A Zero-One Law for Deterministic 2-Party Secure Computation*

Hemanta K. Maji† Manoj Prabhakaran† Mike Rosulek‡

November 5, 2009

Abstract

We use security in the Universal Composition framework as a means to study the "cryptographic complexity" of 2-party secure computation tasks (functionalities). We say that a functionality $\mathcal{F}$ *reduces to* another functionality $\mathcal{G}$ if there is a UC-secure protocol for $\mathcal{F}$ using ideal access to $\mathcal{G}$. This reduction is a natural and fine-grained way to compare the relative complexities of cryptographic tasks. There are two natural "extremes" of complexity under the reduction: the *trivial* functionalities, which can be reduced to any other functionality; and the *complete* functionalities, to which any other functionality can be reduced.

In this work we show that under a natural computational assumption (the existence of a protocol for oblivious transfer secure against semi-honest adversaries), there is a **zero-one law** for the cryptographic complexity of 2-party deterministic functionalities. Namely, *every such functionality is either trivial or complete.* No other qualitative distinctions exist among functionalities, under this computational assumption.

While nearly all previous work classifying multi-party computation functionalities has been restricted to the case of secure function evaluation, our results are the first to consider completeness of arbitrary *reactive* functionalities, which receive input and give output repeatedly throughout several rounds of interaction. One important technical contribution in this work is to initiate the comprehensive study of the cryptographic properties of reactive functionalities. We model these functionalities as finite automata and develop an automata-theoretic methodology for classifying and studying their cryptographic properties. Consequently, we completely characterize the reactive behaviors that lead to cryptographic non-triviality.

*Work supported by NSF grants CNS 07-16626 and CNS 07-47027.

†Department of Computer Science, University of Illinois, Urbana-Champaign. {hmaji2,mmp}@uiuc.edu.

‡Department of Computer Science, University of Montana. mikero@cs.umt.edu.

1 Introduction

Secure multi-party computation (MPC) is one of the most surprising computational phenomena known. In fact, the paradigm encompasses not one, but a wide range of phenomena, depending on the MPC task (functionality) in question. One may ask whether, within this vast range of tasks, some tasks have more cryptographic sophistication than others, in terms of how easily they can be securely realized.

In a highly influential piece of work, Goldreich, Micali, and Wigderson [GMW87] showed that under a natural computational assumption, there is *no qualitative distinction* among the MPC functionalities: they are all securely realizable. However, in another influential work, Canetti [C01] showed that under a more demanding but more realistic model of security, *at least one qualitative distinction exists* among MPC functionalities, regardless of any computational assumption: the separation between "trivial" and "non-trivial" functionalities. In this paper we show that, under the same intractability assumption needed for the results in [GMW87], the distinction between trivial and non-trivial functionalities is *the only qualitative distinction among deterministic 2-party functionalities* in Canetti's stronger security framework for MPC.

More formally, we use a natural complexity-theoretic *reduction* to compare the qualitative "cryptographic complexities" of MPC functionalities. We say that a functionality $\mathcal{F}$ reduces to $\mathcal{G}$ (written $\mathcal{F} \sqsubseteq_{\text{PPT}} \mathcal{G}$) if there is a secure protocol for $\mathcal{F}$ that uses ideal access to $\mathcal{G}$. We use the strong definition of security from the the framework of Universal Composability (UC) [C01]. Under this reduction, there are two natural extremes of cryptographic complexity: we call a functionality "trivial" if it can be reduced to every other functionality and "complete" if every functionality can be reduced to it. Stated in these terms, our main result is the following:

> *The following two statements are equivalent:*
>
> **Zero-One Law:** *Every deterministic, finite 2-party functionality is either trivial or complete.*
>
> sh-OT **Assumption:** *There exists a 2-party protocol that securely realizes the oblivious transfer functionality against semi-honest (a.k.a., passive, honest-but-curious) PPT adversaries.*

The zero-one law applies not just to secure function evaluation functionalities, but also to **reactive** ones that receive input and give output repeatedly over many rounds of interaction, maintaining secret state between rounds. To the best of our knowledge, ours is the first work that considers how to *use* arbitrary reactive functionalities for other cryptographic purposes. In comparison, previous works like [GMW87, CLOS02] only give protocols to securely *realize* arbitrary reactive functionalities; other works exclusively considered non-reactive functionalities, or else considered only specific reactive functionalities, like commitment.

Behavioral Components of Functionalities. To establish the zero-one law, we advance on two technical fronts in the study of complexity of secure multi-party computation. The first front focuses on understanding distinct *non-trivial behavioral components* of (possibly reactive) functionalities. We identify a list of four qualitatively distinct such components. For each one we can associate a familiar "canonical" functionality which is non-trivial for only that reason:

- Allowing simultaneous exchange of information, exemplified by the boolean XOR functionality $\mathcal{F}_{\text{XOR}}$.
- Selectively hiding one party's inputs from the other, exemplified by a simple SFE functionality we introduce called *simple cut-and-choose*, $\mathcal{F}_{\text{CC}}$.
- Selectively hiding both party's inputs simultaneously, exemplified by the oblivious transfer functionality $\mathcal{F}_{\text{OT}}$.
- Holding meaningful hidden information in internal memory between rounds, exemplified by the commitment functionality $\mathcal{F}_{\text{COM}}$. (This component can appear only in a reactive functionality.)

Formally defining each of these components, especially the last one, requires us to develop new automata-theoretic tools for reasoning about the behavior of reactive functionalities. A more detailed overview of these techniques is given in Section 2.

We show that these four fundamental behaviors are in fact an *exhaustive* characterization of non-triviality: in Theorems 1 and 4, we show that a reactive functionality $\mathcal{G}$ is non-trivial if and only if $\mathcal{F} \sqsubseteq \mathcal{G}$ *unconditionally* for some $\mathcal{F} \in \{\mathcal{F}_{\text{XOR}}, \mathcal{F}_{\text{CC}}, \mathcal{F}_{\text{OT}}, \mathcal{F}_{\text{COM}}\}$.[1] In other words, every non-trivial functionality must include at least one of the above four components above. Since our definitions of the four fundamental non-trivial behaviors are all combinatorial, we are able to give the first complete *combinatorial* characterization of non-triviality (consequently, completeness) for *reactive* functionalities.

Demonstrating Completeness. Our second technical front focuses on building protocols under minimal computational intractability assumptions. To establish the zero-one law, it suffices to show that each of the four canonical non-trivial functionalities is complete. Two of these, oblivious transfer and commitment, are already known to be complete under the reduction we consider. We show that $\mathcal{F}_{\text{XOR}}$ and $\mathcal{F}_{\text{CC}}$ are also complete under the minimal sh-OT assumption. The techniques used in these new protocols are summarized in more detail in Section 2.

It is instructive to compare our new protocol constructions to that of [CLOS02], which has been the closest to an analogue of the [GMW87] result for the stronger security definition in [C01]. While [CLOS02] only shows the completeness of one specific functionality, we show that *every* deterministic non-trivial functionality is complete (under sh-OT assumption). Even for the complete functionality considered in [CLOS02] (coin-tossing), we improve over their protocol, because the computational intractability assumption used there is not known to follow from the sh-OT assumption. However, the protocol in [CLOS02] provides security against *adaptive* corruption, whereas we do not know whether the zero-one law extends to that setting (even under a stronger intractability assumption).

A Framework for Computational Intractability Assumptions. An important contribution of this work is that, in concert with subsequent results in a companion paper [MPR10], it forms the foundations for a *framework to study the computational intractability assumptions necessary for cryptography* by relating them to secure multi-party computation functionalities. While our results in this work focus on the *sufficiency* of various assumptions for reductions of the form $\mathcal{F} \sqsubseteq_{\text{PPT}} \mathcal{G}$, the complementary results of [MPR10] classify the *necessity* of various assumptions for such reductions.

Since several reductions of the form $\mathcal{F} \sqsubseteq_{\text{PPT}} \mathcal{G}$ turn out to be exactly *equivalent* to well-known computational assumptions, this motivates an approach of *defining* computational assumptions in terms of such reductions. Such assumptions must be of a fundamental nature for MPC, since they are not introduced to prove security of protocols, but are derived directly from the definitions of MPC functionalities themselves.

