black-box separation from CPA. Given the black-box separation results of [39], our construction above implies that there is no shielding[7] black-box construction of a HCCA-secure scheme satisfying the condition of Theorem 6.6 from a CPA-secure scheme.

[6] Strong one-time signature schemes are implied by one-way functions, and it is easy to construct a one-way function from the KeyGen procedure of any HCCA-secure scheme.

[7] A shielding construction is one in which the HCCA scheme's decryption algorithm does not make calls to the CPA scheme's encryption algorithm.

6.4.4 Restricting the Transformation Space

In general, one cannot easily modify a $\mathcal{T}_1$-unlinkable-HCCA-secure scheme into a $\mathcal{T}_2$-unlinkable-HCCA-secure scheme, even if $\mathcal{T}_2 \subseteq \mathcal{T}_1$. The problem of "disabling" the transformations in $\mathcal{T}_1 \setminus \mathcal{T}_2$ while at the same time maintaining those in $\mathcal{T}_2$ appears just as challenging as constructing a $\mathcal{T}_2$-unlinkable-HCCA scheme from scratch. However, a simple black-box transformation is possible for the special case where $\mathcal{T}_2$ is a singleton set containing only the identity transformation. Recall that this special case is known as *rerandomizable RCCA security* [21].

Definition 6.7. *Let $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{CTrans})$ be a unary homomorphic encryption scheme, and let $\mathcal{E}' = (\mathsf{KeyGen}', \mathsf{Enc}', \mathsf{Dec}')$ be a (not necessarily homomorphic) encryption scheme. We define the* encapsulation of $\mathcal{E}'$ inside $\mathcal{E}$*, denoted $\mathcal{E} \circ \mathcal{E}'$, to be a unary homomorphic encryption scheme, given by the following algorithms:*

- KeyGen^**: Run $(pk, sk) \leftarrow \mathsf{KeyGen}$ and $(pk', sk') \leftarrow \mathsf{KeyGen}'$. Output $PK = (pk, pk')$ and $SK = (sk, sk')$.*
- $\mathsf{Enc}^*_{pk,pk'}(\mathsf{msg}) = \mathsf{Enc}_{pk}(\mathsf{Enc}'_{pk'}(\mathsf{msg}))$.
- $\mathsf{Dec}^*_{sk,sk'}(\zeta) = \mathsf{Dec}'_{sk'}(\mathsf{Dec}_{sk}(\zeta))$*, where we let $\mathsf{Dec}'_{sk'}(\perp) = \perp$ for simplicity.*
- CTrans^**: same as* CTrans*.*

Theorem 6.8. *If $\mathcal{E}^H$ is a $\mathcal{T}$-unlinkable-HCCA-secure scheme (for any $\mathcal{T}$) and $\mathcal{E}^R$ is a (not necessarily rerandomizable) RCCA-secure scheme, then $\mathcal{E}^H \circ \mathcal{E}^R$ is rerandomizable RCCA-secure.*

Intuitively, the outer scheme's unlinkability is preserved by the encapsulation, but the inner scheme's non-malleability renders useless all transformations but the identity transformation.

Note that RCCA security without rerandomizability is a weaker requirement than CCA security [21]. Thus, for example, an unlinkable HCCA-secure scheme encapsulating a *plain* CCA-secure encryption scheme will yield a rerandomizable RCCA-secure encryption scheme. For many instantiations of HCCA-secure schemes, there is a black-box reduction to a CCA-secure scheme (Theorem 6.6), and in these cases we get a direct black-box reduction from unlinkable HCCA security to rerandomizable RCCA security.

Proof. For clarity, we superscript with "H" the algorithms of $\mathcal{E}^H$, and superscript with "R" the algorithms of $\mathcal{E}^R$. We write the keys of $\mathcal{E}^H$ as (hpk, hsk), and similarly the keys of $\mathcal{E}^R$ as (rpk, rsk). We write the algorithms of the encapsulated scheme $\mathcal{E}^H \circ \mathcal{E}^R$ without superscripts.

Note that CTrans^H accepts possibly many allowed transformations as input when viewed in the context of $\mathcal{E}^H$. However, in the context of the encapsulated scheme $\mathcal{E}^H \circ \mathcal{E}^R$, $\mathsf{CTrans} = \mathsf{CTrans}^H$ is only meaningful when called with the identity transformation. We note that to achieve HCCA security with respect to $\mathcal{T}$, $\mathcal{T}$ must indeed contain the identity function (see Section 6.4.1). It is easy to see that the unlinkability of the outer scheme (with respect to the identity transformation) is preserved by the construction.

To show RCCA security (HCCA security with the identity function as the only allowed transformation), we must demonstrate appropriate RigEnc and RigExtract procedures for the new scheme. Let $(\mathsf{RigEnc}^H, \mathsf{RigExtract}^H)$ and $(\mathsf{RigEnc}^R, \mathsf{RigExtract}^R)$ be the procedures guaranteed by the two schemes, respectively. Then the new scheme satisfies HCCA security with the following procedures:

- $\mathsf{RigEnc}_{hpk,rpk}$ does the following: Run $(\zeta, S_H) \leftarrow \mathsf{RigEnc}^H_{hpk}$ and $(\zeta_R, S_R) \leftarrow \mathsf{RigEnc}^R_{rpk}$. Set $S = (S_H, \zeta_R, S_R)$ and output (ζ, S).
- $\mathsf{RigExtract}_{hsk,rsk}(\zeta, S)$ does the following: Parse S as (S_H, ζ_R, S_R). Run $T \leftarrow \mathsf{RigExtract}^H_{hsk}(\zeta, S_H)$. If $T = \perp$, output $\perp$; otherwise output $\mathsf{RigExtract}^R_{rsk}(T(\zeta_R), S_R)$, which must be either the identity function or $\perp$, by the RCCA security of the inner scheme.

Consider a hybrid HCCA experiment where the challenge ciphertext is generated from msg^* as:

- Run $(\zeta^*, S^*) \leftarrow \mathsf{RigEnc}^H_{hpk}$ and $\zeta^*_R \leftarrow \mathsf{Enc}^R_{rpk}(\mathsf{msg}^*)$. Remember ζ^*_R and output ζ^*.

and RigDec in Phase II of the IND-HCCA experiment is implemented as:

$$\mathsf{RigDec}_{hsk,rsk}(\zeta) = \begin{cases} \mathsf{Dec}^R_{rsk}(T(\zeta^*_R)) & \text{if } \perp \neq T \leftarrow \mathsf{RigExtract}^H_{hsk}(\zeta, S^*) \\ \mathsf{Dec}^R_{rsk}(\mathsf{Dec}^H_{hsk}(\zeta)) & \text{otherwise.} \end{cases}$$

It is straight-forward to verify that this hybrid experiment is indistinguishable from both the $b = 0$ and $b = 1$ branches of the HCCA experiment instantiated with the new scheme and its RigEnc and RigExtract procedures described above. □

6.4.5 Unlinkable HCCA Implies the UC Definition

Theorem 6.9. *Every $\mathcal{T}$-homomorphic encryption scheme which is HCCA-secure, unlinkably homomorphic (with respect to $\mathcal{T}$) and satisfies the correctness properties, is a UC-secure realization of $\mathcal{F}_{\text{HMP}}^{\mathcal{T}}$, against static (non-adaptive) corruptions.*

Let $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{CTrans})$ be an unlinkably homomorphic, HCCA-secure encryption scheme (with allowable homomorphisms $\mathcal{T}$). To prove Theorem 6.9, for any real-world adversary $\mathcal{A}$, we must demonstrate an ideal-world adversary (simulator) $\mathcal{S}$, so that for all PPT environments $\mathcal{Z}$, $\text{EXEC}[\mathcal{Z}, \mathcal{A}, \mathcal{E}, \mathcal{F}_{\text{BCAST}}] \approx \text{EXEC}[\mathcal{Z}, \mathcal{S}, \pi_{\mathsf{dummy}}, \mathcal{F}_{\text{HMP}}^{\mathcal{T}}]$. We assume that all communication is done on an authenticated broadcast channel $\mathcal{F}_{\text{BCAST}}$.

In the case where the recipient P is corrupt, the simulation is trivial. Each time it is asked to generate a handle, it is given the underlying message. Each time the adversary itself outputs a ciphertext, the simulator can register it as a dummy handle, after which it is notified each time that handle is REPOST'ed. We now focus on the case where P is not corrupt.

We construct $\mathcal{S}$ in a sequence of hybrids, starting from the real-world interactions and altering it step by step to get an ideal-world adversary, at every stage ensuring that each change remains indistinguishable to all environments. All the simulators below exist in the ideal world, but are also given (progressively less) information about the inputs to the honest parties. We conveniently model this access to extra information using modified functionalities.

$\mathcal{S}_0$ and $\mathcal{F}_0$ (correctness): $\mathcal{F}_0$ behaves exactly like $\mathcal{F}_{\text{HMP}}^{\mathcal{T}}$ except that in its HANDLE-REQ interactions with the adversary, it reveals the message and whether the handle is being requested for a repost. Thus $\mathcal{S}_0$ effectively learns all the honest parties' inputs to $\mathcal{F}_0$. $\mathcal{S}_0$ internally simulates the encryption scheme algorithms for all honest parties, and lets the adversary $\mathcal{A}$ interact with these simulated parties and directly with the environment, as follows:

1. When an honest party P sends a SETUP command to $\mathcal{F}_0$, the functionality sends $(\text{ID-REQ}, P)$ to $\mathcal{S}_0$ and expects an ID in return. $\mathcal{S}_0$ generates a key pair $(PK, SK) \leftarrow \mathsf{KeyGen}$ and uses PK as the ID string. It also internally simulates to $\mathcal{A}$ that P broadcast the public key.

2. When an honest party sender sends a command $(\text{POST}, \mathsf{msg})$ to $\mathcal{F}_0$, the functionality sends $(\text{HANDLE-REQ}, \mathsf{sender}, \mathsf{msg})$ to $\mathcal{S}_0$ and expects a handle in return. $\mathcal{S}_0$ computes $\mathsf{handle} \leftarrow \mathsf{Enc}_{PK}(\mathsf{msg})$ and uses it as the handle. It also internally simulates to $\mathcal{A}$ that sender broadcast handle.

3. When an honest party sender sends a command $(\text{REPOST}, \mathsf{handle}, T)$ to $\mathcal{F}_0$, and handle is internally recorded, the functionality sends $(\text{HANDLE-REQ}, \mathsf{sender}, \mathsf{handle}, T)$ to $\mathcal{S}_0$ and expects a handle in return. $\mathcal{S}_0$ computes $\mathsf{handle}' \leftarrow \mathsf{CTrans}(\mathsf{handle}, T)$ and uses it as the handle. It also internally simulates to $\mathcal{A}$ that sender broadcast handle'.

4. When the adversary broadcasts a ciphertext ζ, $\mathcal{S}_0$ does the following:

   - If $\mathsf{Dec}_{SK}(\zeta) = \mathsf{msg} \neq \perp$, then $\mathcal{S}_0$ sends $(\text{POST}, \mathsf{msg})$ to the functionality on behalf of $\mathcal{A}$. It uses ζ as the corresponding handle.
   - Otherwise, $\mathcal{S}_0$ sends (DUMMY, ζ) to $\mathcal{F}_0$.

Claim 6.10. *For any given PPT adversary, let $\mathcal{F}_0$ and $\mathcal{S}_0$ be as described above. Then for all PPT environments $\mathcal{Z}$,* $\text{EXEC}[\mathcal{Z}, \mathcal{A}, \mathcal{E}, \mathcal{F}_{\text{BCAST}}] \approx \text{EXEC}[\mathcal{Z}, \mathcal{S}_0, \pi_{\mathsf{dummy}}, \mathcal{F}_0]$.

Proof. This follows from the correctness properties of encryption scheme $\mathcal{E}$, and the fact that $\mathcal{S}_0$ exactly emulates the real world actions of all parties. □

$\mathcal{S}_1$ and $\mathcal{F}_1$ (unlinkable homomorphism): $\mathcal{F}_1$ is identical to $\mathcal{F}_0$ except that it does not tell the adversary whether a HANDLE-REQ was the result of a POST or REPOST command, except for dummy handles. The exact differences are as follows:

1. When an honest party sender sends a command $(\text{REPOST}, \mathsf{handle}, T)$ to $\mathcal{F}_1$, and $(\mathsf{handle}, \mathsf{msg})$ is internally recorded, and $\mathsf{msg} \neq \perp$, the functionality now does the same thing as if sender had given the command $(\text{POST}, \mathsf{id}, T(\mathsf{msg}))$ command. If $\mathsf{msg} = \perp$, the functionality sends $(\text{HANDLE-REQ}, \mathsf{sender}, \mathsf{handle}, T)$ to the simulator just as in $\mathcal{F}_0$.

2. $\mathcal{S}_1$ and $\mathcal{S}_0$ are identical, although they receive different types of HANDLE-REQ requests when interacting with $\mathcal{F}_1$ instead of $\mathcal{F}_0$.

Claim 6.11. *For any given PPT adversary $\mathcal{A}$, let $\mathcal{S}_0$, $\mathcal{F}_0$, $\mathcal{S}_1$ and $\mathcal{F}_1$ be as described above. Then for all PPT environments $\mathcal{Z}$,* $\text{EXEC}[\mathcal{Z}, \mathcal{S}_0, \pi_{\mathsf{dummy}}, \mathcal{F}_0] \approx \text{EXEC}[\mathcal{Z}, \mathcal{S}_1, \pi_{\mathsf{dummy}}, \mathcal{F}_1]$.

Proof. This follows from the unlinkable homomorphism property of the encryption scheme. The only manner in which the adversary's view differs in the two executions is in whether certain ciphertexts are generated via a transformation (just as $\mathcal{S}_0$ does on receiving a $(\text{HANDLE-REQ}, \mathsf{sender}, \mathsf{handle}, T)$ request) or as a fresh encryptions of the appropriate message (just as $\mathcal{S}_1$ does on receiving a $(\text{HANDLE-REQ}, \mathsf{sender}, T(\mathsf{msg}))$ request). We note that $\mathcal{F}_1$ only behaves differently when $\mathsf{msg} \neq \perp$. Thus the difference between executions only involves ciphertexts which were either honestly generated by the simulator, or adversarially generated ciphertexts that successfully decrypted under PK. The unlinkable homomorphism property of the scheme implies that the difference in these interactions' outcomes is negligible.

More formally, we can only apply the unlinkable homomorphism property to one ciphertext at a time. It is straightforward to construct a sequence of hybrid simulators where the difference between successive hybrids is in whether a single handle was freshly re-encrypted or had a transformation applied, and such that the rest of the interaction can be carried out within the unlinkability experiment. Thus $\text{EXEC}[\mathcal{Z}, \mathcal{S}_0, \pi_{\mathsf{dummy}}, \mathcal{F}_0] \approx \text{EXEC}[\mathcal{Z}, \mathcal{S}_1, \pi_{\mathsf{dummy}}, \mathcal{F}_1]$. □

$\mathcal{S}_2$ and $\mathcal{F}_2$ (HCCA security): $\mathcal{F}_2$ is identical to $\mathcal{F}_1$ except that it does not tell the adversary the contents of the posted messages when the receiver is not corrupted. $\mathcal{S}_2$ differs from $\mathcal{S}_1$ accordingly, and uses the RigEnc and RigExtract features guaranteed by HCCA security. The exact differences are as follows:

1. When P is not corrupt and $\mathcal{F}_1$ would send $(\text{HANDLE-REQ}, \mathsf{sender}, \mathsf{msg})$ to the simulator (i.e, when a party posts or reposts a non-dummy handle), $\mathcal{F}_2$ instead sends $(\text{HANDLE-REQ}, \mathsf{sender})$.
2. When $\mathcal{S}_2$ receives a request of the form $(\text{HANDLE-REQ}, \mathsf{sender})$ from $\mathcal{F}_2$, it computes $(\mathsf{handle}, S) \leftarrow \mathsf{RigEnc}_{PK}$ and uses handle as the message's handle. It internally keeps track of (handle, S) for later use.
3. When the adversary broadcasts a ciphertext ζ, $\mathcal{S}_2$ does the following: For each (handle, S) recorded above, $\mathcal{S}_2$ checks if $\mathsf{RigExtract}_{SK}(\zeta, S) = T \neq \perp$. If so, $\mathcal{S}_2$ sends $(\text{REPOST}, \mathsf{handle}, T)$ to $\mathcal{F}_2$ and uses ζ as the handle. If none of these RigExtract calls succeed, then $\mathcal{S}_2$ proceeds just as $\mathcal{S}_1$ (i.e, attempts to decrypt ζ under SK and so on).

Claim 6.12. *For any given PPT adversary $\mathcal{A}$, let $\mathcal{S}_1$, $\mathcal{F}_1$, $\mathcal{S}_2$ and $\mathcal{F}_2$ be as described above. Then for all PPT environments $\mathcal{Z}$,* $\text{EXEC}[\mathcal{Z}, \mathcal{S}_1, \pi_{\mathsf{dummy}}, \mathcal{F}_1] \approx \text{EXEC}[\mathcal{Z}, \mathcal{S}_2, \pi_{\mathsf{dummy}}, \mathcal{F}_2]$.

Proof. This follows from the HCCA security of the scheme. Intuitively, the only way the two executions differ is in whether the simulator provides honest ciphertexts (as $\mathcal{S}_1$ does) or "rigged" ciphertexts (as $\mathcal{S}_2$ does).

More formally, we can only apply the HCCA guarantee to one ciphertext at a time. It is straightforward to construct a sequence of hybrid simulators where the difference between successive hybrids is in whether a single handle was encrypted with the correct message or else via RigEnc, and such that the rest of the interaction can be carried out within the HCCA experiment. We note that in most hybrids, there must be calls to RigEnc and RigExtract for other ciphertexts, so the GRigEnc and GRigExtract oracles are crucial in the HCCA definition. □

Concluding the proof. Combining the above claims we get that for all adversaries $\mathcal{A}$, there exists a simulator $\mathcal{S}_2$ such that $\text{EXEC}[\mathcal{Z}, \mathcal{A}, \mathcal{E}, \mathcal{F}_{\text{BCAST}}] \approx \text{EXEC}[\mathcal{Z}, \mathcal{S}, \pi_{\mathsf{dummy}}, \mathcal{F}_2]$ for all environments $\mathcal{Z}$. Note that $\mathcal{F}_2$ is in fact identical to $\mathcal{F}_{\text{HMP}}^{\mathcal{T}}$. So letting $\mathcal{S} = \mathcal{S}_2$ completes the proof.

Weak unlinkability. If a scheme achieves only weak unlinkability (Definition 6.3), then a slight relaxation of the UC functionality can realized. In the $\mathcal{F}_{\text{HMP}}^{\mathcal{T}}$ UC functionality, call a handle *adversarially influenced* if:

- it is the result of a POST or REPOST command issued by a corrupted party,
- or it is the result of a $(\text{REPOST}, \mathsf{handle})$ command, where handle is adversarially influenced.

An encryption scheme which is HCCA secure and only weakly unlinkably homomorphic is a secure realization of a variant of $\mathcal{F}_{\text{HMP}}^{\mathcal{T}}$, in which the adversary is notified every time an adversarially influenced handle is reposted (in the same way it is notified when its dummy handles are reposted). The proof is very similar to that of Theorem 6.9, except that unlinkability is only applied to handles which are not adversarially influenced (i.e., a ciphertext which was generated by the simulator honestly using Enc).

6.5 Constructions

We now present several constructions of various strengths, and supporting different classes of homomorphic operations.

6.5.1 Achieving Transformation Hiding

Our first construction is an encryption scheme which is HCCA secure with respect to *any* unary group operation, and achieves the transformation hiding property.

The construction. Let $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be any RCCA-secure encryption scheme with message space $\mathbb{G}$, which is an abelian group with group operation "$*$". Without loss of generality, we assume that $\mathbb{G}$ isomorphic to the direct product $\mathbb{H}_1 \times \mathbb{H}_2$, where $\mathbb{H}_1$ is a parameter of our scheme, and elements in $\mathbb{G}$ are represented as $(m_1, m_2) \in \mathbb{H}_1 \times \mathbb{H}_2$.[8]

For $t \in \mathbb{H}_1$, let T_t denote the "multiplication by t" transformation $T_t(m_1, m_2) = (t * m_1, m_2)$. For simplicity, we let $T_t(\perp) = \perp$.

Our construction has message space $\mathcal{M}$, and supports transformations $\mathcal{T} = \{T_t \mid t \in \mathbb{H}_1\}$. The construction is specified by the following algorithms:

- KeyGen^*: Same as KeyGen.
- $\mathsf{Enc}^*_{PK}(m_1, m_2)$: Choose random $r \leftarrow \mathbb{H}_1$, and set $s = m_1 * r \in \mathbb{H}_1$. Output $(\mathsf{Enc}_{PK}(r, m_2), s)$.
- $\mathsf{Dec}^*_{SK}(\zeta, s)$: Decrypt $(r, m_2) \leftarrow \mathsf{Dec}_{SK}(\zeta)$, and output $\perp$ if the decryption fails. Otherwise, output $(s * r^{-1}, m_2)$.
- $\mathsf{CTrans}^*((\zeta, s), T_t)$: Output $(\zeta, s * t)$.