While our results in this work imply that the sh-OT assumption is the *maximal* assumption that emerges in this framework, we conjecture that the existence of one-way functions is the *minimal* assumption emerging in the framework. A more intriguing question is whether there are other intermediate assumptions. We conjecture there are, but for a broad class of reductions considered in [MPR10], all such assumptions are equivalent to one of the two mentioned above. Put differently, one likely outcome of this line of investigation is to discover new cryptographically interesting worlds in "Impagliazzo's multiverse" [I95] between Cryptomania (which we interpret as a world where the sh-OT assumption is true) and Minicrypt (where only one-way functions exist), or to show there are none. The classification and protocols here and the lower bounds in [MPR10] form the basic results in such an investigation.

Related Work. In Appendix A we briefly survey some of the important works relevant to our study of 2-party functionalities. Here we focus on describing important differences between previous work and our own. The bulk of the work on complexity of 2-party functionalities considers the computationally unbounded setting [K88, CK89, K89B, B89, K91, K00, KKMO00, KMQR09, MPR09].

[1] Indeed just $\mathcal{F}_{\text{XOR}}$ and $\mathcal{F}_{\text{CC}}$ by themselves are an exhaustive characterization of non-triviality, as they can both be unconditionally obtained from $\mathcal{F}_{\text{OT}}$ and $\mathcal{F}_{\text{COM}}$. However, we include all four functionalities in our list of fundamental behavioral components because completeness is established differently for each of them.

In the computationally bounded setting, there have been fewer relevant results: For the special case of SFE functionalities in which only one party receives the output, Beimel et al. [BMM99] showed that the sh-OT assumption is implied by the existence of a *semi-honest* secure protocol for any functionality that is not unconditionally trivial. [HNRR06] partially extends this result beyond finite functionalities, but is still restricted to the case of asymmetric-output.[2]

The above results were in the standalone setting, and are not true for the setting with an arbitrary environment. Since [C01] introduced the Universal Composition framework to formalize security in arbitrary environments, there have been several works on cryptographic complexity of functionalities in this setting. In particular, [C01, CF01, CKL03, PR08] characterize trivial functionalities. On the other hand, [CLOS02] showed that the "coin-tossing" functionality is complete, assuming the existence of enhanced trapdoor permutations and dense cryptosystems. [DNO09] independently show the completeness of the coin-tossing functionality under the minimal assumption, as we do. Their construction is similar in spirit to our protocol for the same task, though more complicated due to the use of an intermediate "public-key infrastructure" functionality. In fact our current protocol is the result of a simplification to a protocol in an earlier draft of this work, motivated by the result of [DNO09].

We stress that virtually all the above mentioned prior results are either restricted to characterizing *triviality* (not completeness), characterizing only non-reactive functionalities, or considering only specific and not arbitrary reactive functionalities. Ours is the first characterization of completeness for arbitrary reactive functionalities.

Finally, in subsequent work we use the results in this paper, in conjunction with new lower bounds, to initiate a foundational study of cryptographically relevant intractability assumptions [MPR10] using secure multi-party functionalities.

2 Overview of Our Techniques

In proving our main result, the more interesting direction is to show that sh-OT assumption implies the zero-one law. That is, we must construct protocols to demonstrate the completeness of every non-trivial functionality, using only the sh-OT assumption. We do this in a series of steps, outlined in Figure 1.

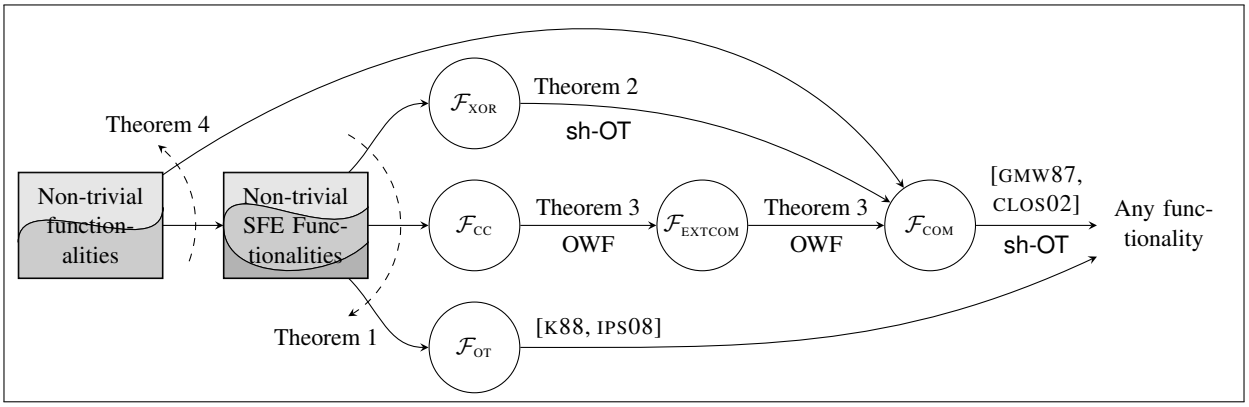

Figure 1: Overview of protocol constructions used in proving main result. An arrow from functionality $\mathcal{F}$ to $\mathcal{G}$ denotes a secure protocol for $\mathcal{G}$ using ideal access to $\mathcal{F}$. Arrows not labeled by a computational assumption indicate unconditionally secure protocols.

The Non-Reactive Case. In Section 4, we first prove the main result for the special case of non-reactive (also known as secure function evaluation, or SFE) functionalities, which simply evaluate a function on the two parties' inputs and then stop responding.

[2] In Section 7 we show that, as in [HNRR06], there is a gap between triviality and completeness when our results are extended to unbounded-memory functionalities.

As described in the introduction, we identify $\mathcal{F}_{\text{XOR}}$, $\mathcal{F}_{\text{OT}}$, and $\mathcal{F}_{\text{CC}}$ as the canonical components of non-triviality for SFE functionalities. $\mathcal{F}_{\text{CC}}$ is defined as the secure evaluation of the symmetric-output function whose function table is $\boxed{\begin{smallmatrix}0 & 2\\ 1 & 2\end{smallmatrix}}$; note that Bob (whose input corresponds to the choice of columns) can choose whether to learn Alice's input (the choice of rows), and Alice learns Bob's choice. Thus $\mathcal{F}_{\text{CC}}$ exemplifies a selective hiding of information about just one party's inputs.

In Theorem 1 we show that every non-trivial SFE functionality $\mathcal{F}$ must satisfy either $\mathcal{F}_{\text{XOR}} \sqsubseteq_{\text{STAT}} \mathcal{F}$, $\mathcal{F}_{\text{OT}} \sqsubseteq_{\text{STAT}} \mathcal{F}$, or $\mathcal{F}_{\text{CC}} \sqsubseteq_{\text{STAT}} \mathcal{F}$. Our analysis involves a characterization of important 2×2 minors in the function table of $\mathcal{F}$. For each of $\mathcal{F}_{\text{XOR}}$, $\mathcal{F}_{\text{OT}}$, and $\mathcal{F}_{\text{CC}}$, we introduce a general form (in which parties may receive different outputs) of a 2×2 minor which succinctly captures the general behavior that causes each functionality to be non-trivial.

To establish the zero-one law for this special case, we must therefore show that each of these three canonical functionalities is complete. It is well-known that $\mathcal{F}_{\text{OT}}$ is (unconditionally) complete, even under the strong notion of reduction that we consider [K88, IPS08]. For the other two cases, we use the fact that the commitment functionality $\mathcal{F}_{\text{COM}}$ is complete the UC framework under the sh-OT assumption. This follows from the well-known CLOS result [CLOS02]. Thus, to complete our claim it suffices to show that $\mathcal{F}_{\text{COM}} \sqsubseteq_{\text{PPT}} \mathcal{F}_{\text{CC}}$ and $\mathcal{F}_{\text{COM}} \sqsubseteq_{\text{PPT}} \mathcal{F}_{\text{XOR}}$.

We give a new commitment protocol in the $\mathcal{F}_{\text{XOR}}$-hybrid model (Theorem 2). We note that [CLOS02] show (implicitly) that $\mathcal{F}_{\text{XOR}}$ is complete;[3] however, their protocol focuses on achieving adaptive security and, as such, depends on a hardness assumption that is not known to be implied by sh-OT assumption. Our new protocol achieves static security using a novel non-black-box usage of the minimal sh-OT assumption.

We also give a new commitment protocol in the $\mathcal{F}_{\text{CC}}$-hybrid model (Theorem 3). $\mathcal{F}_{\text{CC}}$ is a less sophisticated functionality than $\mathcal{F}_{\text{XOR}}$; consequently, our protocol using $\mathcal{F}_{\text{CC}}$ is more involved. We first define an intermediate commitment functionality called $\mathcal{F}_{\text{EXTCOM}}$ which captures the requirements of a standalone-secure commitment protocol with a *straight-line* extracting simulator, and we show that this intermediate functionality can be securely realized using $\mathcal{F}_{\text{CC}}$. We then use a technique similar to the $\binom{2}{1}$ commitments of [NV06] to show that $\mathcal{F}_{\text{EXTCOM}}$ can be used to obtain a full-fledged $\mathcal{F}_{\text{COM}}$ protocol. Interestingly, both of these protocols require only the existence of one-way functions.

Dealing with reactive functionalities. Our next main technical contribution is to develop tools for classifying and reasoning about arbitrary *reactive* functionalities. We model reactive functionalities as finite-state automata, and initiate an automata-theoretic analysis of their behaviors. More specifically, we identify the states of an automaton which are cryptographically non-trivial, a notion that we define in terms of secure protocols but that also has a purely combinatorial characterization.

Given the appropriate automata-theoretic definitions, we are then able to show that a reactive functionality can only be non-trivial for two reasons:

- having input/output behavior (in a single round) like that of a non-trivial SFE, or
- keeping relevant information about a party's inputs "hidden" in its memory between rounds

More formally, we show in Theorem 4 that every non-trivial reactive functionality can either be used to obtain some non-trivial SFE (i.e., the first case above), or can be used to achieve the commitment functionality $\mathcal{F}_{\text{COM}}$ (the canonical functionality which exemplifies the second case above). In this way, we reduce the zero-one law for reactive functionalities to the zero-one law for SFE functionalities, since $\mathcal{F}_{\text{COM}}$ is complete. We note that formally defining non-trivial usage of internal memory by a reactive functionality is very challenging, and comprises the bulk of our contributions for reactive functionalities. However, as a result of our new analysis, we obtain the first complete combinatorial characterization for triviality or completeness of arbitrary reactive functionalities.

[3] They show that the coin-tossing functionality, for which there is an elementary protocol using $\mathcal{F}_{\text{XOR}}$, is complete.

3 Preliminaries

Model and Security Definition. Our security definitions are grounded in the framework of Universal Composable (UC) security [C01], with which we assume the reader has slight familiarity. For concreteness we consider the model used in [PR08], which in turn is based on that in [P05]. However, we emphasize that very few specifics of the model (including ideal functionalities, an interactive environment and simulation based security) are important for the results.

The UC model allows a large class of MPC functionalities, not all of which are "natural." For instance, a functionality that announces the identities of the corrupt parties is not natural; a reactive functionality which introduces a race condition depending on the order in which it receives inputs from parties is also not natural. Following the convention in all previous works (to the best of our knowledge), we do not consider such functionalities. We formally define the exact class of functionalities considered in this work in Appendices B and E. We note that the functionalities in this class do not offer a guarantee of output *fairness*; that is, they allow the adversary to control the delivery of outputs.

We write $\mathcal{F} \sqsubseteq \mathcal{G}$ if there is a protocol that securely realizes $\mathcal{F}$ in the "$\mathcal{G}$-hybrid model;" see [C01] or [P05] for a formal definition. In the $\mathcal{G}$-hybrid model, the parties in the protocol can interact with any number of (asynchronous) copies of $\mathcal{G}$, and can access $\mathcal{G}$ in any "role". This second convention is crucial to our results (see Section 7). We consider only efficient protocols, but make a notational distinction between unconditionally (statistically) secure protocols (denoted by $\sqsubseteq_{\text{STAT}}$) and protocols whose security depends on a computational assumption (denoted by $\sqsubseteq_{\text{PPT}}$). As is standard, we require security against active (i.e., malicious) adversaries. However, as we point out in Section 7, our results extend to a stronger definition where security is required against both active and semi-honest adversaries.[4]

By default, we also allow protocols access to a communication channel. Following [PR08], we consider the natural model of a private communication channel, in which parties can send fixed-length messages (with the adversary controlling delivery). The choice of public vs. private channel is not crucial to our results (see Section 7).

All results in this work are restricted to static corruption (where the adversary has to corrupt any parties before the protocol begins). In fact, we leave open the possibility that our main theorem breaks down in the case of adaptive corruption.

The sh-OT assumption. The primary computational assumption we consider is the existence of a protocol for $\mathcal{F}_{\text{OT}}$ secure against semi-honest, PPT adversaries (sh-OT assumption, for short). It is possible to express this assumption using the definition of UC security restricted to semi-honest adversaries (in both the real and the ideal executions). However, we point out that the traditional (standalone) security definition is equivalent to the UC security definition, since the simulation required by semi-honest security does not, and need not, *extract* the inputs of the corrupt players; it simply uses the input given by the environment.

Some of our protocol constructions additionally rely on statistically binding (standalone secure) commitment schemes, pseudorandom generators, (standalone secure) witness-indistinguishable proofs or zero-knowledge proofs of knowledge for NP. All of these primitives have well-known constructions assuming the existence of one-way functions [N91, HILL99, G01]. One-way functions are in turn implied by the sh-OT assumption [IL89].

4 Zero-One Law for Non-Reactive Functionalities

In this section we prove the zero-one law for the special case of *non-reactive* finite functionalities, also known as *secure function evaluation* (SFE) functionalities. An SFE functionality is parameterized by a pair of functions (f_A, f_B) with input domains $X \times Y$, where X and Y are finite sets. The functionality waits to receive input $x \in X$ from Alice and $y \in Y$ from Bob, then outputs $f_A(x, y)$ to Alice and $f_B(x, y)$ to

[4] Note that when considering semi-honest adversaries, the simulator must also be semi-honest.

Bob. In Appendix B we give a complete formal definition of SFE functionalities, which explicitly models adversarial control over output delivery, as is standard in the UC framework.

Three "Canonical" Non-Reactive Functionalities. Our main characterization in this section demonstrates that an SFE functionality can be non-trivial for only one of three simple reasons, as outlined in Section 2. The following three SFE functionalities exemplify the three different cases, respectively:

$\mathcal{F}_{\text{XOR}}$ (exclusive-or): Alice gives input $x \in \{0,1\}$ and Bob gives input $y \in \{0,1\}$. Both parties receive output $x \oplus y$.

$\mathcal{F}_{\text{CC}}$ (simple cut-and-choose): Alice gives input $x \in \{0,1\}$ and Bob gives input $y \in \{0,1\}$. If $y = 0$, then both parties receive output x. If $y = 1$, then both parties receive output 2. Intuitively, Bob decides whether to learn Alice's bit, and Alice learns Bob's choice.

$\mathcal{F}_{\text{OT}}$ (oblivious transfer): Alice gives inputs $x_0, x_1 \in \{0,1\}$ and Bob gives input $y \in \{0,1\}$. Bob receives output x_y and Alice receives no output.

We show that these three fundamental properties exhaustively characterize non-triviality, as follows:

Theorem 1. *Let $\mathcal{F}$ be an SFE functionality. Then $\mathcal{F}$ is non-trivial if and only if $\mathcal{F}_{\text{XOR}} \sqsubseteq_{\text{STAT}} \mathcal{F}$ or $\mathcal{F}_{\text{CC}} \sqsubseteq_{\text{STAT}} \mathcal{F}$ or $\mathcal{F}_{\text{OT}} \sqsubseteq_{\text{STAT}} \mathcal{F}$.*

Proof Sketch. One direction of the proof follows directly from the characterization of trivial SFE functionalities from [PR08]. Each of $\mathcal{F}_{\text{XOR}}$, $\mathcal{F}_{\text{CC}}$, and $\mathcal{F}_{\text{OT}}$ is unconditionally non-trivial.