Intuitively, our desired transformations preserve the $\mathbb{H}_2$-component of the plaintext, but allow the group operation to be applied to the $\mathbb{H}_1$-component. Thus our encryption scheme places the $\mathbb{H}_2$-component of the plaintext into the RCCA-secure encryption. We also place inside the RCCA encryption a random one-time pad that masks the $\mathbb{H}_1$-component of the plaintext. The masked $\mathbb{H}_1$-component is provided in the clear of the ciphertext, so that anyone can apply the appropriate algebraic operations to it.

Theorem 6.13. *The above construction is HCCA-secure with respect to $\mathcal{T}$, and is transformation-hiding.*

Proof. First, it is easy to see that the scheme is transformation-hiding, since the distributions of $\mathsf{Enc}^*_{PK}(T_t(m_1, m_2))$ and $\mathsf{CTrans}^*(\mathsf{Enc}^*_{PK}(m_1, m_2), T_t)$ are identical.

To show that the scheme is HCCA-secure, we must demonstrate appropriate RigEnc and $\mathsf{RigExtract}$ procedures. Let RigEnc and $\mathsf{RigExtract}$ be the procedures guaranteed by the RCCA security of the component scheme $\mathcal{E}$. We define the following procedures:

- RigEnc^*_{PK}: Choose random $s \leftarrow \mathcal{T}_1$ and compute $(\zeta, S) \leftarrow \mathsf{RigEnc}_{PK}$. Output (ζ, s) and private information (S, s).
- $\mathsf{RigExtract}^*_{SK}((\zeta', s'); (S, s))$: If $\mathsf{RigExtract}_{SK}(\zeta'; S) = \perp$, then output $\perp$. Otherwise, the output of $\mathsf{RigExtract}_{SK}(\zeta'; S)$ is the identity transformation, by the RCCA security of $\mathcal{E}$. In that case, output $t = s' * s^{-1}$.

Our definition of $\mathsf{RigExtract}^*$ does not use the private key except to implement Dec^* as a black box, so we can consider the simplified HCCA experiment, IND-HCCA-simp.

We must prove that the two branches of the IND-HCCA-simp experiment are indistinguishable. First consider the branch $b = 1$ of the experiment that involves RigEnc^* and RigDec. We can equivalently write the branch as follows:

- Generating the challenge ciphertext: Given message (m_1^*, m_2^*), choose random $r \leftarrow \mathcal{T}_1$ and set $s = m_1^* * r$. Compute $(\zeta^*, S) \leftarrow \mathsf{RigEnc}_{PK}$, and output (ζ^*, s).
- Implementation of the $\mathsf{RigDec}(\zeta', s')$ oracle: If $\mathsf{RigExtract}_{SK}(\zeta'; S) \neq \perp$, then compute $t = s' * s^{-1}$. Then output $(m_1^* * t, m_2^*)$. Otherwise, set $(r', m_2') \leftarrow \mathsf{Dec}_{SK}(\zeta')$. If this decryption fails, output $\perp$; otherwise output $(s' * (r')^{-1}, m_2')$.

[8] For example, one may choose $\mathbb{H}_1 = \mathbb{G}$ and $\mathbb{H}_2 = \{1\}$, leading to a useful instantiation of our scheme.

We have simply filled in the steps of the IND-HCCA-simp experiment, but generated the value s slightly differently than in RigEnc^*. However, the value s is still uniform in $\mathbb{H}_1$, so this does not affect the outcome of the experiment.

However, suppose we modify this branch as follows: Let ζ^* be instead generated via $\mathsf{Enc}_{PK}(r, m_2^*)$; and in RigDec, we remove the condition that checks RigExtract. By the RCCA security of $\mathcal{E}$, this difference is indistinguishable, since whenever $\mathsf{RigExtract}_{SK}(\zeta'; S) \neq \perp$ in the RigDec algorithm above, we have that $\mathsf{Dec}_{SK}(\zeta') = (r, m_2^*)$ in the modified interaction, so the other clause of RigDec will give a consistent answer.

But in this modified interaction, the challenge ciphertext is generated according to the honest Enc^* procedure, and RigDec is implemented exactly as Dec^*. Thus the modified experiment is exactly the $b = 0$ branch of the IND-HCCA-simp experiment. We established that the two branches of the experiment are indistinguishable, thus the scheme is HCCA secure. □

6.5.2 Achieving Full Unlinkability

Our main result is a parameterized constructions which achieves both HCCA security and unlinkable homomorphism, with respect to a wide range of transformations, under the standard DDH assumption in two related groups.

Requirements. Our construction requires two (multiplicative) cyclic groups with a specific relationship: $\mathbb{G}$ of prime order p, and $\widehat{\mathbb{G}}$ of prime order q, where $\widehat{\mathbb{G}}$ is a subgroup of $\mathbb{Z}_p^*$. We require the DDH assumption to hold in both groups (with respect to the same security parameter). Given a sequence of primes $q, 2q+1, 4q+3$ (a *Cunningham chain* of the first kind of length 3 [2]), the two quadratic-residue groups $\widehat{\mathbb{G}} = \mathbb{QR}_{2q+1}^*$ and $\mathbb{G} = \mathbb{QR}_{4q+3}^*$, in which the DDH assumption is believed to hold, represent a suitable choice.

Notation and supported transformations. Let "$*$" denote the group operation in the product group $\mathbb{G}^n$ defined by $(\alpha_1, \ldots, \alpha_n) * (\beta_1, \ldots, \beta_n) = (\alpha_1\beta, \ldots \alpha_n\beta_n)$.

For $\tau \in \mathbb{G}^n$, define T_τ to be the "multiplication by τ" transformation in $\mathbb{G}^n$; i.e., $T_\tau(m) = \tau * m$. We also let $T_\tau(\perp) = \perp$ for simplicity. Now let $\mathbb{H}$ be a subgroup of $\mathbb{G}^n$. Our construction provides a scheme whose message space is $\mathbb{G}^n$, and whose set of allowable transformations is $\mathcal{T}_\mathbb{H} = \{T_\tau \mid \tau \in \mathbb{H}\}$. By choosing $\mathbb{H}$ appropriately, we can obtain the following notable classes $\mathcal{T}_\mathbb{H}$:

- The identity function alone (i.e., rerandomizable RCCA security), by setting $\mathbb{H} = \{1\}$.
- All transformations T_τ (that is, all component-wise multiplications in $\mathbb{G}^n$), by setting $\mathbb{H} = \mathbb{G}^n$.
- "Scalar multiplication" of tuples in $\mathbb{G}^n$ by coefficients in $\mathbb{G}$, by setting $\mathbb{H} = \{(s, \ldots, s) \mid s \in \mathbb{G}\}$.

Double-strand malleable encryption scheme. We now define a homomorphic encryption scheme which we call the "Double-strand malleable encryption" (DSME), and which we use as a component in our main construction.

System parameters: A cyclic multiplicative group $\widehat{\mathbb{G}}$ of prime order q. $\widehat{\mathbb{G}}$ also acts as the message space for this scheme. Similar to above, we denote T_σ as the multiplication-by-σ transformation in $\widehat{\mathbb{G}}$, with $T_\sigma(\perp) = \perp$.

Key generation (MKeyGen): Pick random generators $\widehat{g}_1, \widehat{g}_2$ from $\widehat{\mathbb{G}}$, and random $\vec{a} = (a_1, a_2), \vec{b} = (b_1, b_2)$ from $(\mathbb{Z}_q)^2$. The private key is $(\vec{a}, \vec{b})$. The public key consists of $\widehat{g}_1$, $\widehat{g}_2$, and the following values:

$$A = \textstyle\prod_{j=1}^2 \widehat{g}_j^{a_j}; \qquad B = \textstyle\prod_{j=1}^2 \widehat{g}_j^{b_j}$$

Encryption (MEnc): To encrypt $u \in \widehat{\mathbb{G}}$ under public key $(\widehat{g}_1, \widehat{g}_2, A, B)$, first pick random $v \in \mathbb{Z}_q$, $w \in \mathbb{Z}_q^*$. Output

$$(\widehat{g}_1^v, \widehat{g}_2^v, uA^v, B^v; \widehat{g}_1^w, \widehat{g}_2^w, A^w, B^w).$$

Decryption (MDec): To decrypt $U = (V_1, V_2, A_V, B_V; W_1, W_2, A_W, B_W)$ under private key $(\vec{a}, \vec{b})$: First, if $W_1 = W_2 = 1$, then output $\perp$. Then check the following constraints:

$$A_W \stackrel{?}{=} \textstyle\prod_{j=1}^2 W_j^{a_j}; \qquad B_V \stackrel{?}{=} \textstyle\prod_{j=1}^2 V_j^{b_j}; \qquad B_W \stackrel{?}{=} \textstyle\prod_{j=1}^2 W_j^{b_j};$$

If any fail, output $\perp$. Otherwise, output $u = A_V / \prod_{j=1}^2 V_j^{a_j}$.

Ciphertext transformation (MCTrans): To apply transformation T_σ to the ciphertext $U = (\vec{V}, A_V, B_V; \vec{W}, A_W, B_W)$, choose random $s \in \mathbb{Z}_q$, $t \in \mathbb{Z}_q^*$ and output

$$(V_1 W_1^s, V_2 W_2^s, \sigma A_V A_W^s, B_V B_W^s; W_1^t, W_2^t, A_W^t, B_W^t)$$

It is not hard to see that if U is in the support of $\mathsf{MEnc}_{\widehat{PK}}(u)$ (with random choices v and w), then the $\mathsf{MCTrans}(U, T_\sigma)$ is in the support of $\mathsf{MEnc}_{\widehat{PK}}(\sigma u)$, corresponding to random choices $v' = v + sw$ and $w' = tw$.

We emphasize that this DSME scheme does not achieve our desired definitions of a multiplicative homomorphic scheme, because given an encryption of u and a value $r \in \mathbb{Z}_q$, one can easily construct an encryption of u^r, and exponentiation by r is not an allowed transformation. Our main construction uses only the multiplicative transformation of DSME as a feature, although the security analysis accounts for the fact that other types of transformation are possible.

Main construction. We now present our main construction, which uses the previous DSME scheme as a component.

System parameters: A cyclic multiplicative group $\mathbb{G}$ of prime order p. A space of messages. We also require a secure DSME scheme over a group $\widehat{\mathbb{G}}$ of prime order q, where $\widehat{\mathbb{G}}$ is also a subgroup of $\mathbb{Z}_p^*$. This relationship is crucial, as the ciphertext transformation MCTrans of the DSME scheme must coincide with multiplication in the exponent in $\mathbb{G}$.

As described above, we let $\mathbb{G}^n$ be the message space, for any parameter n. The scheme is also parameterized by the subgroup $\mathbb{H}$ of allowed transformations. For $m \in \mathbb{G}^n$, let $f(m)$ be an injective encoding (say, an arbitrary representative) of the coset $m * \mathbb{H}$ to $\{0,1\}^*$. Let $\mathcal{H}$ be a family of collision-resistant hash functions from $\{0,1\}^*$ to $\mathbb{Z}_p$.[9] Finally, we require any fixed vector $\vec{z} \in (\mathbb{Z}_p)^4$, which is not a scalar multiple of the all-ones vector.

Key generation (KeyGen): Run $(\widehat{PK}, \widehat{SK}) \leftarrow \mathsf{MKeyGen}$ for the DSME scheme in $\widehat{\mathbb{G}}$. Pick random generators $g_1, \ldots, g_4$ from $\mathbb{G}$. For $i \in [n]$, choose random $\vec{c}_i = (c_{i,1}, \ldots, c_{i,4})$ from $(\mathbb{Z}_p)^4$ and compute $C_i = \prod_{j=1}^4 g_j^{c_{i,j}}$. Choose random $\vec{d} = (d_1, \ldots, d_4), \vec{e} = (e_1, \ldots, e_4)$ from $(\mathbb{Z}_p)^4$ and compute $D = \prod_{j=1}^4 g_j^{d_j}$ and $E = \prod_{j=1}^4 g_j^{e_j}$. Choose a random hash $\mathsf{H} \leftarrow \mathcal{H}$.

The private key is $(\widehat{SK}, \vec{c}_1, \ldots, \vec{c}_n, \vec{d}, \vec{e})$. The public key is $(\widehat{PK}, g_1, \ldots, g_4, C_1, \ldots, C_n, D, E, \mathsf{H})$.

Encryption (Enc): To encrypt $(m_1, \ldots, m_n) \in \mathbb{G}^n$ under a public key of the preceding form, first compute $\mu = \mathsf{H}(f(m_1, \ldots, m_n))$. Then pick random $x \in \mathbb{Z}_p$, $y \in \mathbb{Z}_p^*$ and random $u \in \widehat{\mathbb{G}}$, and output

$$\begin{array}{cccccccc} g_1^{(x+z_1)u}, & \ldots, & g_4^{(x+z_4)u}, & m_1 C_1^x, & \ldots, & m_n C_n^x, & (DE^\mu)^x; \\ g_1^{yu}, & \ldots, & g_4^{yu}, & C_1^y, & \ldots, & C_n^y, & (DE^\mu)^y; \\ & & & & & & \mathsf{MEnc}_{\widehat{PK}}(u) \end{array}$$

Decryption (Dec): Let ζ be a ciphertext of the preceding form, say, $\zeta = (\vec{X}, \vec{C}_X, P_X; \vec{Y}, \vec{C}_Y, P_Y; U)$, where

$$\vec{X} = (X_1, \ldots, X_4) \qquad \vec{C}_X = (C_{X,1}, \ldots, C_{X,n})$$
$$\vec{Y} = (Y_1, \ldots, Y_4) \qquad \vec{C}_Y = (C_{Y,1}, \ldots, C_{Y,n})$$

First compute $u = \mathsf{MDec}_{\widehat{SK}}(U)$. If $u = \perp$, output $\perp$. Otherwise, strip off u and $\vec{z}$ from the exponents as follows: For $j = 1, \ldots, 4$: set $\overline{X}_j = X_j^{1/u} g_j^{-z_j}$ and $\overline{Y}_j = Y_j^{1/u}$.

Compute the purported plaintext $(m_1, \ldots, m_n)$ via $m_i = C_{X,i} / \prod_{j=1}^4 \overline{X}_j^{c_{i,j}}$, and then compute $\mu = \mathsf{H}(f(m_1, \ldots, m_n))$. Finally, check the integrity of the ciphertext in the following way. If $Y_1 = \cdots = Y_4 = 1$ (the identity element in $\mathbb{G}$), output $\perp$. Check the following conditions:

$$P_X \stackrel{?}{=} \prod_{j=1}^4 \overline{X}_j^{d_j + \mu e_j}; \qquad C_{Y,i} \stackrel{?}{=} \prod_{j=1}^4 \overline{Y}_j^{c_{i,j}} \text{ (for each } i \in [n]);$$
$$P_Y \stackrel{?}{=} \prod_{j=1}^4 \overline{Y}_j^{d_j + \mu e_j}$$

If any checks fail, output $\perp$, otherwise output $(m_1, \ldots, m_n)$.

[9] Using the same technique as in the Cramer-Shoup scheme [30], our use of a hash can be removed, but at the expense of longer public keys.

Ciphertext transformation (CTrans): Let ζ be a ciphertext of the preceding form. To apply transformation $T_{(\tau_1,\ldots,\tau_n)}$ to ζ, choose random $\sigma \in \widehat{\mathbb{G}}$ and random $s \in \mathbb{Z}_p$, $t \in \mathbb{Z}_p^*$. Output:

$$\begin{array}{rrrrrrr} (X_1Y_1^s)^\sigma, & \ldots, & (X_4Y_4^s)^\sigma, & \tau_1 C_{X,1} C_{Y,1}^s, & \ldots, & \tau_n C_{X,n} C_{Y,n}^s, & P_X P_Y^s; \\ Y_1^{t\sigma}, & \ldots, & Y_4^{t\sigma}, & C_{Y,1}^t, & \ldots, & C_{Y,n}^t, & P_Y^t; \\ & & & & & & \mathsf{MCTrans}(U, T_\sigma) \end{array}$$

It is not hard to see that if ζ is in the support of $\mathsf{Enc}_{PK}(m_1, \ldots, m_n)$, say, with random choices x, y, and u, then the above ciphertext is in the support of $\mathsf{Enc}_{PK}(\tau_1 m_1, \ldots, \tau_n m_n)$, corresponding to random choices $x' = x + sy$, $y' = ty$, and $u' = \sigma u$.

High-level overview. Disregarding $\vec{z}$ and u, ciphertexts in our scheme resemble those in the original Cramer-Shoup scheme [30]. Two similar-looking "strands" are given, with only the first one directly carrying the message. This allows us to refresh the randomness x and y and achieve unlinkability when applying a transformation to the ciphertext. A similar "double-strand" paradigm was used by Golle, et al. [43], applied to the ElGamal encryption scheme to achieve a rerandomizable and anonymous CPA-secure scheme.

Without the additional random value u appearing in the exponents of some of the ciphertext components, the second strand's components would be completely independent of the first strand's. Thus, the scheme would be malleable via an attack which combined the first strand of one ciphertext and the second strand of another – the combination would result in a valid ciphertext if and only if the two ciphertexts shared the same μ value. The addition of u and its encryption under MEnc correlates the two strands, leaves u hidden from eavesdroppers, yet still allows for the random choice of u to be refreshed.

In our "double-strand" paradigm of achieving unlinkability, x's randomness is refreshed additively (as $x + sy$) and y's multiplicatively (as ty). However, without the $\vec{z}$ vector perturbing the randomness in x, there is a possible attack whereby x can be rerandomized *multiplicatively* and still result in a valid ciphertext (say, by squaring each component of the first strand). By adding $\vec{z}$, (intuitively) any attack that would multiply x would also multiply $\vec{z}$ as well. The decryption procedure only strips away one copy of $\vec{z}$, so x would remain perturbed for the Cramer-Shoup-like integrity checks on the ciphertext. In our analysis, it is important that $\vec{z}$ is linearly independent of the all-ones vector, so that an adversary would not be able to successfully compensate for additional perturbances in the Cramer-Shoup integrity checks.

We achieve our desired level of non-malleability by a technique similar to the Cramer-Shoup CCA-secure scheme [30]. It uses a ciphertext component of the form $(DE^\mu)^x$, where D, E are parts of the public key, x is a random value used in encryption, and μ is a hash of the ciphertext's prefix. The rerandomizable RCCA scheme of [83] uses the same paradigm, except that the value μ is a direct encoding of the plaintext (in rerandomizable RCCA, ciphertexts are malleable but only in ways which preserve the plaintext). In our HCCA-secure scheme, μ is a hash of an encoding of the coset $m * \mathbb{H}$. Intuitively, our scheme can therefore only be malleable in ways which preserve the $\mathbb{H}$-coset of the plaintext. A similar variation the Cramer-Shoup hashing was used in [71] to construct an encryption scheme which is non-malleable with respect to public "tags." In our construction, however, the tag/invariant is a function of the (private) plaintext.

Theorem 6.14. *The construction satisfies the correctness requirements, HCCA security, and unlinkable homomorphism properties with respect to $\mathcal{T}_{\mathbb{H}}$, for any subgroup $\mathbb{H}$ of $\mathbb{G}^n$, under the DDH assumption in the two cyclic groups.*

The lengthy proof follows in Appendix A.1. An overview is given below:

Proof Sketch. To show HCCA security, we must demonstrate appropriate RigEnc and RigExtract procedures. Our RigEnc encrypts a fixed dummy plaintext and uses a randomly chosen μ value instead of one derived from the plaintext. Our RigExtract similarly checks the integrity of the ciphertext using the same random μ value, and checks that the dummy plaintext was altered by an allowed transformation.

To show the suitability of these procedures in the HCCA experiment, we first describe an alternate encryption procedure which is implemented using the private key instead of the public key. When this procedure is used in place of Enc or RigEnc to generate the challenge ciphertext ζ^* in the HCCA security experiment, it follows from the DDH assumption that the difference is indistinguishable to any adversary. The ciphertexts produced by this alternate procedure are information-theoretically independent of the secret coin flip in the HCCA experiment, as well as some internal randomness used to generate the ciphertext.

Next, we show that given a fixed view of an adversary in the HCCA experiment, any ciphertext which is not in the support of $\mathsf{Enc}_{PK}(\cdot)$ or $\mathsf{CTrans}(\zeta^*, \cdot)$ is rejected by the decryption oracles Dec and RigDec (i.e, they output $\perp$ for such ciphertexts) with overwhelming probability over the remaining randomness in the experiment (which is independent

of the adversary's view). This is the most delicate part of our proof. It uses a linear-algebraic characterization of our scheme, and relies on the fact that certain quantities in the challenge ciphertext are distributed independently of the adversary's view. We also show an analogous statement for the GRigExtract oracles.

From the previous observation, we may replace the Dec, RigDec, and GRigExtract oracles (which use the secret key) with oracles that can be implemented using only information that is public to the adversary (e.g, the public key and challenge ciphertext). These oracles are computationally unbounded, as they exhaustively search the supports of $\mathsf{Enc}_{PK}(\cdot)$ and $\mathsf{CTrans}(\zeta^*, \cdot)$. Only with negligible probability do these alternate oracles give an answer which disagrees with the original oracles.