We give an overview of the proof of the other direction, deferring the full details to Appendix B. Kraschewski and Müller-Quade [KMQ08] identify a 2×2 minor within the function table of an SFE, which generalizes the (symmetric-output) boolean OR functionality $\boxed{\begin{smallmatrix}0&1\\1&1\end{smallmatrix}}$ that is known to be complete. They show that an SFE with such a minor can be used to construct an unconditionally UC-secure protocol for $\mathcal{F}_{\text{OT}}$.

Similarly, we also identify another important 2×2 minor called a *generalized-*CC *minor*. Intuitively, a minor is a generalized-CC minor if one party can choose whether to learn the other's input, and this choice is made public in the function's output. We show that if $\mathcal{F}$ has such a minor, then the protocol in which the parties simply restrict their inputs to that minor while accessing $\mathcal{F}$ is a UC-secure protocol for $\mathcal{F}_{\text{CC}}$.[5]

Finally, if $\mathcal{F}$ does not have either kind of 2×2 minor mentioned above, then we show that $\mathcal{F}$ must simply be (equivalent to) a function that takes inputs $x \in X$ from Alice and $y \in Y$ from Bob, then outputs (x, y) to both parties. If $\max\{|X|, |Y|\} \geq 2$, then there is an elementary UC-secure protocol for $\mathcal{F}_{\text{XOR}}$ in the $\mathcal{F}$-hybrid model. Otherwise, $\mathcal{F}$ is trivial: the protocol in which one party simply sends their input to the other party is a UC-secure (plain) protocol for $\mathcal{F}$. □

Completeness of the Three Canonical Non-Reactive Functionalities. Since $\mathcal{F}_{\text{OT}}$ is unconditionally complete (even with respect to UC secure protocols) [K88, IPS08], and the commitment functionality $\mathcal{F}_{\text{COM}}$ is complete under the sh-OT assumption [CLOS02], it suffices to prove the following two theorems:

Theorem 2. *If the* sh-OT *assumption is true, then $\mathcal{F}_{\text{COM}} \sqsubseteq_{\text{PPT}} \mathcal{F}_{\text{XOR}}$.*

Proof Sketch. We first observe that the coin-tossing functionality $\mathcal{F}_{\text{COIN}}$ [6] has an elementary, unconditionally secure protocol in the $\mathcal{F}_{\text{XOR}}$-hybrid model. Thus it suffices to show that $\mathcal{F}_{\text{COM}} \sqsubseteq_{\text{PPT}} \mathcal{F}_{\text{COIN}}$. The well-known CLOS result [CLOS02] proves exactly this; however, their focus was on achieving *adaptive* security, and their protocol relied on a stronger computational assumption than the sh-OT assumption. Thus we must use an entirely different approach for achieving $\mathcal{F}_{\text{COM}}$ (with static security) from $\mathcal{F}_{\text{COIN}}$. We give an overview of our protocol, whose complete description and security proof are given in Appendix C.

[5] Note that, in general, restricting inputs to a minor of $\mathcal{F}$ does not give a secure protocol (against malicious adversaries) for the SFE corresponding to that minor, since a malicious adversary may send inputs to $\mathcal{F}$ outside of the prescribed minor.

[6] $\mathcal{F}_{\text{COIN}}$ is a functionality which, upon activation, samples an unbiased coin $b \leftarrow \{0,1\}$ and outputs it to both parties.

Suppose ψ_{sh} is the semi-honest protocol for $\mathcal{F}_{\text{OT}}$ guaranteed by the sh-OT assumption. We suppose that the sender in ψ_{sh} provides two bits (x_0, x_1), the receiver provides a bit y, and the receiver learns x_y.

Our commitment protocol is as follows, when Alice is committing to $b \in \{0, 1\}$. First, both parties use $\mathcal{F}_{\text{COIN}}$ to generate a sequence of random coins σ. The sender Alice and receiver Bob interact in an instance of ψ_{sh}, with Alice using inputs $(x_0 = 0, x_1 = b)$, and Bob using input $y = 0$. To ensure that both parties provide inputs of the required form, we "compile" the ψ_{sh} subprotocol using a variant of the standard GMW compiler [GMW87]. Unlike the GMW compiler, at each step we make the parties give a witness-indistinguishable proof that *either* they are following the protocol honestly with the appropriate inputs, *or* the public coins σ are from a pseudorandom distribution. In the reveal phase, Alice gives a witness-indistinguishable proof that *either* σ was from a pseudorandom distribution, *or* all her messages in the ψ_{sh} subprotocol were consistent with her having input $x_1 = b$.

Intuitively, in the real interaction (where σ is generated honestly using $\mathcal{F}_{\text{COIN}}$), the GMW compilation ensures that both parties are executing the ψ_{sh} subprotocol honestly and appropriately. Thus, Bob learns nothing about b in the commit phase, and Alice can only open the commitment to the value of b that she used in the commit phase.

However, when the simulator is corrupt it can choose σ from a pseudorandom distribution. If Alice is corrupt, the simulator can act as Bob using input $y = 1$ to the ψ_{sh} subprotocol, while still giving convincing GMW proofs. By the correctness and security of the ψ_{sh} protocol, the simulator correctly extracts b in the commit phase, and the simulation is indistinguishable from the real interaction.

If Bob is corrupt, the simulator can give a commitment to 0 in the commit phase, but open it to any value in the reveal phase (using the clause in the witness-indistinguishable proof related to σ). Thus the simulator can successfully equivocate to a corrupt Bob.

To show that both of these simulations are sound, we must apply the semi-honest security of ψ_{sh}, which is the most delicate part of the proof, since the simulator exists in the UC setting. We construct a sequence of hybrid interactions between the real and ideal UC (straight-line) interactions, and show that if any adversary can distinguish between certain hybrids, then we can construct a corresponding adversary (possibly using rewinding) which violates the semi-honest security properties of ψ_{sh}. □

Theorem 3. *If one-way functions exist, then* $\mathcal{F}_{\text{COM}} \sqsubseteq_{\text{PPT}} \mathcal{F}_{\text{CC}}$.

Proof Sketch. In many respects, $\mathcal{F}_{\text{CC}}$ is a less cryptographically sophisticated functionality than $\mathcal{F}_{\text{XOR}}$ (we provide some evidence for this fact in Section 7). Consequently, our protocol for obtaining $\mathcal{F}_{\text{COM}}$ from $\mathcal{F}_{\text{CC}}$ is more complicated than the one using $\mathcal{F}_{\text{XOR}}$. On the other hand, our protocol in the $\mathcal{F}_{\text{CC}}$-hybrid model uses only the assumption of one-way functions, which is weaker than the sh-OT assumption.

The simulator for a UC-secure commitment protocol has two main tasks: (1) to extract the committed value from a corrupt sender during the commit phase, and (2) to give an equivocal commitment to a corrupt receiver that can be convincingly opened to any value during the reveal phase. Our construction of a UC-secure commitment protocol is broken into two major conceptual steps, which tackle these two properties in a somewhat modular fashion.

We first define an intermediate "extractable commitment" functionality called $\mathcal{F}_{\text{EXTCOM}}$. The complete formulation of $\mathcal{F}_{\text{EXTCOM}}$ is highly non-trivial, and is contained in Appendix D.1. $\mathcal{F}_{\text{EXTCOM}}$ succinctly expresses the requirements of a *statistically binding, computationally hiding* commitment scheme (in the traditional standalone-secure sense) which also admits a *straight-line extracting* simulator. We believe that this method of expressing a combination of standalone and universally composable security properties may be of independent interest. Using a technique similar in spirit to the $\binom{2}{1}$-commitments of Nguyen and Vadhan [NV06], we show that if one-way functions exist, then $\mathcal{F}_{\text{COM}} \sqsubseteq_{\text{PPT}} \mathcal{F}_{\text{EXTCOM}}$ (Lemma 4).

Thus it suffices to construct a commitment protocol which has a UC extraction property, but only a standalone-secure hiding property. This commitment protocol is as follows. To commit to a bit b, Alice

first chooses a random bitstring s and then applies a good linear error-correcting code to obtain a codeword t. She commits to t using a statistically binding (standalone-secure) commitment protocol. For each bit t_i of t, Alice gives t_i as input to $\mathcal{F}_{\text{CC}}$, and Bob chooses to learn it with some probability. Recall that in $\mathcal{F}_{\text{CC}}$, Alice learns which bits Bob chose to see. Alice ensures that Bob only learned sufficiently few bits of t so that some uncertainty about s remains. This remaining uncertainty can be deterministically extracted (as a linear function of s), and Alice uses it as a one-time pad to mask b. She sends the masked b to Bob to complete the commitment phase. In the reveal phase, Alice opens the commitment to t, and Bob checks for consistency with the bits that he learned in the commit phase. The full details and security proof are provided in Appendix D.2.