Finally, we conclude that with these two modifications — alternate encryption and decryption procedures — the adversary's entire view (the public key, challenge ciphertext and responses from the oracles) in the HCCA security experiment is independent of the secret bit b, and so the adversary's advantage is zero. Furthermore, this modified experiment is indistinguishable from the original experiment for any PPT adversary, so the HCCA security claim follows.

The correctness and unlinkable homomorphism properties are a direct consequence of the lemmas needed to prove the HCCA security. □

6.6 Opinion Polling Protocol Application

We now present an "opinion poll" protocol that elegantly illustrates the power of HCCA-secure encryption. The protocol is motivated by the following scenario:

A pollster wishes to collect information from many respondents. However, the respondents are concerned about the anonymity of their responses. Indeed, it is in the interest of the pollster to set things up so that the respondents are guaranteed anonymity, especially if the subject of the poll is sensitive personal information. To help collect responses anonymously, the pollster can enlist the help of an external tabulator. The respondents require that the external tabulator too does not see their responses, and that if the tabulator is honest, then responses are anonymized for the pollster (i.e., so that he cannot link responses to respondents). The pollster, on the other hand, does not want to trust the tabulator at all: if the tabulator tries to modify any responses, the pollster should be able to detect this so that the poll can be invalidated.

More formally, we give a secure protocol for the UC ideal functionality $\mathcal{F}_{\text{POLL}}$, described in Figure 6.9, where P_{client} is the pollster, P_{server} is the tabulator, and $P_1, \ldots, P_n$ are the respondents.

On input $(\text{SETUP}, P_{\mathsf{client}}, P_{\mathsf{server}}, P_1, \ldots, P_n)$ from party P_{client}:

- Give output $(\text{SETUP}, P_{\mathsf{client}}, P_{\mathsf{server}})$ to each party P_i.
- Give output $(\text{SETUP}, P_{\mathsf{client}}, P_1, \ldots, P_n)$ to P_{server}.

On input (INPUT, x_i) from input party P_i:

- Give output $(\text{INPUTFROM}, P_i)$ to P_{server}, and remember x_i.

On input OK from P_{server}:

- If P_{server} is corrupt, expect to receive from P_{server} a permutation σ on $\{1, \ldots, n\}$. If P_{server} is honest, choose σ at random.
- If not all $P_1, \ldots, P_n$ parties have supplied an input, or if some $x_i = \perp$, then give output $\perp$ to P_{client}.
- Otherwise, give output $(x_{\sigma(1)}, \ldots, x_{\sigma(n)})$ to P_{client}.

On input CANCEL from a corrupt P_{server}, give output $\perp$ to P_{client}.

Figure 6.9: UC ideal functionality $\mathcal{F}_{\text{POLL}}$.

Verifiable shuffling, mix-nets, and voting. Our opinion poll functionality can be viewed as an instantiation of verifiable shuffling (see e.g., [44, 47]). In a verifiable shuffle, a server takes in a collection of ciphertexts and outputs a random permutation in such a way that other parties are convinced that the shuffling server did not cheat; i.e., a shuffler cannot tamper with or omit any input ciphertext.

Verifying a shuffle typically involves using special-purpose zero-knowledge proofs, which are generally interactive and complicated. Even protocols whose verification is non-interactive rely on a common reference string setup [46].

Our approach is novel in that the shuffle's integrity can be verified without any zero-knowledge proof mechanism. Instead we leverage the strong limitations that the encryption scheme places on a malicious shuffler, resulting in a very efficient and simple protocol, which is secure even in the UC framework with no setups.

Verifiable shuffles are used in mix-nets [24] and in voting protocols. However, in our setting the shuffle is only verified to the pollster, and not to the respondents. In an election, the respondents also have an interest in the integrity of the shuffle (to know that their votes are included in the tally). We note that an election protocol (in which all participants receive guaranteed correct results) is not possible in the UC framework without trusted setups, given the impossibility results of Chapter 3.

The protocol. Our protocol is described in detail in Figure 6.10. The main idea is to use an HCCA-secure, transformation-hiding scheme, whose message space $\mathbb{G}^2$ (for a cyclic group $\mathbb{G}$), and whose only allowed operations are those of the form $(\alpha, \beta) \mapsto (\alpha, t\beta)$. In other words, anyone can apply the group operation to (multiply) the second plaintext component with a known value t, but the first component is completely non-malleable, and the two components remain "tied together." Both of our construction from Section 6.5 can easily accommodate these requirements.

To initiate the opinion poll, the pollster generates a (multiplicative) secret sharing $r_1, \ldots, r_n$ of a random secret group element R, then sends to the ith respondent a share r_i. Each respondent sends $\mathsf{Enc}(m_i, r_i)$ to the tabulator, where m_i is his response to the poll. Now the tabulator can blindly re-randomize the shares (multiply the ith share by a random s_i, such that $\prod_i s_i = 1$), shuffle the resulting ciphertexts, and send them to the pollster. The pollster will ensure that the shares encode the secret R and accept the results.

The security of the protocol is informally argued as follows. An honest pollster only sees a random permutation of the responses, and a completely random sharing of R. There is no way to link any responses to the r_i shares he originally dealt to the respondents, either by looking at the new shares of R, or via the encryption scheme itself. The tabulator sees only encrypted data, and in particular has no information about the shares r_i. The only way the tabulator could successfully generate ciphertexts whose second components are shares of R is by making exactly one of his ciphertexts be derived from each respondent's ciphertext. By the non-malleability of the encryption scheme, each response m_i is inextricably "tied to" the corresponding share r_i and cannot be modified, so each respondent's response must be represented exactly once in the tabulator's output without tampering. Finally, observe that the responses of malicious respondents must be independent of honest parties' responses – by "copying" an honest respondent's ciphertext to the tabulator, a malicious respondent also "copies" the corresponding r_i, which would cause the set of shares to be inconsistent with overwhelming probability.

Let $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{CTrans})$ be a homomorphic encryption scheme with message space $\mathbb{G}^2$. We suppose the CTrans operation accepts arguments as $\mathsf{CTrans}(C, t)$, where $t \in \mathbb{G}$ indicates the transformation $(\alpha, \beta) \mapsto (\alpha, t\beta)$. The protocol proceeds as follows:

1. P_{client} generates a key pair $(SK, PK) \leftarrow \mathsf{KeyGen}$ and chooses random elements $r_1, \ldots, r_n \leftarrow \mathbb{G}$, remembering $R = \prod_i r_i$. She then broadcasts PK, sends $(r_i, P_{\mathsf{server}})$ to each party P_i, and sends $(P_{\mathsf{client}}, P_1, \ldots, P_n)$ to P_{server}.
2. Each input party P_i holds input x_i (encoded into an element of $\mathbb{G}$). He receives PK and $(r_i, P_{\mathsf{server}})$ from P_{client}, then sends $\mathsf{Enc}_{PK}(x_i, r_i)$ to P_{server} through a secure channel.
3. P_{server} collects ciphertext C_i from each input party P_i, then chooses a random permutation σ on $[n]$ and random $s_1, \ldots, s_n \leftarrow \mathbb{G}$ subject to $\prod_i s_i = 1$. He computes $C'_i = \mathsf{CTrans}(C_{\sigma(i)}, s_{\sigma(i)})$ for each i and sends $(C'_1, \ldots, C'_n)$ to P_{client}.
4. P_{client} receives $(C'_1, \ldots, C'_n)$ from P_{server} and decrypts each C'_i as $(x'_i, r'_i) \leftarrow \mathsf{Dec}_{SK}(C'_i)$. If any decryptions fail, or if $\prod_i r'_i \neq R$, she aborts. Otherwise, she outputs $(x'_1, \ldots, x'_n)$.

Figure 6.10: Opinion poll protocol

Theorem 6.15. *If $\mathcal{E}$ is HCCA-secure and transformation-hiding with parameters as described above, and $|\mathbb{G}|$ is superpolynomial in the security parameter, then our protocol is a secure realization of $\mathcal{F}_{\text{POLL}}$ against static adversaries.*

Proof. Given a real-world adversary $\mathcal{A}$, we construct an ideal-world simulator $\mathcal{S}$. We break the proof down into 4 cases according to which parties $\mathcal{A}$ corrupts:

Case 1: If $\mathcal{A}$ corrupts neither P_{server} nor P_{client}, then suppose by symmetry that $\mathcal{A}$ corrupts some input parties $P_1, \ldots, P_k$. Then the main task for $\mathcal{S}$ is to extract the inputs of each corrupt P_i and send them to $\mathcal{F}_{\text{POLL}}$. $\mathcal{S}$ simply does the following:

- On receiving $(\text{SETUP}, P_{\mathsf{client}}, P_{\mathsf{server}}, P_1, \ldots, P_n)$ from $\mathcal{F}_{\text{POLL}}$, generate $(PK, SK) \leftarrow \mathsf{KeyGen}$. Choose random $r_1, \ldots, r_k \leftarrow \mathbb{G}$ and simulate that P_{client} broadcast PK and sent $(r_i, P_{\mathsf{server}})$ to each corrupt input party P_i.
- If not all corrupt parties P_i send a ciphertext C_i to P_{server}, then abort. Otherwise, set $(x_i, r'_i) \leftarrow \mathsf{Dec}_{SK}(C_i)$.
- If any of the above decryption fails, or if $\prod_i r'_i \neq \prod_i r_i$, then send $(\text{INPUT}, \perp)$ to $\mathcal{F}_{\text{POLL}}$ on behalf of *each* corrupt input party P_i.
- Otherwise send (INPUT, x_i) to $\mathcal{F}_{\text{POLL}}$ on behalf of each corrupt input party P_i.

It is straight-forward to see that in the cases where $\mathcal{S}$ sends $(\text{INPUT}, \perp)$, then by the honest behavior of P_{server} and P_{client}, the protocol would have mandated that P_{client} refuse the output.

Case 2: If $\mathcal{A}$ corrupts P_{client} and (without loss of generality) input parties $P_1, \ldots, P_k$, then $\mathcal{S}$ does the following:

- When corrupt P_{client} broadcasts PK and sends $(r_i, P_{\mathsf{server}})$ to each honest input party P_i, send $(\text{SETUP}, P_{\mathsf{client}}, P_{\mathsf{server}}, P_1, \ldots, P_n)$ to $\mathcal{F}_{\text{POLL}}$ on behalf of P_{client}.
- When a corrupt input party P_i sends a ciphertext C_i to honest P_{server}, send (INPUT, x_0) to $\mathcal{F}_{\text{POLL}}$ on behalf of P_i, where x_0 is any arbitrary fixed message.
- When $\mathcal{F}_{\text{POLL}}$ gives the final output to $\mathcal{S}$, remove as many x_0's from the output list as there are corrupt input parties. Arbitrarily order the remaining outputs as $x_{k+1}, \ldots, x_n$. For each $i \in [n]$, choose a random s_i such that $\prod_i s_i = 1$. Simulate that P_{server} sends a random permutation of $\{\mathsf{Enc}_{PK}(x_i, r_i s_i) \mid i > k\} \cup \{\mathsf{CTrans}(C_i, s_i) \mid i \leq k\}$ to P_{client}.

Since P_{client} is corrupt, $\mathcal{S}$ can legally obtain the set of honest input parties' inputs. The only difference therefore between the view of $\mathcal{A}$ in the real world and our simulation is that in the real world, P_{client} sees $\mathsf{CTrans}(\mathsf{Enc}_{PK}(x_i, r_i), s_i)$ for each honest party P_i, while in the simulation, P_{client} sees $\mathsf{Enc}_{PK}(x_i, r_i s_i)$. By the transformation-hiding property of the scheme, this difference is indistinguishable. Also in the simulation, each x_i is paired with a potentially different r_i than might be the case in the real world protocol (since the simulator receives a shuffled list of x_i values). However, the distribution of $(x_i, r_i s_i)$ pairs is independent of the initial assignments of x_i's to r_i's.

Case 3: If $\mathcal{A}$ corrupts P_{server} and input parties $P_1, \ldots, P_k$, then $\mathcal{S}$ does the following:

- When $\mathcal{F}_{\text{POLL}}$ gives $(\text{SETUP}, P_{\mathsf{client}}, P_1, \ldots, P_n)$ to $\mathcal{S}$, generate $(PK, SK) \leftarrow \mathsf{KeyGen}$. Pick random $r_1, \ldots, r_n \leftarrow \mathbb{G}$ and simulate that P_{client} broadcast PK and sent $(r_i, P_{\mathsf{server}})$ to each corrupt P_i.
- When $\mathcal{F}_{\text{POLL}}$ gives $(\text{INPUTFROM}, P_i)$ to $\mathcal{S}$ for an honest party $(i > k)$, generate $(C_i, S_i) \leftarrow \mathsf{RigEnc}_{PK}$ and simulate that P_i sent C_i to P_{server}. Remember S_i.
- When P_{server} sends P_{client} a list of ciphertexts $(C'_1, \ldots, C'_n)$, do the following for each i:
  - If $\mathsf{Dec}_{SK}(C'_i) \neq \perp$, then set $(x_i, r'_i) \leftarrow \mathsf{Dec}_{SK}(C'_i)$.
  - Else, if $\mathsf{RigExtract}_{SK}(C'_i, S_j) \neq \perp$ for some j, set $r'_i := r_i \cdot \mathsf{RigExtract}_{SK}(C'_i, S_j)$.
  - If both these operations fail, send CANCEL to $\mathcal{F}_{\text{POLL}}$ on behalf of P_{server}.

  If $\prod_i r'_i \neq \prod_i r_i$ or for some $j > k$, there is more than one i such that $\mathsf{RigExtract}_{SK}(C'_i, S_j) \neq \perp$, then send CANCEL to $\mathcal{F}_{\text{POLL}}$ on behalf of P_{server}.

  Otherwise, let σ be any permutation on $[n]$ that maps each $j > k$ to the unique i such that $\mathsf{RigExtract}_{SK}(C'_i, S_j) \neq \perp$. Send $(\text{INPUT}, x_{\sigma(i)})$ to $\mathcal{F}_{\text{POLL}}$ on behalf of corrupt P_i $(i \leq k)$, and then send OK to $\mathcal{F}_{\text{POLL}}$ on behalf of P_{server}, with σ as the permutation that $\mathcal{F}_{\text{POLL}}$ expects.

In this case, the primary task of $\mathcal{S}$ is to determine whether the corrupt P_{server} gives a valid list of ciphertexts to P_{client}. Applying the HCCA definition in a sequence of hybrid interactions, we see that the behavior of the real world interaction versus this simulation interaction is preserved when appropriately replacing Enc/Dec with RigEnc/RigExtract.

Note that the adversary's view is independent of $r_{k+1}, \ldots, r_n$. If $\mathsf{Dec}_{SK}(C'_i) \neq \perp$, then the corresponding r'_i value computed by the simulator is also independent of $r_{k+1}, \ldots, r_n$. Thus the only way $\prod_i r_i = \prod_i r'_i$ can be satisfied with non-negligible probability is if for each honest party P_j, exactly one i satisfies $\mathsf{RigExtract}_{SK}(C'_i, S_j) \neq \perp$. In this

case, there will be exactly as many x_i's as corrupt players, and the simulator can legitimately send these to $\mathcal{F}_{\text{POLL}}$ as instructed (with the appropriate permutation).

Case 4. If $\mathcal{A}$ corrupts P_{server}, P_{client}, and input parties $P_1, \ldots, P_k$, then $\mathcal{S}$ can legitimately obtain each honest input party's input, so simulation is relatively straight-forward. More formally, $\mathcal{S}$ does the following:

- Send $(\text{SETUP}, P_{\text{client}}, P_{\text{server}}, P_1, \ldots, P_n)$ to $\mathcal{F}_{\text{POLL}}$ on behalf of P_{client}.
- Send (INPUT, x_0) to $\mathcal{F}_{\text{POLL}}$ on behalf of each corrupt input party P_i, where x_0 is an arbitrary fixed message.
- After receiving $(\text{INPUTFROM}, P_i)$ for all honest input parties P_i, send OK to $\mathcal{F}_{\text{POLL}}$ on behalf of P_{server}, and give the identity permutation as σ.
- After receiving $(x_1, \ldots, x_n)$ as output, we know that party P_i was invoked with input x_i, so we can perfectly simulate the honest parties to $\mathcal{A}$. □

Boolean OR on encrypted data. Using a similar technique, we can obtain a UC-secure protocol for a boolean-OR functionality. This functionality is identical to $\mathcal{F}_{\text{POLL}}$ except that P_{server} also gets to provide an input (i.e., we identify P_{server} with P_0), and instead of giving $(x_{\sigma(0)}, \ldots, x_{\sigma(n)})$, it gives $\bigvee_i x_i$ as the output to P_{client}.

We can achieve this new functionality with a similar protocol — this time, using an encryption scheme that is unlinkable HCCA-secure with respect to all group operations in $\mathbb{G}^2$. P_{client} sends shares r_i to the input parties as before. The input parties send $\text{Enc}_{PK}(x_i, r_i)$ to P_{server}, where $x_i = 1$ if P_i's input is 0, and x_i is randomly chosen in $\mathbb{G}$ otherwise. Then, P_{server} rerandomizes the r_i shares as before, and also randomizes the x_i's in the following way: P_{server} multiplies each x_i by s_i such that $\prod_i s_i = 1$ if P_{server}'s input is 0, and $\prod_i s_i$ is random otherwise (P_{server} can randomize both sets of shares simultaneously using the homomorphic operation). P_{client} receives the processed ciphertexts and ensures that $\prod_i r'_i = 1$. Then if $\prod_i x'_i = 1$, it outputs 0, else it outputs 1.

We note that this approach to evaluating a boolean OR (where the induced distribution is a fixed element if the result is 0, and is random if the result is 1) has previously appeared elsewhere, e.g., [12, 13].

6.7 Beyond Unary Transformations

Many interesting applications of homomorphic encryptions involve (at least) *binary* operations — those which accept encryptions of plaintexts m_0 and m_1 and output a ciphertext encoding $T(m_0, m_1)$. A common example is ElGamal encryption, where T is the group operation of the underlying cyclic group. In this section, we examine the possibility of extending our results to schemes with *binary* transformations.

Before presenting our results, we must first define appropriate extensions of our definitions to the case of binary homomorphic operations. Developing appropriate (and succinct) indistinguishability-style definitions appears to be a difficult task. Thus, the results in this section use security formulations as ideal functionalities in the UC model, as in Section 6.3.3.

6.7.1 Negative Result for Binary Group Operations

For an impossibility result, we make the security requirements on the ideal functionality as weak as possible. Throughout this subsection, we consider an ideal functionality $\mathcal{F}$ similar to $\mathcal{F}_{\text{HMP}}^{\mathcal{T}}$, with the following properties:

- Any party may post a new handle by either providing a plaintext message, or by providing *two* existing handles and a (binary) transformation $T \in \mathcal{T}$. In the latter case, the message for the new handle is calculated as $T(m_1, m_2)$, where m_1, m_2 are the messages corresponding to the given handles.
- Only the "owner" of the functionality can obtain the message corresponding to each handle. All other parties simply receive notification that the handle was generated.
- Handles are generated by the adversary, without knowledge of the corresponding plaintext message, or which of the two ways the handle was produced.

For simplicity, we have not considered the functionality's behavior on handles originally posted by the adversary (so-called "dummy" handles in the case of $\mathcal{F}_{\text{HMP}}^{\mathcal{T}}$). However, our impossibility results do not depend on these details, and one may consider the weakest possible ideal functionality, which reveals everything to the adversary when honest parties try to use dummy handles.

We can now formalize our impossibility result:

Theorem 6.16. *There is no secure realization of $\mathcal{F}$ via a homomorphic encryption scheme, when $\mathcal{T}$ contains a group operation on the message space, and the size of the message space is superpolynomial.*

The main observation is that each handle (ciphertext) must have a bounded length independent of its "history" (i.e., whether it was generated via the homomorphic reposting operation and if so, which operations applied to which existing handles), and thus can only encode a bounded amount of information about its history. We show that any simulator for $\mathcal{F}$ must be able to extract a reasonable history from any handle output by the adversary.

However, when a group operation is an allowed transformation, there can be far more possible histories than can be encoded in a single handle. We use this fact to construct an environment and adversary which can distinguish between the real world and the ideal world with any simulator, contradicting the security definition.

Proof. We will construct an environment that will distinguish between the ideal interaction with $\mathcal{F}$ and the real-world protocol interaction involving any homomorphic encryption scheme. Let $\otimes$ be the group operation over message space $\mathcal{M}$.

The environment invokes an interaction with two honest parties Alice and Bob, and a dummy adversary Carol. The environment instructs Bob to SETUP an instance of the, then chooses d random messages $m_1, \ldots, m_d \leftarrow \mathcal{M}$, where d is a parameter to be fixed later, and instructs Alice to POST each of them. Then, the environment chooses a random $S \subseteq \{1, \ldots, d\}$ and then, given the handles for the posted messages, internally runs the encryption scheme algorithm to obtain a ciphertext h^* encoding $\bigotimes_{i \in S} m_i$. The environment can do this locally because this protocol implements the REPOST operation via a non-interactive procedure CTrans. Finally, the environment instructs the adversary to broadcast the resulting handle/ciphertext h^*, then instructs Bob to open it. The environment outputs 1 if Bob outputs $\bigotimes_{i \in S} m_i$.