Intuitively, the protocol is computationally hiding and statistically binding because the deterministic extraction of the mask is perfect (using a simple linear function). The only information about the mask is given in a statistically-binding standalone-secure commitment to t.

However, the simulator provides the interface for $\mathcal{F}_{\text{CC}}$ to a corrupt Alice. Consequently, the simulator can see all of Alice's inputs to $\mathcal{F}_{\text{CC}}$, which are the (purported) bits of t. Because Bob has a certain probability of revealing each one of the bits of t and he verifies them against Alice's statistically binding commitment to t, we argue that Alice cannot supply too many incorrect values of to $\mathcal{F}_{\text{CC}}$. In particular, Alice cannot give more incorrect bits than can be corrected by the error correcting code, except with negligible probability. Thus the simulator can perform a noisy decoding to obtain s and then easily extract b. □

5 Classifying Reactive Functionalities

We model reactive functionalities as finite automata. Each state transition is labeled by a tuple in $X \times Y \times Z \times Z$, where X, Y, and Z are finite sets. We require the automaton to be deterministic; that is, for every state q and every $(x, y) \in X \times Y$, there is at most one transition leaving q whose label begins with (x, y). A transition from q to q' with label (x, y, s, t) means that upon receiving input x from Alice and y from Bob in state q, the functionality will deliver output s to Alice and t to Bob, and change to state q'. The formal definition is given in Definition 8, and it explicitly models adversarial control over output delivery.

As outlined in Section 2, we show that a reactive functionality can be non-trivial only for two simple reasons: (1) behaving like a non-trivial SFE functionality during a single round, or (2) using its internal memory in a non-trivial way. Formally defining this second condition requires a careful new automata-theoretic analysis of reactive behaviors. Intuitively, memory is used in a non-trivial way when some part of the memory is both *hidden* (has not yet affected the its external behavior) and *meaningful* (may eventually influence the its future external behavior). The commitment functionality $\mathcal{F}_{\text{COM}}$ represents the canonical functionality which uses its internal memory in such a way (between the commit and reveal phases).

Automata-theoretic Characterization. To formally define these intuitive non-triviality conditions, we develop three new important properties, all defined automata-theoretically.

Say that an input $\hat{x}$ *dominates* another input x if (informally) Alice can use $\hat{x}$ as her input to $\mathcal{F}$ in the first round of interaction, but then convince any environment that she had really used x (Definition 9). In other words, any behavior that can be induced by sending x to $\mathcal{F}$ in the first round can also be induced by instead sending $\hat{x}$ and thereafter engaging in some local "translation" protocol. We emphasize that Alice must perform this translation *online*, without knowledge of the inputs that the environment will provide in future rounds. When $\hat{x}$ dominates x, Alice can use $\hat{x}$ in place of x in the first round without loss of generality.

The input-output behavior of each state in the functionality naturally defines a corresponding SFE. Take any SFE and say that $x \sim x'$ if Alice inputs x and x' always induce the same output for Bob. When an SFE is trivial, then Bob's output from the SFE always reveals the $\sim$-equivalence class of Alice's input (and vice-versa, exchanging the roles of Alice and Bob).[7] We say that the start state of $\mathcal{F}$ is *simple* if its associated

[7] In fact, this is true whenever the SFE is (isomorphic to) a *simultaneous exchange* function $\mathcal{F}(x, y) = (x, y)$. Thus our protocol for $\mathcal{F}_{\text{COM}}$ (from Theorem 4) which exploits these automata-theoretic properties can also be used when the functionality is comprised

SFE is a trivial SFE, *and* if each equivalence class of $\sim$ (over Alice inputs and Bob inputs) contains some input that dominates all other inputs in its class (Definition 13).

Intuitively, when the start state is simple, then just by looking at the output from the first round, Bob can determine the $\sim$-class of the input Alice used. Consequently, Bob can safely assume that Alice used the input x that dominates that $\sim$-class, since x is the input that gives Alice the maximal flexibility over influencing the functionality's future behavior. The same is true for Alice determining Bob's likely input. Suppose x and y are inputs for Alice and Bob, respectively, which are each maximally dominant for their $\sim$-equivalence classes. We call the transition from the start state on inputs (x, y) a *safe transition* (Definition 14). Intuitively, only such safe transitions are relevant; furthermore, after a safe transition, neither party has meaningful uncertainty about the other party's input in the previous round, or the functionality's resulting state.

These automata-theoretic properties characterize non-triviality as follows:

Theorem 4. *Let $\mathcal{F}$ be a deterministic, finite (reactive) functionality. Then the following are equivalent:*

1. *$\mathcal{F}$ is non-trivial.*
2. *$\mathcal{F}_{\text{COM}} \sqsubseteq_{\text{STAT}} \mathcal{F}$ or $\mathcal{G} \sqsubseteq_{\text{STAT}} \mathcal{F}$ for some non-trivial SFE functionality $\mathcal{G}$.*
3. *Every state reachable from $\mathcal{F}$'s start state via a sequence of safe transitions is a safe state.*

Condition (3) of this theorem can be expressed completely combinatorially, giving the first *combinatorial* characterization of triviality (and thus completeness) for any class of arbitrary reactive functionalities.

Proof Sketch. The full proof is given in Appendix E. We consider all the states of the machine that are reachable by a sequence of safe transitions; intuitively, these are the only states that are of any significance. If all such states are simple, then $\mathcal{F}$ has a trivial protocol (Lemma 10). Intuitively, if at each state $\mathcal{F}$ evaluates a a trivial SFE, and if after each round, both parties have no uncertainty about the next state of $\mathcal{F}$, then the protocol for $\mathcal{F}$ is a straight-forward composition of trivial SFE protocols.

Otherwise, assume that one of the safely reachable states of $\mathcal{F}$ is non-simple; for simplicity, assume it is the start state. If the start state is non-simple because of its input-output behavior, then there is an elementary protocol which securely realizes that associated SFE in the $\mathcal{F}$-hybrid model. Otherwise, the start state is non-simple because there exist two inputs for (by symmetry) Alice, say x_0 and x_1, which are in the same $\sim$-class, but no Alice input dominates *both* of $\{x_0, x_1\}$. In other words, any first-round input for Alice will "bind" her to the behaviors consistent with x_0 or to those consistent with x_1, but not both.

We formalize this natural connection to commitment by constructing an unconditional commitment protocol, as follows (Lemma 9). Alice commits to b by sending x_b in the first round. The commitment is perfectly hiding since $x_0 \sim x_1$. To reveal, it suffices for Alice to convince Bob that in the first round she used an input that dominates x_b, since no input can dominate both x_0 and x_1. We argue that if Alice sends an input in the first round that *doesn't* dominate x_b there is a *fixed* environment that has a probability of forcing $\mathcal{F}$'s external behavior to be inconsistent with x_b, with some probability. Thus our protocol instructs Bob to play the role of such an environment, sending a sequence inputs to $\mathcal{F}$ himself and sending a sequence of inputs to Alice. Just like in the definition of domination, Alice must report back to Bob her own purported responses from $\mathcal{F}$, in an online manner, and there is no way to do this with guaranteed consistency if she used input x_{1-b} in the first round. Bob aborts if these responses or his own responses from $\mathcal{F}$ are not consistent with Alice having sent x_b in the first round. By repeating this basic protocol in parallel an appropriate number of times, Bob can be assured of catching Alice with overwhelming probability if she tries to equivocate. □

of simultaneous exchange SFEs instead of trivial SFEs, as observed and applied in [MPR10].

6 Necessity of the sh-OT Assumption

Finally, we show that the sh-OT assumption is not only sufficient but also necessary for the zero-one law.