Clearly in the real-world interaction, the environment outputs 1 with overwhelming probability, by the correctness of the encryption scheme's homomorphic operation. We will show that no simulator can achieve the same effect in the ideal interaction.

Any simulator for this adversary must post the handle h^* to $\mathcal{F}$ according to one of the "legal" reposting features. In order to induce the correct output, the simulator must specify a legal transformation T' such that $T'(m_1, \ldots, m_d) = \bigotimes_{i \in S} m_i$ with overwhelming probability. From the definition of $\mathcal{F}$, the simulator's view is independent of the choice of $m_1, \ldots, m_d$ (the handles are generated without knowledge of the plaintext messages). Thus the simulator must in fact specify a legal function T' such that $T'(m_1, \ldots, m_d) = \bigotimes_{i \in S} m_i$, with overwhelming probability over the choice of $m_1, \ldots, m_d$.

Let $\ell(k)$ be a polynomial bound on the length of handles in the given encryption scheme (when the security parameter is k); say, the running time of simulator when answering HANDLE-REQ requests. Let us choose $d = \ell(k) + 1$. Then our environment remains polynomial-time in k. However, then the simulator's view (namely, the ciphertext/handle h^*) contains at least 1 bit of uncertainty about the environment's choice of S.

However, we observe that if $S \neq S' \subseteq \{1, \ldots, d\}$, then the probability (taken over choice of $m_1, \ldots, m_d$) that $\bigotimes_{i \in S} m_i = \bigotimes_{i \in S'} m_i$ is negligible. Suppose a transformation T' agrees with some $\bigotimes_{i \in S'}$ on an overwhelming fraction of inputs. Then the set S' is unique, since all other products in the group disagree with it in most inputs.

Therefore, any simulator must output a transformation T' that with overwhelming probability *uniquely* identifies the random subset S chosen by the environment. However, the simulator's view has one bit of uncertainty about the environment's choice of S, so this required task is impossible. □

6.7.2 Positive Result for a Relaxation of Unlinkability

The impossibility result of the previous section leaves open the possibility of achieving a relaxation of the unlinkability requirement. We consider a relaxation similar to Sander, Young, and Yung [92]; namely, we allow the ciphertext to leak the number of operations applied to it (i.e., the depth of the circuit applied), but no additional information. To make this this requirement more formal, we associate a *length* parameter with each ciphertext. If a length-ℓ and a length-ℓ' ciphertext are combined, then the result is a length $\ell + \ell'$ ciphertext. Our security definition insists that ciphertexts reveal (at most) their associated length parameter.

The main idea in our construction is to encode a group element m into a length-ℓ ciphertext as a vector $\big(\mathsf{Enc}(m_1), \ldots, \mathsf{Enc}(m_\ell)\big)$, where the m_i's are a random multiplicative sharing of m in the group. and Enc is HCCA-secure, with respect to the group operation. To "multiply" two such encrypted encodings, we can simply concatenate the two vectors of ciphertexts together, and rerandomize the new set of shares (multiply the ith component by s_i, where $\prod_i s_i = 1$) to bind the sets together.

The above outline captures the main intuition, but our actual construction uses a slightly different approach to ensure UC security. In the scheme described above, anyone can split the vector $\big(\mathsf{Enc}(m_1), \ldots, \mathsf{Enc}(m_\ell)\big)$ into two smaller vectors that encode two (random) elements whose product is m. We interpret this as a violation of our

desired properties, since it is a way to derive two encodings whose values are related to a *longer* encoding. To avoid the problem of "breaking apart" ciphertexts, we instead encode m as $\big(\mathsf{Enc}(\alpha_1, \beta_1), \ldots, \mathsf{Enc}(\alpha_\ell, \beta_\ell)\big)$, where the α_i's and β_i's form two *independently random* multiplicative sharings of m. Rerandomizing these encodings is possible when we use a scheme that is homomorphic with respect to the group operation in $\mathbb{G}^2$ (i.e., by setting the parameter $\mathbb{H} = \mathbb{G}^2$ in our construction). Intuitively, these encodings cannot be split up in such a way that the first components and second components are shares of the same value. Note that it is crucial that the (α_i, β_i) pairs cannot themselves be "broken apart."

Security definition. The functionality, called $\mathcal{F}_{\mathbb{G}}$, is given in full detail in Figure 6.11. Below we explain and motivate the details of the definition.

The functionality keeps track of a database of records of the form $(\mathsf{handle}, \ell, m)$. Let $\mathsf{GetHandle}(args)$ be a subroutine which sends $(\text{HANDLE-REQ}, args)$ to the adversary and expects in return a string handle. If handle is previously recorded in the database, abort; otherwise, return handle.

Setup: On receiving a command SETUP from a party P: If a previous SETUP command has been processed, abort. Else, send $(\text{ID-REQ}, P)$ to the adversary, and expect in response a string id. Broadcast $(\text{ID-ANNOUNCE}, P, \mathsf{id})$ to all other parties.

Dummy handles: On receiving a command $(\text{DUMMY}, \ell, \mathsf{handle})$ from a *corrupt party only*, internally record $(\mathsf{handle}, \ell, \perp)$ and broadcast $(\text{HANDLE-ANNOUNCE}, \mathsf{handle})$ to all parties.

Posting messages: On receiving a command $(\text{POST}, \ell, m_0, \mathsf{handle}_1, \ldots, \mathsf{handle}_k)$ from a party sender: If any handle_i is not recorded internally, or $m_0 \notin \mathbb{G}$, ignore the request. Otherwise, suppose $(\mathsf{handle}_i, \ell_i, \mathsf{msg}_i)$ is recorded for each i. If $\ell < \sum_i \ell_i$, ignore the request. Let $D = \{i \mid m_i = \perp\} \subseteq [k]$, the indices of the dummy handles. Set $m^* = m_0 * \prod_{i \notin D} m_i$, the product of known plaintexts involved.

- If $D = \emptyset$ (no dummy handles involved): If P is corrupt, set $\mathsf{handle}^* \leftarrow \mathsf{GetHandle}(\mathsf{sender}, \ell, m^*)$; otherwise let $\mathsf{handle}^* \leftarrow \mathsf{GetHandle}(\mathsf{sender}, \ell)$. Internally record $(\mathsf{handle}^*, \ell, m^*)$ and broadcast $(\text{HANDLE-ANNOUNCE}, \mathsf{handle}^*)$ to all parties.
- If $\ell > \sum_{i \in D} \ell_i$ (not entirely derived from dummy handles): If P is corrupt, set $\mathsf{handle}' \leftarrow \mathsf{GetHandle}(\mathsf{sender}, \ell', m^*)$, else set $\mathsf{handle}' \leftarrow \mathsf{GetHandle}(\mathsf{sender}, \ell')$. Internally record $(\mathsf{handle}', \ell', m^*)$.

  Set $\mathsf{handle}^* \leftarrow \mathsf{GetHandle}(\mathsf{sender}, \ell, \{\mathsf{handle}'\} \cup \{\mathsf{handle}_i \mid i \in D\})$. Internally record $(\mathsf{handle}^*, \ell, \perp)$ and send $(\text{HANDLE-ANNOUNCE}, \mathsf{handle}^*)$ to all parties.
- Otherwise (dummy handles only), Set $\mathsf{handle}^* \leftarrow \mathsf{GetHandle}(\mathsf{sender}, \ell, m_0, \{\mathsf{handle}_i \mid i \in D\})$. Internally record $(\mathsf{handle}^*, \ell, \perp)$ and send $[\text{HANDLE-ANNOUNCE}, \mathsf{handle}^*]$ to all parties.

Message reading: On receiving a command $(\text{GET}, \mathsf{handle})$ from party P (who gave the first SETUP command): If $(\mathsf{handle}, \ell, \mathsf{msg})$ is recorded internally, send msg to P; else send $\perp$.

Figure 6.11: UC ideal functionality $\mathcal{F}_{\mathbb{G}}$, parameterized by a cyclic group $\mathbb{G}$.

Following our desired intuition, users can only generate new messages in two ways (for uniformity, all handled in the same part of the functionality's code). A user can simply post a message by supplying a group element m (this is the case where $k = 0$ in the user's POST command). Alternatively, a user can provide a list of existing handles along with a group element m. If all these handles correspond to honestly-generated posts, then this has the same effect as if the user posted the product of all the corresponding messages (though note that the user does not have to know what these messages are to do this). We model the fact that handles reveal nothing about the message by letting the adversary choose the actual handle string, without knowledge of the message. The designated recipient can obtain the message by providing a handle to the functionality. Note that there is no way (even for corrupt parties) to generate a handle derived from existing handles in a non-approved way.

As in $\mathcal{F}_{\text{HMP}}^{\mathcal{T}}$, adversaries can also post *dummy handles*, which contain no message. When a user posts a derived message using such a handle, the resulting handle also contains no message. When the handle is used in a derived POST command, the adversary is informed. The adversary also gets access to an "intermediate" handle corresponding to all the non-DUMMY handles that were combined in the POST request. Still, the adversary learns nothing about the messages corresponding to these handles.

The construction. Let $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{CTrans})$ be an unlinkable HCCA-secure scheme, whose message space is $\mathbb{G}^2$ for a cyclic group $\mathbb{G}$, and whose allowed (unary) transformations are all group operations in $\mathbb{G}^2$. We suppose the CTrans operation accepts arguments as $\mathsf{CTrans}(C, (r, s))$, where $r, s \in \mathbb{G}$ specify the transformation $(\alpha, \beta) \mapsto (r\alpha, s\beta)$. We abbreviate the $\mathsf{CTrans}(C, (r, s))$ operation as "$(r, s) * C$". Thus $(r, s) * \mathsf{Enc}_{PK}(\alpha, \beta)$ is indistinguishable from $\mathsf{Enc}_{PK}(r\alpha, s\beta)$, in the sense of the unlinkability definition.

The new scheme $\mathcal{E}^*$ is given by the following algorithms:

Key generation (KeyGen^*): Same as KeyGen.

Encryption (Enc^*): To encrypt an element $m \in \mathbb{G}$ in a length-ℓ ciphertext, output

$$C = \Big(\mathsf{Enc}_{PK}(\alpha_1, \beta_1), \ldots, \mathsf{Enc}_{PK}(\alpha_\ell, \beta_\ell)\Big)$$

where α_i, β_i are randomly chosen in $\mathbb{G}$ subject to the constraint $\prod_i \alpha_i = \prod_i \beta_i = m$.

Decryption (Dec^*): To decrypt a ciphertext $C = (C_1, \ldots, C_\ell)$, decrypt each C_i to get (α_i, β_i). If any decryption returns $\perp$, or if $\prod_i \alpha_i \neq \prod_i \beta_i$, output $\perp$. Else output $\prod_i \alpha_i$.

Transformation operation (CTrans^*): To "multiply" two given ciphertexts $C = (C_1, \ldots, C_\ell)$ and $C' = (C'_1, \ldots, C'_{\ell'})$, output a *random permutation* of:

$$\Big((r_1, s_1) * C_1, \ldots, (r_\ell, s_\ell) * C_\ell, (r_{\ell+1}, s_{\ell+1}) * C'_1, \ldots, (r_{\ell+\ell'}, s_{\ell+\ell'}) * C'_{\ell'}\Big)$$

where r_i, s_i are randomly chosen in $\mathbb{G}$ subject to $\prod_i r_i = \prod_i s_i = 1$

To "multiply" a single given ciphertext $C = (C_1, \ldots, C_\ell)$ by a given known group element $R \in \mathbb{G}$ (without increasing the ciphertext length), output a random permutation of:

$$\Big((r_1, s_1) * C_1, \ldots, (r_\ell, s_\ell) * C_\ell\Big)$$

where r_i, s_i are randomly chosen in $\mathbb{G}$ subject to $\prod_i r_i = \prod_i s_i = R$.

We note that the syntax of CTrans^* can be naturally extended to support multiplying several ciphertexts and/or a known group element at once, simply by composing the operations described above.

Theorem 6.17. *If $\mathcal{E}$ is unlinkable and HCCA-secure with respect to $\mathbb{G}^2$, where $|\mathbb{G}|$ is superpolynomial in the security parameter, then $\mathcal{E}^*$ (as described above) is a secure realization of $\mathcal{F}_\mathbb{G}$, with respect to static corruptions.*

Proof. Let $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{CTrans})$ be the unlinkable HCCA-secure scheme used as the main component in our construction, and let RigEnc and $\mathsf{RigExtract}$ be the procedures guaranteed by HCCA security.

We proceed by constructing an ideal-world simulator for any arbitrary real-world adversary $\mathcal{A}$. The simulator $\mathcal{S}$ is constructed by considering a sequence of hybrid functionalities that culminate in $\mathcal{F}_\mathbb{G}$. These hybrids differ from $\mathcal{F}_\mathbb{G}$ only in how much they reveal in their HANDLE-REQ requests to the adversary.

Correctness. Note that $\mathcal{F}_\mathbb{G}$ only makes two kinds of HANDLE-REQ requests: those containing a lone message, and those containing a list of handles.

Let $\mathcal{F}_1$ be the functionality that behaves exactly as $\mathcal{F}_\mathbb{G}$, except that every time it sends a HANDLE-REQ to the simulator, it also includes the entire party's input that triggered the HANDLE-REQ. Define $\mathcal{S}_1$ to be the simulator that internally runs the adversary $\mathcal{A}$, and does the following:

- When $\mathcal{F}_1$ gives $(\text{ID-REQ}, P)$ to $\mathcal{S}_1$, it generates a key pair $(PK, SK) \leftarrow \mathsf{KeyGen}$ and responds with PK. It simulates to $\mathcal{A}$ that party P broadcast PK.
- When $\mathcal{F}_1$ gives a HANDLE-REQ to $\mathcal{S}_1$, it generates the handle appropriately — with either Enc^*_{PK} or CTrans^* on an existing handle, depending on the party's original command which is included in the HANDLE-REQ. It simulates to $\mathcal{A}$ that the appropriate party output the handle.
- When $\mathcal{A}$ broadcasts a length-ℓ ciphertext C, $\mathcal{S}_1$ tries to decrypt it with Dec^*_{SK}. If it decrypts (say, to m), then $\mathcal{S}_1$ sends a (POST, ℓ, m) command to $\mathcal{F}_1$ and later gives C as the handle; else it sends (DUMMY, ℓ, C).

$\mathcal{S}_1$ exactly simulates the honest parties' behavior in the real world interaction. By the correctness properties of $\mathcal{E}^*$, the outputs of the honest ideal-world parties match that of the real world, except with negligible probability; thus, $\text{EXEC}[\mathcal{Z}, \mathcal{A}, \mathcal{E}^*, \mathcal{F}_{\text{BCAST}}] \approx \text{EXEC}[\mathcal{Z}, \mathcal{S}_1, \pi_{\mathsf{dummy}}, \mathcal{F}_1]$ for all environments $\mathcal{Z}$.

Unlinkability. Let $\mathcal{F}_2$ be exactly like $\mathcal{F}_1$, except for the following change: For requests of the form $(\text{HANDLE-REQ}, \mathsf{sender}, \ell, m)$, $\mathcal{F}_2$ does not send the handles that caused this request. That is, whereas $\mathcal{F}_1$ would tell the simulator that the handle is being requested for a POST command combining some non-dummy handles, $\mathcal{F}_2$ would instead act like sender had sent (POST, ℓ, m) (that this is closer to what $\mathcal{F}_{\mathbb{G}}$ does; internally behaving identically for such requests). Let $\mathcal{S}_2 = \mathcal{S}_1$, since $\mathcal{F}_1$ is only sending one fewer type of HANDLE-REQ to the simulator.

By a standard hybrid argument, we can see that $\text{EXEC}[\mathcal{Z}, \mathcal{S}_1, \pi_{\mathsf{dummy}}, \mathcal{F}_1] \approx \text{EXEC}[\mathcal{Z}, \mathcal{S}_2, \pi_{\mathsf{dummy}}, \mathcal{F}_2]$ for all environments $\mathcal{Z}$. The hybrids are over the number of POST requests affected by this change. Consecutive hybrids differ by whether a single handle was generated by Enc^* or by CTrans^*. The only handles that are affected here are non-DUMMY handles, and thus ciphertexts which decrypt successfully under SK. Thus distinguishing between consecutive hybrids can be reduced to succeeding in the unlinkability experiment (by further hybridizing over the individual Enc ciphertext components).

HCCA. If the owner P of the functionality is corrupt, then $\mathcal{S}_2$ is already a suitable simulator for $\mathcal{F}_{\mathbb{G}}$, and we can stop at this point.

Otherwise, the difference between $\mathcal{F}_{\mathbb{G}}$ and $\mathcal{F}_2$ is that $\mathcal{F}_{\mathbb{G}}$ does not reveal the message in certain HANDLE-REQ requests. Namely, those in which the simulator receives $(\text{HANDLE-REQ}, \mathsf{sender}, \ell)$.

Let $\mathcal{S}_3$ be exactly like $\mathcal{S}_2$, except for the following changes: Each time $\mathcal{S}_2$ would generate a ciphertext component via $\mathsf{Enc}_{PK}(\alpha, \beta)$, $\mathcal{S}_3$ instead generates it with RigEnc_{PK}. It keeps track of the auxiliary information S and records (S, α, β) internally. Also, whenever $\mathcal{S}_2$ would decrypt a ciphertext component using Dec_{SK}, $\mathcal{S}_3$ instead decrypts it via:

$$D(C) = \begin{cases} (r\alpha, s\beta) & \text{if } (S, \alpha, \beta) \text{ is recorded s.t. } (r, s) \leftarrow \mathsf{RigExtract}_{SK}(C, S) \\ \mathsf{Dec}_{SK}(C) & \text{otherwise} \end{cases}$$

By a straight-forward hybrid argument (in which distinguishing between adjacent hybrids reduces to success in a single instance of the HCCA experiment), we have that $\text{EXEC}[\mathcal{Z}, \mathcal{S}_2, \pi_{\mathsf{dummy}}, \mathcal{F}_2] \approx \text{EXEC}[\mathcal{Z}, \mathcal{S}_3, \pi_{\mathsf{dummy}}, \mathcal{F}_2]$ for all environments $\mathcal{Z}$.

We now examine when a ciphertext given by the adversary is successfully decrypted by the simulator (and thus given to the functionality as a POST instead of as a DUMMY handle).

Given a ciphertext (sequence of HCCA ciphertexts) $C = (C_1, \ldots, C_\ell)$, $\mathcal{S}_3$ first decrypts each C_i to obtain $(\alpha'_i, \beta'_i) = D(C_i)$. The overall decryption succeeds if $\prod_i (\alpha'_i / \beta'_i) = 1$.

Suppose the internal records (S, α, β) are labeled as (S_j, α_j, β_j) for $j \geq 1$. Then for some constants $r, s \in \mathbb{G}$ and exponent $p \in \{0, 1\}$, we have that $\alpha'_i / \beta'_i = (r/s)(\alpha_j / \beta_j)^p$. Now, let $\gamma'_i = \alpha'_i / \beta'_i$. We denote γ'_i as a linear function in a single formal variable of the form $\gamma_j = \alpha_j / \beta_j$. The adversary's view is independent of the choice of γ_j's, except for the fact that $\prod_{j \in J} \gamma_j = 1$ for some disjoint sets J.

Recall that the overall decryption of C is successful if $\prod_i \gamma'_i = 1$. However, note that if is only with negligible probability that $\prod_i \gamma'_i = 1$ when evaluated on the simulator's choice of γ_j's, but $\prod_j \gamma'_i \neq 1$ as a polynomial. Thus consider a simulator $\mathcal{S}_4$ that sends (DUMMY, C) to the functionality whenever $\prod_i \gamma'_i \neq 1$, *as a polynomial* (accounting for the constraints on γ_j's). This simulator's behavior differs from $\mathcal{S}_3$ with only negligible probability.

Suppose $\prod_i \gamma'_i = 1$ as a polynomial, and let J be such that that we have a constraint of the form $\prod_j \gamma_j = 1$. Then for all $j \in J$, there exists n_J such that $\perp \neq \mathsf{RigExtract}_{SK}(C_i, S_j)$ for exactly n_J values of i. In other words, for each $j \in J$, the variable γ_j appears in the expansion of $\prod_i \gamma'_i$ with the same multiplicity. Then $\mathcal{S}$ 4 can do the following when $\mathcal{A}$ outputs a ciphertext $C = (C_1, \ldots, C_\ell)$:

- If for some C_i, we have $D(C_i) = \perp$, the ciphertext is invalid; send (DUMMY, C) to the functionality.
- Otherwise, compute $(\alpha'_i, \beta'_i) = D(C_i)$. If $\prod_i \alpha'_i / \beta'_i \neq 1$, when viewed as a polynomial in variables α_j / β_j, then send (DUMMY, C) to the functionality.
- Otherwise, let I be the set of indices such that $\perp \neq (\alpha'_i, \beta'_i) \leftarrow \mathsf{Dec}_{SK}(C_i)$. Let $(r_i, s_i) \leftarrow \mathsf{RigExtract}_{SK}(C_i, S_j)$ for each $i \notin I$. We have that $\prod_{i \in I} (\alpha'_i / \beta'_i) = 1$ and $\prod_{i \notin I} (r_i / s_i) = 1$ by the above argument. Then send $(\text{POST}, \ell, m_0, \text{H})$ to the functionality, where $m_0 = \prod_{i \in I} \alpha_i \prod_{i \notin I} r_i$ and H contains with multiplicity n_J the handle that resulted when $\{(\alpha_j, \beta_j) \mid j \in J\}$ were generated.

Except with negligible probability, $\mathcal{S}_4$ interacts identically with the functionality as $\mathcal{S}_3$. However, note that $\mathcal{S}_4$ does not actually use the α_j, β_j values that are recorded for each call to RigEnc. Thus $\mathcal{S}_4$ can be successfully implemented even if the functionality does not reveal m in messages of the form $(\text{HANDLE-REQ}, \mathsf{sender}, \ell, m)$. Therefore $\mathcal{S}_4$ is a suitable simulator for $\mathcal{F}_{\mathbb{G}}$ itself, and $\text{EXEC}[\mathcal{Z}, \mathcal{S}_3, \pi_{\mathsf{dummy}}, \mathcal{F}_2] \approx \text{EXEC}[\mathcal{Z}, \mathcal{S}_4, \pi_{\mathsf{dummy}}, \mathcal{F}_{\mathbb{G}}]$ for all environments $\mathcal{Z}$. □

6.8 Conclusion & Open Problems

Improved constructions. A natural next step is to address encryption schemes whose homomorphic operations are more expressive. Currently, all of our constructions support homomorphic transformations related to a group operation. Homomorphic operations involving other algebraic structures (ring, field, or vector space operations) may also prove useful in protocol applications.

Our construction of a transformation-hiding HCCA-secure scheme is quite efficient, having only a small additive overhead over a comparable CCA-secure scheme. However, our unlinkable scheme is much more impractical than the state of the art for CCA security. It may be that unlinkability can be achieved using more generic hardness assumptions than DDH, possibly resulting in a significant improvement in efficiency.

Anonymity. In some applications, it is useful for an encryption scheme to have the additional property of *receiver-anonymity* (also known as *key-privacy*), as introduced by Bellare et al. [7]. Receiver-anonymity means, essentially, that in addition to hiding the underlying plaintext message, a ciphertext does not reveal the public key under which it was encrypted. Encryption schemes with this property are important tools in the design of many systems [37]. The special case of rerandomizable, anonymous, RCCA-secure encryption has interesting applications in mix-nets [43] and anonymous P2P routing [83].

The way we have defined the syntax of the CTrans feature of a homomorphic encryption scheme (i.e, so that it does not require the "correct" public key in addition to the ciphertext), it remains a meaningful feature even in an environment where receiver-anonymity is utilized.

To model the property of receiver-anonymity for HCCA schemes, we consider an anonymous, multi-user variant of the $\mathcal{F}_{\text{HMP}}^{\mathcal{T}}$ UC functionality. This variant allows multiple users to register IDs, and senders to post messages destined for a particular ID. The functionality does not reveal the handle's recipient in its HANDLE-ANNOUNCE broadcasts (or in its HANDLE-REQ requests to the adversary).

Our indistinguishability-based security definitions can also be extended in a simple way to account for receiver-anonymity. We call a homomorphic encryption scheme *HCCA-anonymous* if it is HCCA secure and if the RigEnc and RigExtract procedures from the HCCA security definition can be implemented without the public or private keys (i.e, RigEnc takes no arguments and RigExtract takes only a ciphertext and a saved state).

We also consider an additional correctness requirement on schemes, which is natural in the context of multiple users: With overwhelming probability over $(PK, SK) \leftarrow \mathsf{KeyGen}$ and $(PK', SK') \leftarrow \mathsf{KeyGen}$, we require that $\mathsf{Dec}_{SK'}(\mathsf{Enc}_{PK}(\mathsf{msg})) = \perp$ for every $\mathsf{msg} \in \mathcal{M}$, with probability 1 over the randomness of Enc. In other words, ciphertexts honestly encrypted for one user do not successfully decrypt for another user.

Via a similar argument to the proof of Theorem 6.9, it can be seen that any HCCA-anonymous, unlinkable scheme which satisfies the additional correctness property is a secure realization of the anonymous variant of $\mathcal{F}_{\text{HMP}}^{\mathcal{T}}$.

Note that this notion of anonymity is a chosen-ciphertext and not a chosen-plaintext (simple ciphertext indistinguishability) one. Our construction does not achieve HCCA-anonymity, since it is possible to combine a ciphertext with a public key and obtain a valid ciphertext if and only if the original ciphertext was encrypted under that public key. We consider it an interesting and important open problem to construct an anonymous, unlinkably homomorphic HCCA encryption scheme, for any $\mathcal{T}$.

Alternative UC security definition. For simplicity, we have defined our ideal UC functionality $\mathcal{F}_{\text{HMP}}^{\mathcal{T}}$ in such a way that the adversary is notified on-line every time a handle is generated. As pointed out in [81], this paradigm does not allow the most flexibility. A more general-purpose functionality would be one in which parties privately (i.e., without the adversary being notified) generate new handles, and can have arbitrary control over how the handles are sent to other parties. If a handle never reaches the adversary, the adversary should not know that it was ever generated. To model this, the functionality can be modified so that the adversary is not notified each time a new handle is generated; instead, following [81], the adversary supplies a handle-generating algorithm during the set-up phase so that handles can be generated without the adversary's intervention/notification.

To securely realize such a functionality via a homomorphic encryption scheme, we must ensure that the scheme satisfies an additional security property; namely, that ciphertexts reveal (even to the receiver) at most the *cumulative effect* of all the transformations that have been applied — in particular, the ciphertext does not reveal which particular sequence of transformations has been applied. This property can be specified more formally as a security experiment, where an adversary supplies a ciphertext ζ and two transformations T_1 and T_2. The challenger flips a fair coin and returns either either $\mathsf{CTrans}(\zeta, T_2 \circ T_1)$ or $\mathsf{CTrans}(\mathsf{CTrans}(\zeta, T_1), T_2)$, correspondingly. We insist that the adversary cannot correctly guess the coin with nonnegligible advantage.

With this additional security requirement, an analog of Theorem 6.9 holds for this non-broadcasting definition of $\mathcal{F}_{\text{HMP}}^{\mathcal{T}}$. Our construction does indeed satisfy this additional property, since the two distributions $\mathsf{CTrans}(\zeta, T_2 \circ T_1)$ and $\mathsf{CTrans}(\mathsf{CTrans}(\zeta, T_1), T_2)$ are identical.

Repost-test. In $\mathcal{F}_{\text{HMP}}^{\mathcal{T}}$, when an honest party Alice receives a post from Bob and then another from Carl, Alice has no way of knowing if Carl's message was derived from Bob's (via $\mathcal{F}_{\text{HMP}}^{\mathcal{T}}$'s REPOST feature), or via an independent POST command. In fact, the only time $\mathcal{F}_{\text{HMP}}^{\mathcal{T}}$ informs a recipient that a REPOST occurred is for the adversary's dummy handles.

We can easily modify our schemes and $\mathcal{F}_{\text{HMP}}^{\mathcal{T}}$ to provide such a *feature* for honest parties. We call this feature *repost-test*. In this variant of $\mathcal{F}_{\text{HMP}}^{\mathcal{T}}$, the recipient may issue an additional command $(\text{TEST}, \mathsf{handle}_1, \mathsf{handle}_2)$. The functionality returns a boolean indicating whether the two handles were the result of reposting a common handle (it keeps extra book-keeping to track the ancestor of each REPOST-generated handle).

To realize this modified functionality, we start with a realization of $\mathcal{F}_{\text{HMP}}^{\mathcal{T}}$ on message space $\mathcal{M}^{n+1}$, where $\mathcal{M}$ has superpolynomial size. Suppose every $T \in \mathcal{T}$ always preserves the $(n+1)$th component of the message. Then let $\mathcal{T}'$ be the restrictions of $T \in \mathcal{T}$ to the first n components.

We may then use $\mathcal{F}_{\text{HMP}}^{\mathcal{T}}$ to obtain a secure realization of $\mathcal{F}_{\text{HMP}}^{\mathcal{T}'}$ with repost-test feature in the following way: To post a message $(m_1, \ldots, m_n) \in \mathcal{M}^n$, choose a random $m_{n+1} \leftarrow \mathcal{M}$ and post $(m_1, \ldots, m_{n+1})$ to $\mathcal{F}_{\text{HMP}}^{\mathcal{T}}$. When reading a message, ignore the last component. To perform the repost-test on two handles, simply check whether the last components of their corresponding messages are equal.

Non-malleability beyond encryption. Our results demonstrate that it is possible to envision non-malleability as a precisely defined tradeoff, rather than an all-or-nothing prospect. Cryptographic objects that are non-malleable in this way combine the best aspects of flexible computation and integrity against malicious behavior. It is likely that this approach to non-malleability will be useful for primitives other than encryption schemes.

For example, one may envision a non-interactive zero-knowledge (NIZK) proof scheme in which the proof objects may be mauled into proofs of related statements, but only in certain ways. Indeed, NIZK proofs are an important component in *anonymous credential* schemes [6], and combining such malleability with an unlinkability feature would permit an anonymous *delegation* feature for the credentials.

APPENDIX A

Additional Proofs

A.1 Security Proof for Homomorphic Encryption Scheme

In this section, we give the full proof of Theorem 6.14, that our construction (Section 6.5.2) satisfies the correctness properties for a homomorphic encryption scheme, is unlinkably homomorphic and is HCCA-secure, under the DDH assumption in $\mathbb{G}$ and $\widehat{\mathbb{G}}$.

The full details of the proof are carried out in the following sections. We use the notational conventions of Section 6.5.2.

A.1.1 Rigged Encryption & Extraction

To show HCCA security, we must demonstrate RigEnc and RigExtract procedures. First, we factor out some subroutines that are common to the "rigged" and non-rigged encryption procedures:

Ciphertext generation. $\mathsf{GenCiph}_{PK}((m_1,\ldots,m_n),\mu)$: Pick random $x \in \mathbb{Z}_p$, $y \in \mathbb{Z}_p^*$ and random $u \in \widehat{\mathbb{G}}$, and output

$$\begin{array}{cccccccc} g_1^{(x+z_1)u}, & \ldots, & g_4^{(x+z_4)u}, & m_1 C_1^x, & \ldots, & m_n C_n^x, & (DE^\mu)^x; \\ g_1^{yu}, & \ldots, & g_4^{yu}, & C_1^y, & \ldots, & C_n^y, & (DE^\mu)^y; \\ & & & & & & \mathsf{MEnc}_{\widehat{PK}}(u) \end{array}$$

Deriving purported plaintext. $\mathsf{PurpMsg}_{SK}(\zeta,u)$: Strip off u and $\vec{z}$ from the exponents as follows: For $j = 1,\ldots,4$: set $\overline{X}_j = X_j^{1/u} g_j^{-z_j}$ and $\overline{Y}_j = Y_j^{1/u}$. Output $(m_1,\ldots,m_n)$, where $m_i = C_{X,i} / \prod_{j=1}^4 \overline{X}_j^{c_{i,j}}$.

Checking ciphertext integrity. $\mathsf{Integrity}_{SK}(\zeta,u,\mu)$: Strip off u and $\vec{z}$ from the exponents as follows: For $j = 1,\ldots,4$: set $\overline{X}_j = X_j^{1/u} g_j^{-z_j}$ and $\overline{Y}_j = Y_j^{1/u}$. If $\overline{Y}_1 = \cdots = \overline{Y}_4 = 1$ (the identity element in $\mathbb{G}$), output 0. Otherwise, check the following constraints:

$$P_X \stackrel{?}{=} \prod_{j=1}^4 \overline{X}_j^{d_j + \mu e_j}; \qquad C_{Y,i} \stackrel{?}{=} \prod_{j=1}^4 \overline{Y}_j^{c_{i,j}} \text{ (for each } i \in [n]);$$
$$P_Y \stackrel{?}{=} \prod_{j=1}^4 \overline{Y}_j^{d_j + \mu e_j}$$

If any fail, output 0, otherwise output 1.

We can view the scheme's Enc and Dec routines as using these subroutines:

$\mathsf{Enc}_{PK}(m_1,\ldots,m_n)$: Output $\mathsf{GenCiph}_{PK}((m_1,\ldots,m_n),\mathsf{H}(f(m_1,\ldots,m_n)))$.

$\mathsf{Dec}_{SK}(\zeta)$: Compute $u \leftarrow \mathsf{MDec}_{\widehat{SK}}(U)$. If $u = \perp$, output $\perp$. Otherwise, set $(m_1,\ldots,m_n) = \mathsf{PurpMsg}_{SK}(\zeta,u)$. If $\mathsf{Integrity}_{SK}(\zeta,u,\mathsf{H}(f(m_1,\ldots,m_n))) = 1$, output $(m_1,\ldots,m_n)$; otherwise output $\perp$.

Now, we define the RigEnc and RigExtract procedures for use in our security proof:

$\mathsf{RigEnc}_{PK}()$: Pick random $\mu \leftarrow \mathbb{Z}_p$. Generate $\zeta \leftarrow \mathsf{GenCiph}_{PK}((1,\ldots,1),\mu)$, and output $(\zeta, S = \mu)$.

$\mathsf{RigExtract}_{SK}(\zeta,S)$: Compute $u \leftarrow \mathsf{MDec}_{\widehat{SK}}(U)$. If $u = \perp$, output $\perp$. Otherwise, set $(m_1,\ldots,m_n) = \mathsf{PurpMsg}_{SK}(\zeta,u)$. If $\mathsf{Integrity}_{SK}(\zeta,u,S) = 1$ and $(m_1,\ldots,m_n) \in \mathbb{H}$ (i.e., $T_{(m_1,\ldots,m_n)}$ is an allowed transformation), then output $T_{(m_1,\ldots,m_n)}$; otherwise output $\perp$.

A.1.2 Encryption & Decryption as Linear Algebra

In this section we characterize our construction using linear algebra, which will be useful in the security proof.

Public key constraints. First we examine what information is revealed to the adversary about the private key by the public key.

The following constraints relate the private keys to public keys (the first equation is in the field of order q, and the second is in the field of order p):

$$\begin{bmatrix}\vec{1} & 0\\ 0 & \vec{1}\end{bmatrix}\begin{bmatrix}\hat{G} & 0\\ 0 & \hat{G}\end{bmatrix}\begin{bmatrix}\vec{a}^\top\\ \vec{b}^\top\end{bmatrix} = \begin{bmatrix}\log A\\ \log B\end{bmatrix},\ \text{where } \hat{G} = \begin{bmatrix}\log \widehat{g}_1 & 0\\ 0 & \log \widehat{g}_2\end{bmatrix}$$

$$\begin{bmatrix}\vec{1} & & \\ & \ddots & \\ & & \vec{1}\end{bmatrix}\begin{bmatrix}G & & \\ & \ddots & \\ & & G\end{bmatrix}\begin{bmatrix}\vec{c}_1^\top\\ \vdots\\ \vec{c}_n^\top\\ \vec{d}^\top\\ \vec{e}^\top\end{bmatrix} = \begin{bmatrix}\log C_1\\ \vdots\\ \log C_n\\ \log D\\ \log E\end{bmatrix},\ \text{where } G = \begin{bmatrix}\log g_1 & \cdots & 0\\ \vdots & \ddots & \vdots\\ 0 & \cdots & \log g_4\end{bmatrix} \tag{A.1}$$

We call these constraints the *public-key constraints*.

Definition A.1. *Let $U = (\vec{V}, A_V, B_V, \vec{W}, A_W, B_W)$ be a DSME ciphertext. The two* DSME strands *of U with respect to a public key $(\widehat{g}_1, \widehat{g}_2, A, B)$ are:*

$$\vec{v} = (v_1, v_2),\ \textit{where } v_j = \log_{\widehat{g}_j} V_j$$
$$\vec{w} = (w_1, w_2),\ \textit{where } w_j = \log_{\widehat{g}_j} W_j$$

Observe that applying $\mathsf{MCTrans}(U, T_\sigma)$ gives a ciphertext whose two strands are $\vec{v} + r\vec{w}$ and $s\vec{w}$, for random $r \in \mathbb{Z}_q, s \in \mathbb{Z}_q^*$. In ciphertexts generated by MEnc, both strands are scalar multiples of the all-ones vector.

For ciphertexts in the main scheme, we define a similar notion of strands. However, in such a ciphertext, the first strand is "masked" by u and z_i's, and the second strand is masked by u.

Definition A.2. *Let $\zeta = (\vec{X}, \vec{C}_X, P_X; \vec{Y}, \vec{C}_Y, P_Y; U)$ be a ciphertext in the main scheme. The* strands *of ζ with respect to a public key $(g_1, \ldots, g_4, \ldots)$ and a value $u \in \widehat{\mathbb{G}}$ are:*

$$\vec{x} = (x_1, \ldots, x_4),\ \textit{where } x_i = (\log_{g_i} X_i)/u - z_i$$
$$\vec{y} = (y_1, \ldots, y_4),\ \textit{where } y_i = (\log_{g_i} Y_i)/u$$

As above, applying $\mathsf{CTrans}(\zeta, T_{\vec{\tau}})$ gives a ciphertext whose two strands are $\vec{x} + r\vec{y}$ and $s\vec{y}$, for random $r \in \mathbb{Z}_p, s \in \mathbb{Z}_p^*$. In ciphertexts generated by Enc, both strands are scalar multiples of the all-ones vector.

Decryption constraints. Let $\widehat{SK} = (\vec{a}, \vec{b})$ be a DSME private key, let $U = (\vec{V}, A_V, B_V, \vec{W}, A_W, B_W)$ be a DSME ciphertext, and let $\vec{v}, \vec{w}$ be its two strands with respect to the corresponding public key. Then $\mathsf{MDec}_{\widehat{SK}}(U) = u \neq \perp$ if and only if $\vec{w}$ is a nonzero vector and the following constraints hold in the field of order q:

$$\begin{bmatrix}\vec{v} & 0\\ \vec{w} & 0\\ 0 & \vec{v}\\ 0 & \vec{w}\end{bmatrix}\begin{bmatrix}\hat{G} & 0\\ 0 & \hat{G}\end{bmatrix}\begin{bmatrix}\vec{a}^\top\\ \vec{b}^\top\end{bmatrix} = \begin{bmatrix}\log(A_V/u)\\ \log A_W\\ \log B_V\\ \log B_W\end{bmatrix} \tag{A.2}$$

The logarithms are with respect to any fixed generator of $\widehat{\mathbb{G}}$.

Similarly, let $SK = (\widehat{SK}, \vec{c}_1, \ldots, \vec{c}_n, \vec{d}, \vec{e})$ be a private key and $\zeta = (\vec{X}, \vec{C}_X, P_X; \vec{Y}, \vec{C}_Y, P_Y; U)$ be a ciphertext in the main scheme, such that $\mathsf{MDec}_{\widehat{SK}}(U) = u \neq \perp$. Let $\vec{x}$ and $\vec{y}$ denote the strands of ζ^* with respect to the public key and u.

Then $\mathsf{PurpMsg}_{SK}(\zeta, u) = (m_1, \ldots, m_n)$ and $\mathsf{Integrity}_{SK}(\zeta, u, \mu) = 1$ if and only if $\vec{y}$ is a nonzero vector and the

following constraints hold in the field of order p:

$$\begin{bmatrix} \vec{x} & & & & \\ & \ddots & & 0 & 0 \\ & & \vec{x} & & \\ \vec{y} & & & & \\ & \ddots & & 0 & 0 \\ & & \vec{y} & & \\ 0 & & \vec{x} & \mu\vec{x} & \\ 0 & & \vec{y} & \mu\vec{y} & \end{bmatrix} \begin{bmatrix} G & & \\ & \ddots & \\ & & G \end{bmatrix} \begin{bmatrix} \vec{c}_1^{\top} \\ \vdots \\ \vec{c}_n^{\top} \\ \vec{d}^{\top} \\ \vec{e}^{\top} \end{bmatrix} = \begin{bmatrix} \log(C_{X,1}/m_1) \\ \vdots \\ \log(C_{X,n}/m_n) \\ \log C_{Y,1} \\ \vdots \\ \log C_{Y,n} \\ \log P_X \\ \log P_Y \end{bmatrix} \tag{A.3}$$

The logarithms are with respect to any fixed generator of $\mathbb{G}$.

We call each constraint in these systems of equations the *decryption constraints*, and refer to them by the name of the ciphertext component that is involved (A_V, A_W, P_X, etc.).

The adversary receives ciphertexts throughout the HCCA experiment from GRigEnc oracles and as the challenge ζ^*. These ciphertexts gives the adversary's view additional constraints on the private key.

Observation: Ciphertexts that are generated by GenCiph induce constraints that are linearly dependent on the constraints given by the public key, because their strands are scalar multiples of the appropriate all-ones vectors.