Theorem 5. *If the zero-one law is true, then the* sh-OT *assumption is true.*

Proof. If the zero-one law holds, then $\mathcal{F}_{\text{OT}} \sqsubseteq_{\text{PPT}} \mathcal{F}_{\text{XOR}}$, since $\mathcal{F}_{\text{XOR}}$ is unconditionally non-trivial. $\mathcal{F}_{\text{OT}}$ has the property that any protocol that securely realizes $\mathcal{F}_{\text{OT}}$ (against active adversaries) is also secure against semi-honest adversaries (see [PR08] for more details). Hence the given $\mathcal{F}_{\text{OT}}$ protocol is secure against semi-honest adversaries, in the $\mathcal{F}_{\text{XOR}}$-hybrid model. Since $\mathcal{F}_{\text{XOR}}$ has an elementary plain protocol unconditionally secure against semi-honest adversaries, we can compose these two protocols to obtain a plain protocol that securely realizes $\mathcal{F}_{\text{OT}}$ against semi-honest adversaries. □

We remark that we use the sh-OT assumption not only to implement oblivious transfer, but even to implement $\mathcal{F}_{\text{COM}}$ using $\mathcal{F}_{\text{XOR}}$. The above argument implies nothing about the necessity of the sh-OT assumption for $\mathcal{F}_{\text{COM}} \sqsubseteq_{\text{PPT}} \mathcal{F}_{\text{XOR}}$. However, we point out that sh-OT assumption is in fact a necessary condition for the existence of such a protocol [DG03]. Further, in [MPR10] it is shown that for *any* functionality $\mathcal{F}$, $\mathcal{F} \sqsubseteq \mathcal{F}_{\text{XOR}}$ is either unconditionally true/false, or else $\mathcal{F} \sqsubseteq_{\text{PPT}} \mathcal{F}_{\text{XOR}}$ implies the sh-OT assumption. As such, our dependence on sh-OT assumption to demonstrate the completeness of $\mathcal{F}_{\text{XOR}}$ was necessary.

7 Extensions, Limitations, and Open Problems

We discuss several natural extensions of our main theorem:

Strengthening the Reduction. In general, as one tightens the notion of a reduction, fewer functionalities remain complete. In the extreme, the reduction could be made so restrictive that no functionality reduces to another. In Appendix F we discuss several possible strengthenings of the $\sqsubseteq_{\text{PPT}}$ reduction. We first note that the zero-one law still applies if protocols are given only public channels instead of private channels, or if security is simultaneously required against both active and semi-honest adversaries.

However, if the reduction notion is strengthened to require security against *computationally unbounded* adversaries, then the zero-one law breaks down. Even in the case of SFE functionalities, there exist infintely many qualitative distinctions among functionalities with respect to this stronger reduction [MPR09].

In Appendix G.1, we show that if the reduction requires parties to use the given ideal functionality with only *fixed roles* (i.e., Alice can access $\mathcal{F}$ only in the role of Alice), then $\mathcal{F}_{\text{COM}} \not\sqsubseteq \mathcal{F}_{\text{CC}}$ (indeed, the behavior of $\mathcal{F}_{\text{CC}}$ is not symmetric with respect to the two parties). Since $\mathcal{F}_{\text{CC}}$ is unconditionally non-trivial, the zero-one law no longer holds under this strong reduction. This impossibility highlights the fact that $\mathcal{F}_{\text{CC}}$ indeed has rather low complexity, and justifies our somewhat complicated protocol used to realize $\mathcal{F}_{\text{COM}}$ using $\mathcal{F}_{\text{CC}}$.

Finally, if the reduction is strengthened to require security against *adaptive* corruption, we are unsure whether the zero-one law still holds, even for the class of deterministic, finite functionalities.

Larger Classes of Functionalities. In Appendix G.2 we show that the zero-one law does not extend to deterministic, *unbounded-memory* functionalities. Let $\mathcal{F}$ be a channel which accepts an arbitrary-length string x from Alice and sends $f(x)$ to Bob for a fixed function f. Assuming one-way functions exist, we construct an efficient f that is hard to invert on infinitely many input lengths (thus $\mathcal{F}$ is non-trivial), yet trivially invertible for very long stretches of input lengths (thus $\mathcal{F}$ is cryptographically useless by protocols with certain security parameters). Of course, if one-way functions do not exist, then the sh-OT assumption is false and the zero-one law must still break down, so the break-down of the zero-one law is unconditional. While our construction of f is admittedly contrived, this impossibility result does illustrate 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.

The other natural way to extend the scope of our results is to consider *randomized* functionalities. However, very little is known about randomized functionalities, even in the simplest case of SFE functionalities

and considering perfect security against computationally unbounded, semi-honest adversaries (for comparison, the corresponding characterization for *deterministic* SFE has been known for 20 years [K89B, B89]).

Optimizing Hardness Assumptions. While our main theorem relies on the minimal sh-OT assumption, our use of the assumption itself is non-black-box. In both of our new commitment protocols, we use standalone zero-knowledge and witness-indistinguishable proofs of statements regarding various cryptographic primitives (which are derived eventually from the sh-OT assumption). We do not know whether such non-black-box usage of the assumption is necessary, although it appears that a fundamentally different approach would be required to avoid the use of interactive proofs.

Acknowledgments

We acknowledge helpful discussions with Ran Canetti, Yuval Ishai and Amit Sahai. The protocol in Theorem 2 was simplified from its original form (appearing in [R09]) partly motivated by the recent results of [DNO09].

References

[B89] Donald Beaver. Perfect privacy for two-party protocols. In Joan Feigenbaum and Michael Merritt, editors, *Proceedings of DIMACS Workshop on Distributed Computing and Cryptography*, volume 2, pages 65–77. American Mathematical Society, 1989.

[BMM99] Amos Beimel, Tal Malkin, and Silvio Micali. The all-or-nothing nature of two-party secure computation. In Michael J. Wiener, editor, *CRYPTO*, volume 1666 of *Lecture Notes in Computer Science*, pages 80–97. Springer, 1999.

[C87] Claude Crépeau. Equivalence between two flavours of oblivious transfers. In Carl Pomerance, editor, *CRYPTO*, volume 293 of *Lecture Notes in Computer Science*, pages 350–354. Springer, 1987.

[C01] Ran 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.

[CF01] Ran Canetti and Marc Fischlin. Universally composable commitments. Report 2001/055, Cryptology ePrint Archive, July 2001. Extended abstract appeared in CRYPTO 2001.

[CK89] Benny Chor and Eyal Kushilevitz. A zero-one law for boolean privacy (extended abstract). In *STOC*, pages 62–72. ACM, 1989.

[CKL03] Ran Canetti, Eyal Kushilevitz, and Yehuda Lindell. On the limitations of universally composable two-party computation without set-up assumptions. In Eli Biham, editor, *EUROCRYPT*, volume 2656 of *Lecture Notes in Computer Science*. Springer, 2003.

[CLOS02] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party computation. In *Proc.* 34*th STOC*, pages 494–503. ACM, 2002.

[DG03] Ivan Damgård and Jens Groth. Non-interactive and reusable non-malleable commitment schemes. In *Proc.* 35*th STOC*, pages 426–437. ACM, 2003.

[DNO09] Ivan Damgård, Jesper Buus Nielsen, and Claudio Orlandi. On the necessary and sufficient assumptions for uc computation. Cryptology ePrint Archive: Report 2009/247, 2009.

[EGL85] Shimon Even, Oded Goldreich, and Abraham Lempel. A randomized protocol for signing contracts. *Commun. ACM*, 28(6):637–647, 1985.

[G01] Oded Goldreich. *Foundations of Cryptography: Basic Tools*. Cambridge University Press, 2001. Earlier version available on `http://www.wisdom.weizmann.ac.il/~oded/frag.html`.

[G04] Oded Goldreich. *Foundations of Cryptography: Basic Applications*. Cambridge University Press, 2004.

[GKM+00] Yael Gertner, Sampath Kannan, Tal Malkin, Omer Reingold, and Mahesh Viswanathan. The relationship between public key encryption and oblivious transfer. In *FOCS*, pages 325–335, 2000.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play ANY mental game. In ACM, editor, *Proc.* 19*th STOC*, pages 218–229. ACM, 1987. See [G04, Chap. 7] for more details.

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999. Preliminary versions appeared in STOC' 89 and STOC' 90.

[HNRR06] Danny Harnik, Moni Naor, Omer Reingold, and Alon Rosen. Completeness in two-party secure computation: A computational view. *J. Cryptology*, 19(4):521–552, 2006.