Lemma A.3. *For all key pairs $(\widehat{PK}, \widehat{SK})$, all (purported) ciphertexts U, and all U' in the support of* $\mathsf{MCTrans}(U, T_\sigma)$*, we have* $\mathsf{MDec}_{\widehat{SK}}(U') = T_\sigma(\mathsf{MDec}_{\widehat{SK}}(U))$.

Proof. If $\vec{v}$ and $\vec{w}$ are the two strands of U, then the two strands of U' are $\vec{v} + r\vec{w}$ and $s\vec{w}$ for some $r \in \mathbb{Z}_q$, $s \in \mathbb{Z}_q^*$. The two strands of U' are linear combinations of the strands of U, and the three values checked by a decryption constraint (B_V,A_W,B_W) are the corresponding combinations of the values from U. Thus, a particular decryption constraint fails on U' if and only if it fails for U. Furthermore, $s \neq 0$, so $\vec{w}$ is a nonzero vector if and only if $s\vec{w}$ is a nonzero vector. Finally, it is straight-forward to check that the purported plaintext of U' is σ times the purported plaintext of U. □

Lemma A.4. *For all key pairs (PK, SK), all (purported) ciphertexts ζ, and all ζ' in the support of* $\mathsf{CTrans}(\zeta, T_{\vec{\tau}})$*, we have* $\mathsf{Dec}_{SK}(\zeta') = T_{\vec{\tau}}(\mathsf{Dec}_{SK}(\zeta))$.

Proof. First, by the above lemma, the DSME component of ζ will fail to decrypt if and only if the DSME component of ζ' fails to decrypt.

Otherwise, the two strands of ζ' (with respect to the decryption of its DSME component) are linear combinations of the strands of ζ (with respect to the decryption of its DSME component). A similar argument to above shows that a decryption check fails on ζ' if and only if the same check fails on ζ; and the ratios of the purported plaintexts are $\vec{\tau}$. □

A.1.3 Decisional Diffie-Hellman Assumption

We now describe a more intricate indistinguishability assumption, which is implied by the standard DDH assumption in $\mathbb{G}$ and $\widehat{\mathbb{G}}$.

First, consider the following two distributions:

- $\mathsf{DDH}(\mathbb{G}, j)$ **distribution.** Pick random elements $g_1, \ldots, g_j \leftarrow \mathbb{G}$, and pick a random $v \leftarrow \mathbb{Z}_p$, where $|\mathbb{G}| = p$. Output $(g_1, \ldots, g_j, g_1^v, \ldots, g_j^v)$.
- $\mathsf{Rand}(\mathbb{G}, j)$ **distribution.** Pick random elements $g_1, \ldots, g_j \leftarrow \mathbb{G}$, and pick random $v_1, \ldots, v_j \leftarrow \mathbb{Z}_p$, where $|\mathbb{G}| = p$. Output $(g_1, \ldots, g_j, g_1^{v_1}, \ldots, g_j^{v_j})$.

We will require distributions of this form with $j = 2$ and $j = 4$, in different groups. Note that for fixed n, the standard DDH assumption in $\mathbb{G}$ (which is the special case of $j = 2$) implies that the above distributions are indistinguishable. To see this, consider a hybrid distribution in which the first k exponents are randomly chosen, and the remaining exponents are all equal. The standard DDH assumption is easily seen to imply that the kth hybrid distribution is indistinguishable from the $(k+1)$st.

Now consider the following two "double-strand" distributions:

- DS-DDH$(\mathbb{G}, j)$ **distribution.** Pick random elements $g_1, \ldots, g_j \leftarrow \mathbb{G}$, and pick random $v, w \leftarrow \mathbb{Z}_p$, where $|\mathbb{G}| = p$. Output $(g_1, \ldots, g_j, g_1^v, \ldots, g_j^v, g_1^w, \ldots, g_j^w)$.
- DS-Rand$(\mathbb{G}, j)$ **distribution.** Pick random elements $g_1, \ldots, g_j \leftarrow \mathbb{G}$, and pick random $v_1, \ldots, v_j, w_1, \ldots, w_j \leftarrow \mathbb{Z}_p$, where $|\mathbb{G}| = p$. Output $(g_1, \ldots, g_j, g_1^{v_1}, \ldots, g_j^{v_j}, g_1^{w_1}, \ldots, g_j^{w_j})$.

Again, a simple hybrid argument shows that if the DDH$(\mathbb{G}, j)$ and Rand$(\mathbb{G}, j)$ distributions are indistinguishable, then so are DS-DDH$(\mathbb{G}, j)$ and DS-Rand$(\mathbb{G}, j)$. We call elements in the support of these distributions *double-strand tuples of length* j.

Finally, our security proofs rely on the indistinguishability of the following two distributions:

- Pick $K_0 \leftarrow$ DS-DDH$(\mathbb{G}, 4)$, and pick $K_1 \leftarrow$ DDH$(\widehat{\mathbb{G}}, 2)$. Output (K_0, K_1).
- Pick $K_0 \leftarrow$ DS-Rand$(\mathbb{G}, 4)$, and pick $K_1 \leftarrow$ Rand$(\widehat{\mathbb{G}}, 2)$. Output (K_0, K_1).

A final hybrid argument shows that if DS-DDH$(\mathbb{G}, 4)$ and DS-Rand$(\mathbb{G}, 4)$ are indistinguishable, and DDH$(\widehat{\mathbb{G}}, 2)$ and Rand$(\widehat{\mathbb{G}}, 2)$ are also indistinguishable, then the above two distributions are indistinguishable.

A.1.4 The Alternate Encryption Procedure

We now describe the alternate method of generating ciphertexts AltGenCiph. As a component, it uses AltMEnc, an alternate encryption procedure for the DSME scheme. Both of these procedures use the secret keys instead of the public keys to generate ciphertexts.

DSME alternate encryption: AltMEnc$_{\widehat{SK}}(u)$**.**

- Pick random $v_1, v_2 \in \mathbb{Z}_q$ and $w \in \mathbb{Z}_q^*$. For $j = 1, 2$ let $V_j = \widehat{g}_j^{v_j}$ and $W_j = \widehat{g}_j^{w}$ (alternatively, in the analysis below we also consider V_1, V_2 as inputs instead).
- Output $(\vec{V}, A_V, B_V, \vec{W}, A_W, B_W)$, where

$$\vec{V} = (V_1, V_2) \qquad A_V = u \cdot \textstyle\prod_{j=1}^{2} V_j^{a_j} \qquad B_V = \textstyle\prod_{j=1}^{2} V_j^{b_j}$$
$$\vec{W} = (W_1, W_2) \qquad A_W = \textstyle\prod_{j=1}^{2} W_j^{a_j} \qquad B_W = \textstyle\prod_{j=1}^{2} W_j^{b_j}$$

Alternate ciphertexts: AltGenCiph$_{SK}((m_1, \ldots, m_n), \mu)$**.**

- Pick random $x_1, \ldots, x_4, y_1, \ldots, y_4 \in \mathbb{Z}_p$. For $j = 1, \ldots, 4$, set $\overline{X}_j = g_j^{x_j}$ and $\overline{Y}_j = g_j^{y_j}$, (alternatively, in the analysis below we also consider $\overline{X}_j, \overline{Y}_j$ as inputs instead).
- Pick random $u \in \widehat{\mathbb{G}}$, set $U \leftarrow$ AltMEnc$_{\widehat{SK}}(u)$, Compute:

$$X_j = (\overline{X}_j g_j^{z_j})^u; \qquad C_{X,i} = m_i \textstyle\prod_{j=1}^{4} \overline{X}_j^{c_{i,j}}; \qquad P_X = \textstyle\prod_{j=1}^{4} \overline{X}_j^{d_j + \mu e_j};$$
$$Y_j = \overline{Y}_j^u; \qquad C_{Y,i} = \textstyle\prod_{j=1}^{4} \overline{Y}_j^{c_{i,j}}; \qquad P_Y = \textstyle\prod_{j=1}^{4} \overline{Y}_j^{d_j + \mu e_j};$$

- Finally, output $\zeta = (\vec{X}, \vec{C}_X, P_X; \vec{Y}, \vec{C}_Y, P_Y; U)$

These alternate encryption procedures differ from the normal encryption procedures in that they generate ciphertexts whose decryption constraints are not linearly dependent on the public key constraints. The DSME alternate encryption generates a ciphertext whose first strand is random, and the alternate encryption generates a ciphertext whose two strands are both random. The remainder of the ciphertexts are constructed using the *private keys* to ensure that the decryption constraints are satisfied.

A hybrid HCCA experiment. Consider a variant of the HCCA experiment, where the challenge ciphertext is generated using AltGenCiph. That is, in the challenge phase of the experiment, the implicit call to GenCiph$_{PK}$ is replaced with an identical call to AltGenCiph$_{SK}$.

Lemma A.5. *In the hybrid HCCA experiment (where ζ^* is generated using* AltGenCiph*), conditioned on a negligible-probability event not occurring, ζ^* is distributed independently of the randomness u, and the bit b in the experiment, even given the public key. When $b = 1$, ζ^* is also distributed independently of the random choice of μ used in* RigEnc*.*

Proof. Given a DSME ciphertext from AltMEnc with first strand $\vec{v}^*$, the set $\{\vec{v}^*, \vec{1}\}$ forms a basis for the space of all DSME strands, with overwhelming probability. The adversary's view of the DSME private key $(\vec{a}, \vec{b})$ is constrained by the public key constraints in Equation A.1 and the decryption constraints given by ζ^* in Equation A.3. The constraints involving the second strand are linearly dependent on the public key constraints, as $\vec{w}$ is a scalar multiple of the all-ones vector. Combining these constraints and removing the redundant ones, we have:

$$\begin{bmatrix} \vec{1} & 0 \\ 0 & \vec{1} \\ \vec{v} & 0 \\ 0 & \vec{v} \end{bmatrix} \begin{bmatrix} \hat{G} & 0 \\ 0 & \hat{G} \end{bmatrix} \begin{bmatrix} \vec{a}^\top \\ \vec{b}^\top \end{bmatrix} = \begin{bmatrix} \log A \\ \log B \\ \log(A_V/u) \\ \log B_V \end{bmatrix},$$

The matrix on the left-hand side of this equation is nonsingular.

Similarly, in a ciphertext from AltGenCiph, with overwhelming probability we have that for *every* $u \in \widehat{\mathbb{G}}$, $\{\vec{x}, \vec{y}, \vec{1}, \vec{z}\}$ form a basis for the space of all strands, where $\vec{x}$ and $\vec{y}$ are the strands of the challenge ciphertext with respect to u. Then the adversary's view of the private key is constrained as follows:

$$\begin{bmatrix} \vec{1} & & & & & \\ & \ddots & & & & \\ & & & & & \vec{1} \\ \vec{x} & & & & & \\ & \ddots & & & & \\ & & \vec{x} & & & \\ \vec{y} & & & & & \\ & \ddots & & & & \\ & & \vec{y} & & & \\ & & & \vec{x} & \mu\vec{x} & \\ & & & \vec{y} & \mu\vec{y} & \end{bmatrix} \begin{bmatrix} G & & \\ & \ddots & \\ & & G \end{bmatrix} \begin{bmatrix} \vec{c}_1^\top \\ \vdots \\ \vec{c}_n^\top \\ \vec{d}^\top \\ \vec{e}^\top \end{bmatrix} = \begin{bmatrix} \log C_1 \\ \vdots \\ \log C_n \\ \log D \\ \log E \\ \log(C_{X,1}/m_1) \\ \vdots \\ \log(C_{X,n}/m_n) \\ \log C_{Y,1} \\ \vdots \\ \log C_{Y,n} \\ \log P_X \\ \log P_Y \end{bmatrix}$$

Here $(m_1, \ldots, m_n)$ and μ denote the inputs to AltGenCiph.

Note that when $\{\vec{1}, \vec{x}, \vec{y}\}$ are linearly independent, the matrix on the left-hand side of this equation is nonsingular, for *every* $\mu \in \mathbb{Z}_p$.

Now fix a public key and challenge ciphertext (one that avoids the event that $\{\vec{1}, \vec{x}, \vec{y}\}$ is not linearly independent). When $b = 0$ in the experiment, substitute the appropriate m_i and μ values in the previous system of equations. Then for every choice of $u \in \widehat{\mathbb{G}}$, there are an equal number of solutions for the private keys in this system of equations, as the matrix is nonsingular.

When $b = 1$ in the experiment, substitute $(m_1, \ldots, m_n) = (1, \ldots, 1)$. Then for every choice of $u \in \widehat{\mathbb{G}}$ and $\mu \in \mathbb{Z}_p$, there are an equal number of solutions for the private keys, as the matrix is nonsingular.

Since the underlying randomness in the HCCA experiment is the choice of these private keys, the choices of u, μ, and b are independent of each other and of the adversary's view in the experiment. □

Lemma A.6. *For every PPT adversary, its advantage in the HCCA experiment is negligibly close to its advantage in the hybrid HCCA experiment (when the challenge ciphertext is generated using* AltGenCiph*), if the DDH assumption holds in* $\mathbb{G}$ *and* $\widehat{\mathbb{G}}$*.*

Proof. If the DDH assumption holds for $\widehat{\mathbb{G}}$ and $\mathbb{G}$, then two the distributions described in Section A.1.3 are computationally indistinguishable.

Now consider a simulation of the HCCA experiment, where the input is from one of the above distributions. Let $(\widehat{g}_1, \widehat{g}_2, V_1, V_2)$ be the sample from either $\mathsf{DDH}(\widehat{\mathbb{G}}, 2)$ or $\mathsf{Rand}(\widehat{\mathbb{G}}, 2)$. Set $(\widehat{g}_1, \widehat{g}_2)$ as the corresponding part of the DSME public key, and generate the remainder of the keypair $(\vec{a}, \vec{b})$ honestly. To simulate the encryption of u^* from the challenge ciphertext with this keypair, use AltMEnc with the input values V_1, V_2.

Similarly, let $(g_1, \ldots, g_4, \overline{X}_1, \ldots, \overline{X}_4, \overline{Y}_1, \ldots, \overline{Y}_4)$ be the sample from either $\mathsf{DS\text{-}DDH}(\mathbb{G}, 4)$ or $\mathsf{DS\text{-}Rand}(\mathbb{G}, 4)$. Set $(g_1, \ldots, g_4)$ as the corresponding part of the public key and generate the remainder of the keypairs $(\vec{c}_i, \vec{d}, \vec{e})$ honestly. To simulate the encryption of the challenge ciphertext, use AltGenCiph and AltMEnc with the private keys and input values $\overline{X}_1, \ldots, \overline{X}_4, \overline{Y}_1, \ldots, \overline{Y}_4$.

It is straight-forward to check that when the input is sampled from the first distribution (i.e, the 2 tuples come from the appropriate DDH distributions), the ciphertext is distributed statistically close to a "normal" encryption from GenCiph and MEnc (the distribution is identical when conditioned to avoid the negligible-probability event that $\overline{Y}_1 = \cdots = \overline{Y}_4 = 1$). If the input is sampled from the second distribution (i.e, the 2 tuples comes from the appropriate random distributions), then the ciphertext is distributed identically as an encryption from AltGenCiph.

The rest of this simulation of the HCCA experiment can be implemented in polynomial time. Thus, the outcomes of the two simulations must not differ by more than a negligible amount. □

A.1.5 Decryption Queries

In this section, we argue that with overwhelming probability, the only ciphertexts accepted by the decryption oracles in the HCCA experiment are ciphertexts of the "expected" form (from the supports of Enc or CTrans).

In this section, we let ζ^* denote the challenge ciphertext in the hybrid HCCA experiment, which was generated via AltGenCiph.

Our arguments in this section generally follow the same structure. Fix a set of constraints induced by a public key and challenge ciphertext ζ^* in the hybrid HCCA experiment. Call a query to the Dec, GRigExtract, or RigDec oracle *bad* if that oracle rejects (outputs $\perp$) for an overwhelming fraction of private keys which are consistent with those constraints.

We show that any ciphertext ζ not of the "expected" form is such a bad ciphertext, while on the other hand, ciphertexts which are of the expected form are actually rejected by none of the consistent private keys (and all private keys yield the same response from the oracle).

The following lemma establishes the significance of classifying ciphertexts in this way.

Lemma A.7. *With overwhelming probability,* all *bad queries are rejected in the HCCA experiment.*

Proof. By definition, the response from the oracle for a non-bad query does not introduce any new constraints on the private keys, as they all yield the same oracle response.

Consider the first bad ciphertext submitted to an oracle. At that time, from the adversary's view, the private key is distributed uniformly among all keys consistent with the constraints induced by ζ^* and the public key. Thus it is only with negligible probability that the oracle will return something other than $\perp$. Conditioned on it returning $\perp$, the adversary learns that a negligible fraction of private keys are ruled out. Let ν be a negligible upper bound for this fraction. The correct private key remains distributed among the $(1 - \nu)$ fraction of remaining keys, from the adversary's view.

By a union bound, if the adversary makes N *bad* queries to this decryption oracle, at least one of them is accepted with probability at most $N\nu$. Since the adversary makes a polynomial (in the security parameter) number of queries, this probability is negligible. □

The simplest way a ciphertext can be *bad* is if it one of its decryption integrity constraints (Equation A.2 and Equation A.3) is linearly independent of the constraints given by the public key and challenge ciphertext.

DSME Decryption

Lemma A.8. *Fix a DSME public key and challenge ciphertext U^* in the hybrid HCCA experiment. Let U be an additional DSME ciphertext. Suppose u^* and u are the purported plaintexts of U^* and U, respectively, as computed by $\mathsf{MDec}_{\widehat{SK}}$. Then there exist fixed (with respect to the adversary's view) values $\pi = \pi(U)$ and $\sigma = \sigma(U)$ such that $u = \sigma(u^*)^\pi$.*

Note that even though in the hybrid HCCA experiment, the value of u^* is independent of the adversary's view, the values π and σ are fixed.

Proof. Let $\vec{v}^*$ be the first strand of U^*, and let $\vec{v}$ be the first strand of U. We may unambiguously express $\vec{v} = \pi\vec{v}^* + \epsilon\vec{1}$ for some π, ϵ. Then π and $\sigma = A_V/(A_V^*)^\pi A^\epsilon$ are the values desired in the statement of the lemma.

The purported ciphertext of U is:

$$u = \frac{A_V}{\prod_{j=1}^2 V_j^{a_j}} = \frac{A_V}{\left[\prod_{j=1}^2 (V_j^*)^{a_j}\right]^\pi \left[\prod_{j=1}^2 (\widehat{g}_j^*)^{a_j}\right]^\epsilon} = \frac{A_V}{[A_V^*/u^*]^\pi A^\epsilon} = \sigma(u^*)^\pi$$ □

Lemma A.9. *If the second strand of U is not a (nonzero) scalar multiple of $(1, 1)$, then $\mathsf{MDec}_{SK}(U) = \perp$ for all but a negligible fraction of private keys consistent with the adversary's view.*

Proof. Let $\vec{w}$ be the second strand of U, and $\vec{v}^*$ the first strand of U^*. If $\vec{w}$ is not in the span of $\vec{1}$, then $\vec{w} = \alpha\vec{v}^* + \beta\vec{1}$ for some $\alpha \neq 0$. The following constraint is checked while decrypting U:

$$1 \stackrel{?}{=} \frac{A_W}{\prod_{j=1}^{2} W_j^{a_j}} = \frac{A_W}{\left[\prod_{j=1}^{2} (V_j^*)^{a_j}\right]^{\alpha} \left[\prod_{j=1}^{2} \widehat{g}_j^{a_j}\right]^{\beta}} = \frac{A_W}{[A_V^*/u^*]^{\alpha} A^{\beta}}$$

The value of u^* is independent of the adversary's view and distributed uniformly in $\widehat{\mathbb{G}}$, and thus so is $(u^*)^{\alpha}$. Thus equality holds only with negligible probability. Also, if $\vec{w}$ is the zero vector, then U is explicitly rejected by MDec. □

Lemma A.10. *Let U be a DSME ciphertext, with π, σ as above, and suppose $\mathsf{MDec}(U) \neq \perp$ for a nonnegligible fraction of private keys consistent with the adversary's view. Then*

$$\pi = 0 \implies U \text{ is in the support of } \mathsf{MEnc}_{\widehat{PK}}(\sigma)$$
$$\pi = 1 \implies U \text{ is in the support of } \mathsf{MCTrans}(U^*, T_\sigma)$$

Proof. By the previous lemma, the second strand of U must be a nonzero multiple of $\vec{1}$.

If $\pi = 0$, then the first strand of U is also a multiple of $\vec{1}$. Say, $\vec{v} = v\vec{1}$ and $\vec{w} = w\vec{1}$, where $w \neq 0$. It is trivial to check that U decrypts to σ with nonnegligible probability only if $U = \mathsf{MEnc}_{\widehat{PK}}(\sigma; v, w)$.

If $\pi = 1$, then say $\vec{v} = \vec{v}^* + \beta\vec{1} = \vec{v}^* + s(\vec{w}^*)$ and $\vec{w} = \gamma\vec{1} = t(\vec{w}^*)$, for some $s \in \mathbb{Z}_q$, $t \in \mathbb{Z}_q^*$, since $\vec{w}^*$ (the second strand of U^*) is a nonzero scalar multiple of $\vec{1}$. Then it is trivial to check that U decrypts to σu^* only if $U = \mathsf{MCTrans}(U^*, T_\sigma; s, t)$. □

Decryption

Lemma A.11. *Let (ζ, μ) be an input to $\mathsf{Integrity}_{SK}$. Then it is a bad query unless there exists $\sigma \in \widehat{\mathbb{G}}$ such that one of the following cases holds:*

- *U is in the support of $\mathsf{MEnc}_{\widehat{PK}}(\sigma)$; and there exists $x \in \mathbb{Z}_p, y \in \mathbb{Z}_p^*$ such that $X_j = g_j^{(x+z_j)\sigma}$ and $Y_j = g_j^{y\sigma}$, for $j = 1, \ldots, 4$.*
- *U is in the support of $\mathsf{MCTrans}(U^*, T_\sigma)$; and there exists $s \in \mathbb{Z}_p, t \in \mathbb{Z}_p^*$ such that $X_j = (X_j^*(Y_j^*)^s)^\sigma$ and $Y_j = (Y_j^*)^{t\sigma}$, for $j = 1, \ldots, 4$; and $\mu = \mu^*$.*

We emphasize the similarity between these forms of ciphertexts and those produced by Enc and CTrans.

Proof. As parts of the challenge ciphertext ζ^*, the adversary is given the values: $X_j^* = g_j^{(x_j^* + z_j)u^*}$ and $Y_j^* = g_j^{y_j^* u^*}$, for some u^* corresponding to the decryption of U^* under $\widehat{SK}$. These values $\vec{X}$ and $\vec{Y}$ are fixed, even though the value of u^* is distributed independently of the adversary's view.

Similarly, when submitting a ciphertext ζ to an oracle, the adversary supplies the values: $X_j = g_j^{(x_j + z_j)u}$ and $Y_j = g_j^{y_j u}$ for some u, where $\vec{x}$ and $\vec{y}$ are the strands of the ciphertext with respect to u, and u is related to u^* via $u = \sigma(u^*)^\pi$.

With overwhelming probability in the HCCA experiment, these fixed vectors $\{(\vec{x}^* + \vec{z})u^*, \vec{y}^* u^*, \vec{z}, \vec{1}\}$ span the space of all strands. Thus we can write the following unique linear combination:

$$(\vec{x} + \vec{z})u = \alpha\Big((\vec{x}^* + \vec{z})u^*\Big) + \beta(\vec{y}^* u^*) + \gamma\vec{1} + \delta\vec{z} \tag{A.4}$$
$$\vec{y}u = \alpha'\Big((\vec{x}^* + \vec{z})u^*\Big) + \beta'(\vec{y}^* u^*) + \gamma'\vec{1} + \delta'\vec{z} \tag{A.5}$$

Note that the coefficients of this linear combination are fixed, independent of the randomness in u^*. Our analysis proceeds by showing that if these coefficients are not fixed in a particular way, then the ciphertext would be rejected by Dec with overwhelming probability over the remaining randomness in u^* and the private key.

Let $\vec{X}$ be as above, and consider the decryption constraint involving the P_X component of the ciphertext. Suppose the constraint holds for a nonnegligible fraction of consistent private keys. The constraint that is checked by Integrity is of the following form:

$$\begin{bmatrix}\vec{x} & \mu\vec{x}\end{bmatrix} \begin{bmatrix} G & \\ & G \end{bmatrix} \begin{bmatrix}\vec{d}^\top \\ \vec{e}^\top\end{bmatrix} \stackrel{?}{=} \begin{bmatrix}\log P_X\end{bmatrix}$$

The public key and challenge ciphertext constrain $\vec{d}$ and $\vec{e}$ as follows:

$$\begin{bmatrix} \vec{1} & \\ & \vec{1} \\ \vec{x}^* & \mu^*\vec{x}^* \\ \vec{y}^* & \mu^*\vec{y}^* \end{bmatrix} \begin{bmatrix} G & \\ & G \end{bmatrix} \begin{bmatrix} \vec{d}^\top \\ \vec{e}^\top \end{bmatrix} = \begin{bmatrix} \log D \\ \log E \\ \log P_X^* \\ \log P_Y^* \end{bmatrix}$$

We must have that $[\vec{x}\ \mu\vec{x}]$ is a linear combination of the above set of constraints with nonnegligible probability (over u^*). Furthermore, the coefficients of that linear combination must be fixed with nonnegligible probability over u^*, otherwise the "correct" value of the constraint will be distributed randomly in a superpolynomial-size domain as u^* varies.

Solving for $\vec{x}$ in the first equation and substituting, we have:

$$[\vec{x}\ \mu\vec{x}] = \frac{\gamma}{u}[\vec{1}\ \mu\vec{1}] + \frac{u^*}{u}\Big(\alpha[\vec{x}^*\ \mu\vec{x}^*] + \beta[\vec{y}^*\ \mu\vec{y}^*]\Big) + \left(\alpha\frac{u^*}{u} + \frac{\delta}{u} - 1\right)[\vec{z}\ \mu\vec{z}]$$

Let $\pi = \pi(U)$ and $\sigma = \sigma(U)$. We consider the following cases:

- If $\pi = 0$: Then $u = \sigma$ (independent of u^*) while u^*/u is distributed uniformly over $\widehat{\mathbb{G}}$. We must have $\alpha = \beta = 0$, otherwise the coefficients of $[\vec{x}^*\ \mu\vec{x}^*]$ and $[\vec{y}^*\ \mu\vec{y}^*]$ are distributed randomly with u^*/u. Furthermore, observe that $[\vec{z}\ \mu\vec{z}]$ is linearly independent of the constraints on the adversary's view for any μ, since $\vec{z}$ is linearly independent of $\{\vec{1}, \vec{x}^*, \vec{y}^*\}$. Thus, its coefficient must be zero with nonnegligible probability. This happens only when $\delta = \sigma$.

  Combining everything, we must have $X_j = g_j^{(x+z_i)\sigma}$ for some $x \in \mathbb{Z}_p$.

- If $\pi = 1$: Then $u^*/u = 1/\sigma$ while u (when appearing alone) is distributed uniformly over $\widehat{\mathbb{G}}$. We must have $\gamma = 0$, otherwise the coefficient of $[\vec{1}\ \mu\vec{1}]$ is distributed randomly with u^*. Again, we must have the coefficient of $[\vec{z}\ \mu\vec{z}]$ equal to zero with nonnegligible probability. Substituting, we get that $\delta = 0$ and $\alpha = \sigma$. Finally, we must have $\mu = \mu^*$, or else $[\vec{x}^*\ \mu\vec{x}^*]$ is linearly independent of $[\vec{x}^*\ \mu^*\vec{x}^*]$.

  Combining everything, we must have $X_j = (X_j^*(Y_j^*)^s)^\sigma$ for some $s \in \mathbb{Z}_p$, and that $\mu = \mu^*$.

- If $\pi \notin \{0, 1\}$. Below (when discussing the second strand), we show that the ciphertext would fail its integrity checks with overwhelming probability.

Similarly, we consider the second strand's $\vec{Y}$ as a linear combination (Equation A.4). We then consider the integrity check on the P_Y 1 component:

$$[\vec{y}\quad \mu\vec{y}]\, G \begin{bmatrix} \vec{d}^\top \\ \vec{e}^\top \end{bmatrix} \stackrel{?}{=} [\log P_Y]$$

The public key and challenge ciphertext constrain $\vec{d}$ and $\vec{e}$ in the following way:

$$\begin{bmatrix} \vec{1} & \\ & \vec{1} \\ \vec{x}^* & \mu^*\vec{x}^* \\ \vec{y}^* & \mu^*\vec{y}^* \end{bmatrix} \begin{bmatrix} G & \\ & G \end{bmatrix} \begin{bmatrix} \vec{d}^\top \\ \vec{e}^\top \end{bmatrix} = \begin{bmatrix} \log D \\ \log E \\ \log P_X^* \\ \log P_Y^* \end{bmatrix}$$

We rewrite $[\vec{y}\ \mu\vec{y}]$, substituting according to Equation A.4 to obtain:

$$[\vec{y}\ \mu\vec{y}] = \frac{\gamma'}{u}[\vec{1}\ \mu\vec{1}] + \frac{u^*}{u}\Big(\alpha'[\vec{x}^*\ \mu\vec{x}^*] + \beta'[\vec{y}^*\ \mu\vec{y}^*]\Big) + \left(\alpha'\frac{u^*}{u} + \frac{\delta'}{u}\right)[\vec{z}\ \mu\vec{z}]$$

Similar to the case of the first strand, we see that the coefficient $(\alpha' u^* + \delta')/u$ must be zero with nonnegligible probability over the randomness in u^*. This is only possible with $\alpha' = \delta' = 0$. We further consider 3 cases of π:

- If $\pi = 0$, then $u = \sigma$ and u^*/u is uniform in $\widehat{\mathbb{G}}$. We see that the coefficient of $[\vec{y}^*\ \mu\vec{y}^*]$ is $\beta' u^*/u$, so we must have $\beta' = 0$. Then $\gamma' \neq 0$, since otherwise $\vec{y}$ is the all-zeroes vector, and the ciphertext would be unconditionally rejected by Integrity.

  This implies that $Y_j = g_j^{y\sigma}$ for some $y \in \mathbb{Z}_p^*$.

- If $\pi = 1$, then $u^*/u = 1/\sigma$, while u is uniform in $\widehat{\mathbb{G}}$ when appearing alone. We see that the coefficient of $[\vec{1}\ \mu\vec{1}]$ is γ'/u, so we must have $\gamma' = 0$. Then $\beta' \neq 0$, since otherwise $\vec{y}$ is the all-zeroes vector and the ciphertext would be rejected by Integrity.

  This implies that $Y_j = (Y_j^*)^{t\sigma}$ for some $t \in \mathbb{Z}_p^*$.

- $\pi \notin \{0, 1\}$: First, observe that if $\mu \neq \mu^*$, then $[\vec{y}^*\ \mu\vec{y}^*]$ is independent of the view constraints, and we must have $\beta' = 0$. Then $\alpha' = \beta' = \delta' = 0$, and $[\vec{y}\ \mu\vec{y}] = \gamma'/u[\vec{1}\ \mu\vec{1}]$. Since u is uniformly distributed in $\widehat{\mathbb{G}}$, we must have $\gamma' = 0$. But then $\vec{y}$ is the all-zeroes vector and the ciphertext is unconditionally rejected by Integrity.

  Therefore we may assume $\mu = \mu^*$ if the ciphertext is to pass its integrity checks with nonnegligible probability. We consider the following two constraints simultaneously (simplifying via $\alpha' = \delta' = 0$):

$$\begin{aligned}
\begin{bmatrix} \log C_{Y,1} \\ \log P_Y \end{bmatrix} &\stackrel{?}{=} \begin{bmatrix} \vec{y} & & \\ & \vec{y} & \mu\vec{y} \end{bmatrix} \begin{bmatrix} G & & \\ & G & \\ & & G \end{bmatrix} \begin{bmatrix} \vec{c}_1^\top \\ \vec{d}^\top \\ \vec{e}^\top \end{bmatrix} \\
&= \begin{bmatrix} \frac{\gamma'}{u} & 0 & \frac{\beta' u^*}{u} & 0 \\ 0 & \frac{\gamma'}{u} & 0 & \frac{\beta' u^*}{u} \end{bmatrix} \begin{bmatrix} \vec{1} & & \\ & \vec{1} & \mu^*\vec{1} \\ \vec{y}^* & & \\ & \vec{y}^* & \mu^*\vec{y}^* \end{bmatrix} \begin{bmatrix} G & & \\ & G & \\ & & G \end{bmatrix} \begin{bmatrix} \vec{c}_1^\top \\ \vec{d}^\top \\ \vec{e}^\top \end{bmatrix} \\
&= \begin{bmatrix} \frac{\gamma'}{u} & 0 & \frac{\beta' u^*}{u} & 0 \\ 0 & \frac{\gamma'}{u} & 0 & \frac{\beta' u^*}{u} \end{bmatrix} \begin{bmatrix} \log C_1 \\ \log(DE^{\mu^*}) \\ \log C_{Y,1}^* \\ \log P_Y^* \end{bmatrix}
\end{aligned}$$

  If we multiply through both of these constraints and substitute according to $u = \sigma(u^*)^\pi$, we get the following two polynomials in u^*, which must be simultaneously zero with nonnegligible probability:

$$\begin{aligned}
(\sigma \log P_Y)(u^*)^\pi &\quad - (\beta'\sigma \log P_Y^*)u^* &&\quad - (\gamma' \log(DE^{\mu^*})) = 0 \\
(\sigma \log C_{Y,1})(u^*)^\pi &\quad - (\beta'\sigma \log C_{Y,1}^*)u^* &&\quad - (\gamma' \log C_1) = 0
\end{aligned}$$

  Note that these are polynomials in u^* of degree π, and no terms collect together, as $\pi \notin \{0, 1\}$. We now argue that these two polynomials cannot be simultaneously zero with nonnegligible probability, unless the ciphertext is degenerate and would be rejected on other grounds:

  - If one of the polynomials is not identically zero but has some coefficient equal to zero, then this polynomial is equivalent to (i.e, has the same roots as) an affine polynomial in a single variable; either u^* or $(u^*)^\pi$ or $(u^*)^{\pi-1}$. Each of these variables is uniform in $\widehat{\mathbb{G}}$, so the equation is satisfied with only negligible probability.
  - Otherwise, if the two polynomials have all nonzero coefficients and are identical up to scalar multiplication, then the three pairs of matching coefficients have the same ratios. In particular, we have the following equality (after cancellation):

$$\frac{\log(DE^{\mu^*})}{\log C_1} = \frac{\log P_Y^*}{\log C_{Y,1}^*}$$

    The challenge ciphertext (including the components P_Y^* and $C_{Y,1}^*$) is generated after C_1, D, E, and μ^* are fixed. It is only with negligible probability over the randomness of AltGenCiph that $C_{Y,1}^*$ an P_Y^* satisfy this condition. Thus it does not affect the outcome of our analysis to condition the entire HCCA experiment on this event not happening.
  - If the two polynomials have all nonzero coefficients and are not identical up to scalar multiplication, then some linear combination of them is affine either in the variable u^* or in $(u^*)^\pi$. The two original polynomial equations must have been simultaneously satisfied with noticeable probability. When both equations hold, then so does any linear combination of the two. But we have demonstrated a linear combination of the equations that is affine on a single variable which is distributed uniformly over $\widehat{\mathbb{G}}$, and thus is satisfied with only negligible probability.

– Otherwise one polynomial is identically zero. It is only with negligible probability that AltGenCiph generates a ciphertext with $\log C^*_{Y,1} = 0$ or $\log P_Y 1^* = 0$. Thus our analysis may be conditioned on these events not happening. We must have $\beta' = 0$ to make the u^* coefficient zero (since $\sigma \neq 0$ unconditionally). This makes a coefficient in the other polynomial zero as well. This case overlaps with the first case unless both polynomials are in fact identically zero. If both are identically zero, then by similar reasoning, we must have $\gamma' = 0$ ($\log C_1 = 0$ only with negligible probability over the key generation). But when $\beta' = \gamma' = 0$, $\vec{y}$ is the all-zeroes vector and the ciphertext is rejected by Integrity. □

We now characterize precisely which queries are accepted by the decryption oracles in the HCCA experiment, to show that their outputs are independent of b.

- RigDec when $b = 0$. Then RigDec is the normal Dec oracle. The challenge ciphertext ζ^* is an encryption of $(m_1, \ldots, m_n)$ given by the adversary, and $\mu^* = \mathsf{H}(f(m_1, \ldots, m_n))$.
  - In the $\pi = 0$ case, it is straight-forward to see that the ciphertext passes its integrity check with μ computed from the purported plaintext only if ζ is in the support of $\mathsf{Enc}_{PK}(\cdot)$.
  - In the $\pi = 1$ case, let $(m'_1, \ldots, m'_n) = \mathsf{PurpMsg}_{SK}(\zeta, u)$. Then we must have $\mathsf{H}(f(m_1, \ldots, m_n)) = \mathsf{H}(f(m'_1, \ldots, m'_1))$. By the collision-resistance of H, this only happens when the two plaintexts are in the same $\mathbb{H}$-coset. Thus $(m_1, \ldots, m_n) * (m'_1, \ldots, m'_n)^{-1} \in \mathbb{H}$, and so $(m'_1, \ldots, m'_n)$ is an allowed transformation of $(m_1, \ldots, m_n)$. It is straight-forward to see that the remaining components' integrity checks succeed only if ζ is in the support of $\mathsf{CTrans}(\zeta^*, \cdot)$.
- RigDec when $b = 1$. Then RigDec first calls RigExtract using μ^*, which are distributed independently of the adversary's view.
  - In the $\pi = 0$ case, ciphertexts of this form have a unique fixed (with respect to the adversary's view) value μ such that $\mathsf{Integrity}_{SK}(\zeta, \mu)$ succeeds. Since μ^* is distributed independently of the adversary's view, this happens with negligible probability.
  - In the $\pi = 1$ case, RigExtract accepts only if the purported plaintext of ζ is changed from ζ^* via an allowed transformation, and that the integrity constraints pass with the same μ^* value. It is straight-forward to see that these conditions hold only if ζ is in the support of $\mathsf{CTrans}(\zeta^*, \cdot)$.

  Then RigDec calls the normal Dec procedure:
  - The $\pi = 0$ case is analogous to the $b = 0$ case above, it is not affected by the difference in generation of ζ^*. It accepts only if ζ is in the support of $\mathsf{Enc}_{PK}(\cdot)$.
  - If $\pi = 1$, then the ciphertext's integrity is checked using μ values derived from the purported plaintext. The purported plaintext is information-theoretically fixed from the adversary's view. Only with negligible probability will μ equal the μ^* value used to generate ζ^*, which is necessary for Dec to accept with nonnegligible probability.

Thus, RigDec may be replaced by an (unbounded) oracle which does the following on input ζ.

- If ζ is in the support of $\mathsf{Enc}_{PK}(m_1, \ldots, m_n)$ then output $m_1, \ldots, m_n$.
- Otherwise, if ζ is in the support of $\mathsf{CTrans}(\zeta^*, T)$ for some $T \in \mathcal{T}$, then output $T(m^*_1, \ldots, m^*_n)$, where $m^*_1, \ldots, m^*_n$ is the challenge plaintext given by the adversary in the challenge phase.

By the above argument, the output of this oracle matches that of RigDec on *all* queries with overwhelming probability, in both the $b = 0$ and $b = 1$ branches of the HCCA experiment. Similarly, the argument for RigDec when $b = 0$ implies that we may also replace the Dec oracle given to the adversary in Phase I with an oracle which simply checks that its input is in the support of $\mathsf{Enc}_{PK}(\cdot)$.

We show a similar situation for inputs to the $\mathsf{GRigExtract}(i, \cdot)$ oracle:

- In the $\pi = 0$ case of an input ciphertext, the given ciphertext ζ is accepted only if its purported plaintext is an allowed transformation, and its integrity checks pass with respect to the same values as ζ_i (the ith output of GRigEnc). It is straight-forward to see that this is only possible for ζ from the support of $\mathsf{CTrans}(\zeta_i, \cdot)$.

- In the $\pi = 1$ case of inputs, we must have μ^* equal to the μ value used to generate ζ_i. Note that the μ value used to generate ζ_i is information-theoretically fixed given ζ_i and the public key (it was created using GenCiph instead of AltGenCiph).

  When $b = 1$, the two sets of μ values are only equal with negligible probability, as μ^* is chosen independently of μ.

  When $b = 0$, the μ^* value is computed via $\mathsf{H}(f(m_1^*, \ldots, m_n^*))$, where $m_1^*, \ldots, m_n^*$ is the challenge plaintext given by the adversary. For the adversary to be given ζ_i and subsequently be able to compute $(m_1^*, \ldots, m_n^*)$ which yield the correct μ value, the adversary must be able to compute discrete logs in $\mathbb{G}$.

  To see the reduction, suppose we are given a random pair g, g^μ as input. Then we perform 4 randomized reductions to obtain g_j, g_j^μ pairs, and generate a keypair honestly using these g_j values. We can compute a public key component E as well as the value E^μ needed to generate ζ_i. For the output of GRigEnc, use this value when generating the output of RigEnc. The distribution of this ciphertext is correct, as μ is random. When the adversary gives the challenge plaintext, compute $\mu' = \mathsf{H}(f(m_1^*, \ldots, m_n^*))$. If $g^{\mu'} = g^\mu$, then we have successfully computed the discrete log. In a group where the DDH assumption holds, this can only happen with negligible probability.

Similar to above, we may replace the $\mathsf{GRigExtract}(i, \cdot)$ oracle with an unbounded oracle that checks whether its input is in the support of $\mathsf{CTrans}(\zeta_i, \cdot)$, and if so outputs the transformation. Such an oracle is clearly independent of b, and by the above discussion, all of its responses match that of $\mathsf{GRigExtract}(i, \cdot)$, with overwhelming probability.