[I95] Russell Impagliazzo. A personal view of average-case complexity. In *Structure in Complexity Theory Conference*, pages 134–147, 1995.

[IL89] Russell Impagliazzo and Michael Luby. One-way functions are essential for complexity based cryptography (extended abstract). In *Proc.* 30*th FOCS*, pages 230–235. IEEE, 1989.

[IPS08] Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Founding cryptography on oblivious transfer - efficiently. In David Wagner, editor, *CRYPTO*, volume 5157 of *Lecture Notes in Computer Science*, pages 572–591. Springer, 2008.

[K88] Joe Kilian. Founding cryptography on oblivious transfer. In *STOC*, pages 20–31. ACM, 1988.

[K89A] Joe Kilian. *Uses of Randomness in Algorithms and Protocols.* PhD thesis, Department of Electrical Engineering and Computer Science, Mas sachusetts Institute of Technology, 1989.

[K89B] Eyal Kushilevitz. Privacy and communication complexity. In *FOCS*, pages 416–421. IEEE, 1989.

[K91] Joe Kilian. A general completeness theorem for two-party games. In *STOC*, pages 553–560. ACM, 1991.

[K00] Joe Kilian. More general completeness theorems for secure two-party computation. In *Proc.* 32*th STOC*, pages 316–324. ACM, 2000.

[KKMO00] Joe Kilian, Eyal Kushilevitz, Silvio Micali, and Rafail Ostrovsky. Reducibility and completeness in private computations. *SIAM J. Comput.*, 29(4):1189–1208, 2000.

[KMQ08] Daniel Kraschewski and Jörn 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`.

[KMQR09] Robin Künzler, Jörn Müller-Quade, and Dominik Raub. Secure computability of functions in the it setting with dishonest majority and applications to long-term security, 2009.

[MPR09] Hemanta K. Maji, Manoj Prabhakaran, and Mike Rosulek. Complexity of multi-party computation problems: The case of 2-party symmetric secure function evaluation. In Omer Reingold, editor, *TCC*, volume 5444 of *Lecture Notes in Computer Science*, pages 256–273. Springer, 2009.

[MPR10] Hemanta Maji, Manoj Prabhakaran, and Mike Rosulek. Cryptographic complexity classes and computational complexity assumptions. To appear, *First Symposium on Innovations in Computer Science (ICS 2010)*, 2010.

[MS83] F. J. MacWilliams and N. J. A. Sloane. *The Theory of Error-Correcting Codes*. North-Holland Mathematical Library. North Holland, January 1983.

[N91] Moni Naor. Bit commitment using pseudorandomness. 4(2):151–158, 1991. Preliminary version in CRYPTO' 89.

[NV06] Minh-Huyen Nguyen and Salil P. Vadhan. Zero knowledge with efficient provers. In *STOC*, pages 287–295. ACM, 2006.

[P05] Manoj Prabhakaran. *New Notions of Security*. PhD thesis, Department of Computer Science, Princeton University, 2005.

[PR08] Manoj Prabhakaran and Mike Rosulek. Cryptographic complexity of multi-party computation problems: Classifications and separations. In David Wagner, editor, *CRYPTO*, volume 5157 of *Lecture Notes in Computer Science*, pages 262–279. Springer, 2008.

[R81] M. Rabin. How to exchange secrets by oblivious transfer. Technical Report TR-81, Harvard Aiken Computation Laboratory, 1981.

[R90] John Rompel. One-way functions are necessary and sufficient for secure signatures. In *Proc. 22nd STOC*, pages 387–394. ACM, 1990.

[R09] Mike Rosulek. *The Structure of Secure Multi-Party Computation*. PhD thesis, Department of Computer Science, University of Illinois at Urbana-Champaign, 2009.

[Y86] Andrew Chi-Chih Yao. How to generate and exchange secrets. In *Proc. 27th FOCS*, pages 162–167. IEEE, 1986.

A Related Work

Some of the earliest works in secure multi-party computation already established that under reasonable computational assumptions, all functionalities are realizable [Y86, GMW87].[8] In our parlance, this would suggest that every functionality has the same cryptographic complexity — namely, being trivial and complete. This is indeed the case (under those computational assumptions), if we restrict to security against semi-honest adversaries or standalone security (and polynomial time entities) as was done in [Y86, GMW87].

However, shortly afterwards, a finer study of cryptographic complexity emerged, by using stronger security notions. This strengthening was by removing computational restrictions. In the computationally

[8] Many of the results in secure multi-party computation, including [GMW87], address the setting of more than two parties. In this work we restrict our attention to the 2-party setting.

unbounded setting (a.k.a information theoretic or statistical security setting), not all functionalities are realizable (or "trivial"), and hence it becomes possible to distinguish the complexity of functionalities as trivial and complete (and more generally, to consider *degrees* of various functionalities).

One of the earliest, and most important results along this line is that SFE functionality of Oblivious Transfer (OT) [R81, EGL85, C87] is complete [K88], unconditionally.[9] All subsequent completeness results crucially rely on this result. These include the characterization of complete asymmetric SFE functionalities [K00] and general SFE functionalities [KMQ08].

However much of the initial progress, apart from the completeness of OT, was in the context of security against semi-honest (a.k.a honest-but-curious, and passive) adversaries. Further — and this has been true for almost all the prior work discussed here — only non-reactive (a.k.a secure function evaluation, or SFE) functionalities were considered. Building on early work by [CK89, K89B, B89] (who consider the case of *perfect security*), recently [MPR09] characterized realizable *symmetric*[10] SFEs. Independently, [KMQR09] extended this characterization to general SFEs. [K91, K00] characterize complete symmetric and asymmetric SFEs. For symmetric SFEs, restricted boolean functions, [KKMO00] show a zero-one law, that every such functionality is either trivial or complete.

Another significant progress has been in the setting of *standalone* security. When considering active corruption, composability is significant, and the results in the standalone setting typically do not extend to the UC setting.[11] Nevertheless, studying standalone security does provide important insight into the cryptographic complexity of functionalities. In particular, [BMM99] provided a zero-one law that if any non-trivial asymmetric SFE is securely realizable, then so is every asymmetric SFE. The notion of security in [BMM99] is weak in that it does not consider composability, and restricts to polynomial time entities; however it is strong in that a protocol is considered secure only if it is simultaneously secure against both active and semi-honest corruption. (Our main theorem extends unaltered with this notion of reduction.) [BMM99] shows that every asymmetric SFE is securely realizable (trivial, in our terminology) if and only if there is a semi-honest secure protocol for OT, sh-OT. While it bears some similarity with our results, there are important aspects in which our results provide a very different picture of cryptographic complexity compared to what [BMM99] obtains by considering standalone security. In particular, under the (widely believed) cryptographic assumption that a sh-OT protocol does exist, there is only one level of complexity for all asymmetric SFE functionalities (i.e., all are trivial) by [BMM99], where as in our picture, even restricted to asymmetric SFEs, there are two distinct complexity levels (trivial and complete), no matter what computational assumptions are made.

We remark that [HNRR06] extends results of [BMM99] beyond constant-sized functionalities (but still restricted to asymmetric SFE functionalities); we observe that on extending beyond constant-sized functionalities, like for [HNRR06], for us too the collapse of the complexity levels is not comprehensive. In fact, in our case, when considering non-constant-sized SFE functionalities, we can unconditionally demonstrate the existence of SFE functionalities which are neither trivial nor complete. We leave further exploration of this space for future work.

The introduction of the UC model, as a culmination of a long line of work on composable security, gave a whole new way to understand the cryptographic complexity of functionalities. Right from the beginning, unconditional impossibilities (or non-trivial functionalities) were observed [C01, CF01], which was extended to a large class of SFE functionalities in [CKL03]; subsequently [PR08] provided an exact characterization of all trivial functionalities (as "splittable" functionalities).

[9] The protocol in [K88] is not UC-secure, though an extension presented in [K89A] is likely to be. The construction was significantly simplified, and proven secure in the UC setting in [IPS08].

[10] In a symmetric SFE, both parties receive the same output $f(x, y)$, where x, y stand for their inputs. In an *asymmetric* SFE only one party receives the output (and the other party is given no output). A general SFE provides two, possibly different, outputs $f_A(x, y)$ and $f_B(x, y)$ to the two parties.