Lemma A.12. *Our construction is HCCA secure, if the DDH assumption holds in $\mathbb{G}$ and $\widehat{\mathbb{G}}$.*

Proof. First, by Lemma A.6, any adversary's advantage in the HCCA experiment changes negligibly when GenCiph is replaced by AltGenCiph to generate the challenge ciphertext. By Lemma A.5, this challenge ciphertext is distributed independently of b.

Next, we may replace all the oracles which use the private key with unbounded variants as described above. The entire set of responses from these oracles coincides with the original oracles, with overwhelming probability, so any adversary's advantage is only negligibly affected by this modification to the experiment. However, when these modified oracles are used, the adversary's entire view (public key, challenge ciphertext, and oracle responses) is independent of b. Thus the adversary has zero advantage in this modified HCCA experiment. It follows that the adversary's advantage in the original experiment is negligible. □

Lemma A.13. *Our construction is unlinkably homomorphic.*

Proof. Above, we argued that in the HCCA experiment, with overwhelming probability, the only ciphertexts accepted by the Dec in Phase I are those which are in the support of $\mathsf{Enc}_{PK}(\cdot)$. The unlinkability experiment is a restricted special case of Phase I of the HCCA experiment where there are no GRigEnc or GRigExtract oracles. So in the unlinkability experiment also, with overwhelming probability, only ciphertexts in the support of $\mathsf{Enc}_{PK}(\cdot)$ are accepted by the Dec. Further, observe that the CTrans procedure, when applied to ciphertexts in the support of $\mathsf{Enc}_{PK}(\mathsf{msg})$ and transformation T, outputs ciphertexts distributed identically to $\mathsf{Enc}_{PK}(T(\mathsf{msg}))$. Thus the adversary has zero advantage in the unlinkability experiment, conditioned on the overwhelming-probability event that the correctly-decrypting ciphertext it gives is in the support of $\mathsf{Enc}_{PK}(\cdot)$. The adversary's overall advantage in the unlinkability experiment is therefore negligible. □

References

[1] J. H. An, Y. Dodis, and T. Rabin. On the security of joint signature and encryption. In L. R. Knudsen, editor, *EUROCRYPT*, volume 2332 of *Lecture Notes in Computer Science*, pages 83–107. Springer, 2002.

[2] J. K. Andersen and E. W. Weisstein. Cunningham chain. From MathWorld–A Wolfram Web Resource. `http://mathworld.wolfram.com/CunninghamChain.html`, 2005.

[3] M. Backes, J. Müller-Quade, and D. Unruh. On the necessity of rewinding in secure multiparty computation. In S. P. Vadhan, editor, *TCC*, volume 4392 of *Lecture Notes in Computer Science*, pages 157–173. Springer, 2007.

[4] D. Beaver. Perfect privacy for two-party protocols. In J. Feigenbaum and M. Merritt, editors, *Proceedings of DIMACS Workshop on Distributed Computing and Cryptography*, volume 2, pages 65–77. American Mathematical Society, 1989.

[5] A. Beimel, T. Malkin, and S. Micali. The all-or-nothing nature of two-party secure computation. In M. J. Wiener, editor, *CRYPTO*, volume 1666 of *Lecture Notes in Computer Science*, pages 80–97. Springer, 1999.

[6] M. Belenkiy, M. Chase, M. Kohlweiss, and A. Lysyanskaya. P-signatures and noninteractive anonymous credentials. In R. Canetti, editor, *TCC*, volume 4948 of *Lecture Notes in Computer Science*, pages 356–374. Springer, 2008.

[7] M. Bellare, A. Boldyreva, A. Desai, and D. Pointcheval. Key-privacy in public-key encryption. In C. Boyd, editor, *ASIACRYPT*, volume 2248 of *Lecture Notes in Computer Science*, pages 566–582. Springer, 2001.

[8] M. Bellare and A. Sahai. Non-malleable encryption: Equivalence between two notions, and an indistinguishability-based characterization. In M. J. Wiener, editor, *CRYPTO*, volume 1666 of *Lecture Notes in Computer Science*, pages 519–536. Springer, 1999.

[9] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *STOC*, pages 1–10. ACM, 1988.

[10] J. Benaloh. *Verifiable Secret-Ballot Elections*. PhD thesis, Department of Computer Science, Yale University, 1987.

[11] M. Blaze, G. Bleumer, and M. Strauss. Divertible protocols and atomic proxy cryptography. In K. Nyberg, editor, *EUROCRYPT*, volume 1403 of *Lecture Notes in Computer Science*, pages 127–144. Springer, 1998.

[12] D. Boneh, E.-J. Goh, and K. Nissim. Evaluating 2-dnf formulas on ciphertexts. In J. Kilian, editor, *TCC*, volume 3378 of *Lecture Notes in Computer Science*, pages 325–341. Springer, 2005.

[13] A. Broadbent and A. Tapp. Information-theoretic security without an honest majority. In K. Kurosawa, editor, *ASIACRYPT*, volume 4833 of *Lecture Notes in Computer Science*, pages 410–426. Springer, 2007.

[14] C. Cachin. On the foundations of oblivious transfer. In K. Nyberg, editor, *EUROCRYPT*, volume 1403 of *Lecture Notes in Computer Science*, pages 361–374. Springer, 1998.

[15] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. Electronic Colloquium on Computational Complexity (ECCC) TR01-016, 2001. Previous version "A unified framework for analyzing security of protocols" availabe at the ECCC archive TR01-016. Extended abstract in FOCS 2001.

[16] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067, 2005. Revised version of [15].

[17] R. Canetti and M. Fischlin. Universally composable commitments. In J. Kilian, editor, *CRYPTO*, volume 2139 of *Lecture Notes in Computer Science*, pages 19–40. Springer, 2001.

[18] R. Canetti, S. Halevi, and J. Katz. Chosen-ciphertext security from identity-based encryption. In C. Cachin and J. Camenisch, editors, *EUROCRYPT*, volume 3027 of *Lecture Notes in Computer Science*, pages 207–222. Springer, 2004.

[19] R. Canetti and J. Herzog. Universally composable symbolic analysis of mutual authentication and key-exchange protocols. In S. Halevi and T. Rabin, editors, *TCC*, volume 3876 of *Lecture Notes in Computer Science*, pages 380–403. Springer, 2006.

[20] R. Canetti and S. Hohenberger. Chosen-ciphertext secure proxy re-encryption. In *ACM Computer and Communication Security (CCS)*, 2007.

[21] R. Canetti, H. Krawczyk, and J. B. Nielsen. Relaxing chosen-ciphertext security. In D. Boneh, editor, *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 565–582. Springer, 2003.

[22] R. Canetti, E. Kushilevitz, and Y. Lindell. On the limitations of universally composable two-party computation without set-up assumptions. *J. Cryptology*, 19(2):135–167, 2006.

[23] R. Canetti, Y. Lindell, R. Ostrovsky, and A. Sahai. Universally composable two-party and multi-party secure computation. In *STOC*, pages 494–503. ACM, 2002.

[24] D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 4(2), February 1981.

[25] D. Chaum, C. Crépeau, and I. Damgård. Multiparty unconditionally secure protocols (extended abstract). In *STOC*, pages 11–19. ACM, 1988.

[26] B. Chor, N. Gilboa, and M. Naor. Private information retrieval by keywords. TR CS0917, Department of Computer Science, Technion, 1997.

[27] B. Chor and Y. Ishai. On privacy and partition arguments. *Information and Computation*, 167(1):2–9, 2001.

[28] B. Chor and E. Kushilevitz. A zero-one law for boolean privacy. *SIAM J. Discrete Math.*, 4(1):36–47, 1991.

[29] R. Cramer, M. K. Franklin, B. Schoenmakers, and M. Yung. Multi-authority secret-ballot elections with linear work. In U. M. Maurer, editor, *EUROCRYPT*, volume 1070 of *Lecture Notes in Computer Science*, pages 72–83. Springer, 1996.

[30] R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In H. Krawczyk, editor, *CRYPTO*, volume 1462 of *Lecture Notes in Computer Science*, pages 13–25. Springer, 1998.

[31] I. Damgård, N. Fazio, and A. Nicolosi. Non-interactive zero-knowledge from homomorphic encryption. In S. Halevi and T. Rabin, editors, *TCC*, volume 3876 of *Lecture Notes in Computer Science*, pages 41–59. Springer, 2006.

[32] I. Damgård, J. Kilian, and L. Salvail. On the (im)possibility of basing oblivious transfer and bit commitment on weakened security assumptions. In J. Stern, editor, *EUROCRYPT*, volume 1592 of *Lecture Notes in Computer Science*, pages 56–73. Springer, 1999.

[33] I. Damgård and J. B. Nielsen. Universally composable efficient multiparty computation from threshold homomorphic encryption. In D. Boneh, editor, *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 247–264. Springer, 2003.

[34] G. Danezis. Breaking four mix-related schemes based on universal re-encryption. In S. K. Katsikas, J. Lopez, M. Backes, S. Gritzalis, and B. Preneel, editors, *ISC*, volume 4176 of *Lecture Notes in Computer Science*, pages 46–59. Springer, 2006.

[35] D. Dolev, C. Dwork, and M. Naor. Nonmalleable cryptography. *SIAM J. Comput.*, 30(2):391–437, 2000.

[36] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and D. Chaum, editors, *CRYPTO*, volume 196 of *Lecture Notes in Computer Science*, pages 10–18. Springer, 1985.

[37] Free Haven Project. Anonymity bibliography. `http://freehaven.net/anonbib/`, 2006.

[38] Y. Gertner, S. Kannan, T. Malkin, O. Reingold, and M. Viswanathan. The relationship between public key encryption and oblivious transfer. In *FOCS*, pages 325–335. IEEE, 2000.

[39] Y. Gertner, T. Malkin, and S. Myers. Towards a separation of semantic and cca security for public key encryption. In S. P. Vadhan, editor, *TCC*, volume 4392 of *Lecture Notes in Computer Science*, pages 434–455. Springer, 2007.

[40] O. Goldreich. *Foundations of Cryptography: Basic Tools*. Cambridge University Press, 2001. Earlier version available from `http://www.wisdom.weizmann.ac.il/~oded/frag.html`.

[41] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *STOC*, pages 218–229. ACM, 1987.

[42] S. Goldwasser and S. Micali. Probabilistic encryption. *J. CSS*, 28(2):270–299, Apr. 1984.

[43] P. Golle, M. Jakobsson, A. Juels, and P. Syverson. Universal re-encryption for mixnets. In T. Okamoto, editor, *CT-RSA*, volume 2964 of *Lecture Notes in Computer Science*, pages 163–178. Springer, 2004.

[44] J. Groth. A verifiable secret shuffle of homomorphic encryptions. In Y. Desmedt, editor, *Public Key Cryptography*, volume 2567 of *Lecture Notes in Computer Science*, pages 145–160. Springer, 2002.

[45] J. Groth. Rerandomizable and replayable adaptive chosen ciphertext attack secure cryptosystems. In M. Naor, editor, *TCC*, volume 2951 of *Lecture Notes in Computer Science*, pages 152–170. Springer, 2004.

[46] J. Groth and S. Lu. A non-interactive shuffle with pairing based verifiability. In K. Kurosawa, editor, *ASIACRYPT*, volume 4833 of *Lecture Notes in Computer Science*, pages 51–67. Springer, 2007.

[47] J. Groth and S. Lu. Verifiable shuffle of large size ciphertexts. In T. Okamoto and X. Wang, editors, *Public Key Cryptography*, volume 4450 of *Lecture Notes in Computer Science*, pages 377–392. Springer, 2007.

[48] I. Haitner. Semi-honest to malicious oblivious transfer - the black-box way. In R. Canetti, editor, *TCC*, volume 4948 of *Lecture Notes in Computer Science*, pages 412–426. Springer, 2008.

[49] D. Harnik, Y. Ishai, E. Kushilevitz, and J. B. Nielsen. OT-combiners via secure computation. In R. Canetti, editor, *TCC*, volume 4948 of *Lecture Notes in Computer Science*, pages 393–411. Springer, 2008.

[50] D. Harnik, M. Naor, O. Reingold, and A. Rosen. Completeness in two-party secure computation: A computational view. *J. Cryptology*, 19(4):521–552, 2006.

[51] J. Håstad, R. Impagliazzo, L. A. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999.

[52] M. Hirt and K. Sako. Efficient receipt-free voting based on homomorphic encryption. In *EUROCRYPT*, pages 539–556, 2000.

[53] D. Hofheinz, D. Unruh, and J. Mller-Quade. Polynomial runtime and composability. Cryptology ePrint Archive, Report 2009/023, 2009. `http://eprint.iacr.org/2009/023`.

[54] Y. Ishai, E. Kushilevitz, and R. Ostrovsky. Sufficient conditions for collision-resistant hashing. In J. Kilian, editor, *TCC*, volume 3378 of *Lecture Notes in Computer Science*, pages 445–456. Springer, 2005.

[55] Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai. Zero-knowledge from secure multiparty computation. In D. S. Johnson and U. Feige, editors, *STOC*, pages 21–30. ACM, 2007.

[56] Y. Ishai, M. Prabhakaran, and A. Sahai. Founding cryptography on oblivious transfer - efficiently. In D. Wagner, editor, *CRYPTO*, volume 5157 of *Lecture Notes in Computer Science*, pages 572–591. Springer, 2008.

[57] M. J. Jurik. *Extensions to the Paillier Cryptosystem with Applications to Cryptological Protocols*. PhD thesis, BRICS, 2003.

[58] Y. T. Kalai, Y. Lindell, and M. Prabhakaran. Concurrent composition of secure protocols in the timing model. *J. Cryptology*, 20(4):431–492, 2007.

[59] J. Katz, A. Sahai, and B. Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In N. P. Smart, editor, *EUROCRYPT*, volume 4965 of *Lecture Notes in Computer Science*, pages 146–162. Springer, 2008.

[60] D. Kidron and Y. Lindell. Impossibility results for universal composability in public-key models and with fixed inputs. Cryptology ePrint Archive, Report 2007/478, 2007. `http://eprint.iacr.org/2007/478`.

[61] J. Kilian. Founding cryptography on oblivious transfer. In *STOC*, pages 20–31. ACM, 1988.

[62] J. Kilian. A general completeness theorem for two-party games. In *STOC*, pages 553–560. ACM, 1991.

[63] J. Kilian. More general completeness theorems for secure two-party computation. In *STOC*, pages 316–324. ACM, 2000.

[64] J. Kilian, E. Kushilevitz, S. Micali, and R. Ostrovsky. Reducibility and completeness in private computations. *SIAM J. Comput.*, 29(4):1189–1208, 2000.

[65] D. Kraschewski and J. Müller-Quade. Completeness theorems with constructive proofs for symmetric, asymmetric and general 2-party-functions, 2008. Unpublished Manuscript, 2008. `http://iks.ira.uka.de/eiss/completeness`.

[66] R. Künzler, J. Müller-Quade, and D. Raub. Secure computability of functions in the IT setting with dishonest majority and applications to long-term security. In O. Reingold, editor, *TCC*, volume 5444 of *Lecture Notes in Computer Science*, pages 238–255. Springer, 2009.

[67] E. Kushilevitz. Privacy and communication complexity. In *FOCS*, pages 416–421. IEEE, 1989.

[68] E. Kushilevitz, S. Micali, and R. Ostrovsky. Reducibility and completeness in multi-party private computations. In *FOCS*, pages 478–489. IEEE, 1994.

[69] Y. Lindell. Bounded-concurrent secure two-party computation without setup assumptions. In *STOC*, pages 683–692. ACM, 2003.

[70] Y. Lindell. Lower bounds for concurrent self composition. In M. Naor, editor, *TCC*, volume 2951 of *Lecture Notes in Computer Science*, pages 203–222. Springer, 2004.

[71] P. D. MacKenzie, M. K. Reiter, and K. Yang. Alternatives to non-malleability: Definitions, constructions, and applications (extended abstract). In M. Naor, editor, *TCC*, volume 2951 of *Lecture Notes in Computer Science*, pages 171–190. Springer, 2004.

[72] F. J. MacWilliams and N. J. A. Sloane. *The Theory of Error-Correcting Codes*. North-Holland Mathematical Library. North Holland, January 1983.

[73] H. K. Maji, M. Prabhakaran, and M. Rosulek. Complexity of multi-party computation problems: The case of 2-party symmetric secure function evaluation. In O. Reingold, editor, *TCC*, volume 5444 of *Lecture Notes in Computer Science*, pages 256–273. Springer, 2009.

[74] H. K. Maji, M. Prabhakaran, and M. Rosulek. A zero-one law for deterministic 2-party secure computation. Unpublished manuscript, 2009. Draft available from author's website: `http://www.cs.uiuc.edu/homes/rosulek/`.

[75] M. Naor. Bit commitment using pseudorandomness. *J. Cryptology*, 4(2):151–158, 1991.

[76] M. Naor and B. Pinkas. Efficient oblivious transfer protocols. In *SODA*, pages 448–457, 2001.

[77] M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *STOC*, pages 427–437. ACM, 1990.

[78] M.-H. Nguyen and S. P. Vadhan. Zero knowledge with efficient provers. In J. M. Kleinberg, editor, *STOC*, pages 287–295. ACM, 2006.

[79] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In J. Stern, editor, *EUROCRYPT*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238. Springer, 1999.

[80] R. Pass and A. Rosen. Bounded-concurrent secure two-party computation in a constant number of rounds. In *FOCS*, pages 404–. IEEE Computer Society, 2003.

[81] A. Patil. On symbolic analysis of cryptographic protocols. Master's thesis, Massachusetts Institute of Technology, 2005.

[82] C. Peikert, V. Vaikuntanathan, and B. Waters. A framework for efficient and composable oblivious transfer. In D. Wagner, editor, *CRYPTO*, volume 5157 of *Lecture Notes in Computer Science*, pages 554–571. Springer, 2008.

[83] M. Prabhakaran and M. Rosulek. Rerandomizable RCCA encryption. In A. Menezes, editor, *CRYPTO*, volume 4622 of *Lecture Notes in Computer Science*, pages 517–584. Springer, 2007.

[84] M. Prabhakaran and M. Rosulek. Cryptographic complexity of multi-party computation problems: Classifications and separations. In D. Wagner, editor, *CRYPTO*, volume 5157 of *Lecture Notes in Computer Science*, pages 262–279. Springer, 2008.

[85] M. Prabhakaran and M. Rosulek. Homomorphic encryption with CCA security. In L. Aceto, I. Damgård, L. A. Goldberg, M. M. Halldórsson, A. Ingólfsdóttir, and I. Walukiewicz, editors, *ICALP (2)*, volume 5126 of *Lecture Notes in Computer Science*, pages 667–678. Springer, 2008.

[86] M. Prabhakaran and M. Rosulek. Towards robust computation on encrypted data. In J. Pieprzyk, editor, *ASIACRYPT*, volume 5350 of *Lecture Notes in Computer Science*, pages 216–233. Springer, 2008.

[87] M. Prabhakaran and A. Sahai. New notions of security: achieving universal composability without trusted setup. In L. Babai, editor, *STOC*, pages 242–251. ACM, 2004.

[88] M. Rabin. How to exchange secrets by oblivious transfer. Technical Report TR-81, Harvard Aiken Computation Laboratory, 1981.

[89] C. Rackoff and D. R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In J. Feigenbaum, editor, *CRYPTO*, volume 576 of *Lecture Notes in Computer Science*, pages 433–444. Springer, 1992.

[90] J. Rompel. One-way functions are necessary and sufficient for secure signatures. In *STOC*, pages 387–394. ACM, 1990.

[91] K. Sako and J. Kilian. Secure voting using partially compatible homomorphisms. In Y. Desmedt, editor, *CRYPTO*, volume 839 of *Lecture Notes in Computer Science*, pages 411–424. Springer, 1994.

[92] T. Sander, A. Young, and M. Yung. Non-interactive cryptocomputing for NC^1. In *FOCS*, pages 554–567. IEEE, 1999.

[93] V. Shoup. A proposal for an ISO standard for public key encryption. Cryptology ePrint Archive, Report 2001/112, 2001. `http://eprint.iacr.org/2001/112`.

[94] D. X. Song, D. Wagner, and A. Perrig. Practical techniques for searches on encrypted data. In *IEEE Symposium on Security and Privacy*, pages 44–55, 2000.

[95] D. Wikström. A note on the malleability of the El Gamal cryptosystem. In A. Menezes and P. Sarkar, editors, *INDOCRYPT*, volume 2551 of *Lecture Notes in Computer Science*, pages 176–184. Springer, 2002.

[96] A. C. Yao. Protocols for secure computations (extended abstract). In *FOCS*, pages 160–164. IEEE, 1982.

[97] A. C. Yao. How to generate and exchange secrets. In *FOCS*, pages 162–167. IEEE, 1986.

Index

Author's Biography

Mike Rosulek was born in Minden, Iowa, on July 17, 1981, and grew up in Fredericksburg, Iowa. He received a B.S. in Computer Science, with distinction, from Iowa State University in May 2003. He will complete his Ph.D. in Computer Science at the University of Illinois at Urbana-Champaign, in June 2009. After graduation, he will join the faculty of the Computer Science Department at the University of Montana in Missoula, Montana.