[11] Some important exceptions are the key completeness results in information theoretic setting, like the ones in [K88] and [K00].

In the computationally unbounded setting [MPR09] showed that there is an infinite hierarchy of increasing levels of cryptographic complexity even among symmetric SFE functionalities, as well as functionalities with incomparable complexities. However, this does not give any indication of the complexity landscape in the computationally bounded setting. In [PR08] it was conjectured that under some standard cryptographic assumptions, there should be only two levels of complexity in the computationally bounded setting. Our main result is a sharp resolution of this conjecture, by giving a necessary and sufficient complexity assumption under which the conjecture is true.

Under complexity assumptions, it was already known that some functionalities which are not complete in the computationally unbounded setting become complete in the computationally bounded setting. In particular, [CLOS02] provided the first such result, that the "coin-tossing" functionality is complete, assuming the existence of enhanced trapdoor permutations and dense cryptosystems. The approach in [CLOS02], following that of [GMW87], is to note that if a sh-OT protocol exists, then the commitment functionality (which is a reactive functionality) is complete; then the commitment functionality is reduced to the coin-tossing functionality under the computational assumptions described above. Our approach follows [GMW87, CLOS02] in that to establish the completeness of a given functionality, we need only reduce the commitment functionality to it. However, our reductions are much more general than that in [CLOS02] in that we can reduce commitment to *any* non-trivial (deterministic, constant-sized) functionality. Secondly, we rely only on the minimal assumption (existence of a sh-OT protocol); for the completeness of the coin-tossing functionality, such a result was obtained independently by [DNO09].

We point out that all the above mentioned prior results, except [PR08], are restricted to SFE functionalities, where as an important contribution of this work is to develop tools and techniques for analysing reactive functionalities. On the other hand, the current set of results do share some restrictions common to prior work: for most part, we consider the complexity of deterministic functionalities (the exceptions in prior work being [PR08] and the semi-honest security results of [K00]); secondly we consider only static corruption (a notable exception in prior work being [CLOS02]).

B Non-Reactive Functionalities

B.1 Definitions

A *secure function evaluation (SFE) functionality* $\mathcal{F}$ is fully specified by a pair of functions (f_A, f_B) over a finite input domain $X \times Y$. The behavior of $\mathcal{F}$ as an ideal functionality is defined formally in Figure 2. We emphasize that an SFE functionality provides no guarantee about output fairness — the adversary is in complete control over the delivery of outputs.

1. Wait for input $x \in X$ from Alice and input $y \in Y$ from Bob.
2. If Alice is corrupt, then send $(\text{OUTPUT}, f_A(x, y))$ to the adversary; if Bob is corrupt, then send $(\text{OUTPUT}, f_B(x, y))$ to the adversary; otherwise, send OUTPUT to the adversary.
3. On input DELIVER from the adversary, or if neither party is corrupt, send $f_A(x, y)$ to Alice and $f_B(x, y)$ to Bob.

Figure 2: Semantics of the SFE functionality $\mathcal{F} = (f_A, f_B)$

Definition 1. *Let $\mathcal{F} = (f_A, f_B)$ be a 2-party SFE. We say that x is a* redundant input for Alice *if there exists $x' \neq x$ such that:*

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

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 it is never in the best interests of a malicious party to supply a redundant input.

Definition 2. *Let $\mathcal{F} = (f_A, f_B)$ and $\mathcal{G} = (g_A, g_B)$ be 2-party SFEs. 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,*
- *Injectively re-labeling a party's input domain,*
- *Injectively 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.*

It is easy to see that if $\mathcal{F}$ and $\mathcal{G}$ are isomorphic, then $\mathcal{F}$ can be securely realized in the $\mathcal{G}$-hybrid model (and vice-versa). Thus, we hereafter consider only SFE functionalities that have all redundant inputs removed.

B.2 2×2 Minors

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

Definition 3. *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 4 ([KMQ08])**.** *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 1 ([KMQ08])**.** *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 $\sqsubseteq_{\text{STAT}}$ reductions.*

In fact, the lemma proven by Kraschewski and Müller-Quade [KMQ08] is stronger, giving a complete characterization of completeness against computationally unbounded adversaries. That is, they prove that $\mathcal{F}$ is $\sqsubseteq_{\text{STAT}}$-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 even against computationally unbounded adversaries. Of course, in this work we show that many other functionalities are also complete under the $\sqsubseteq_{\text{PPT}}$ reduction.

Definition 5. *$\mathcal{F} = (f_A, f_B)$ 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.

B.3 Combinatorial Classification

We now prove the first characterization of SFE functionalities, using two technical lemmas:

Lemma 2. *Let $\mathcal{F} = (f_A, f_B)$ be an SFE functionality. If $\mathcal{F}$ has a generalized* CC*-minor and no generalized* OR*-minor, then $\mathcal{F}_{\text{CC}} \sqsubseteq_{\text{STAT}} \mathcal{F}$.*

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.[12] 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.

[12] 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 $r = i$ 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.
- 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 3. *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:

Theorem 1 (restated). *Let $\mathcal{F}$ be an SFE functionality. Then $\mathcal{F}$ is non-trivial if and only if $\mathcal{F}_{\text{XOR}} \sqsubseteq_{\text{STAT}} \mathcal{F}$ or $\mathcal{F}_{\text{CC}} \sqsubseteq_{\text{STAT}} \mathcal{F}$ or $\mathcal{F}_{\text{OT}} \sqsubseteq_{\text{STAT}} \mathcal{F}$.*

Proof. If $\mathcal{F}$ contains no generalized OR-minor, but contains a generalized CC-minor, then $\mathcal{F}_{\text{CC}} \sqsubseteq_{\text{STAT}} \mathcal{F}$ (by Lemma 2). Otherwise, if $\mathcal{F}$ contains neither a generalized CC-minor nor a generalized OR-minor, then $\mathcal{F}$ is a symmetric exchange function (Lemma 3). 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 $\min\{|X|, |Y|\} < 2$, then $\mathcal{F}$ is trivial, by the unconditional, complete characterization of UC triviality from [PR08] ($\mathcal{F}$ is trivially realizable via a protocol in which one party simply sends their input to the other party).

Otherwise, suppose $\min\{|X|, |Y|\} > 2$. Then let $x_0 \neq x_1 \in X$ and $y_0 \neq y_1 \in Y$. A secure protocol for $\mathcal{F}_{\text{XOR}}$ in the $\mathcal{F}$-hybrid model is as follows: On input $a \in \{0, 1\}$, Alice sends x_a; on input $b \in \{0, 1\}$, Bob sends input y_b. Both parties obtain output $\mathcal{F}(x_a, y_b) = (x_a, y_b)$ and output $a \oplus b$. 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. □

C Obtaining $\mathcal{F}_{\text{COM}}$ from $\mathcal{F}_{\text{XOR}}$

In this section, we show how $\mathcal{F}_{\text{COM}}$ can be directly realized using $\mathcal{F}_{\text{XOR}}$. We first observe that $\mathcal{F}_{\text{COIN}} \sqsubseteq_{\text{STAT}} \mathcal{F}_{\text{XOR}}$ via an elementary and well-known protocol. Thus we focus on proving that $\mathcal{F}_{\text{COM}} \sqsubseteq_{\text{PPT}} \mathcal{F}_{\text{COIN}}$.

Parameters. Our construction depends on the sh-OT assumption, so let ψ_{sh} denote a semi-honest protocol for OT.

sh-OT assumption implies the existence of one-way functions, and our construction also relies on the following components, each of which exists given one-way functions alone:

- Let Com be a statistically binding, standalone-secure commitment scheme with a non-interactive reveal phase (for instance, Naor's commitment scheme [N91]).
- Let $G : \{0, 1\}^k \to \{0, 1\}^{2k}$ be a pseudorandom generator.
- A standalone-secure zero-knowledge proof of knowledge protocol for NP statements.
- A standalone-secure witness-indistinguishable proof for NP statements.

The protocol. We define the following interactive protocol ρ in the $\mathcal{F}_{\text{COIN}}$-hybrid model. The security parameter is κ. Suppose the ψ_{sh} protocol uses $R(\kappa)$ bits of randomness when executed with security parameter κ.

1. (Commit phase.) Both parties obtain random coins $\sigma_A, \sigma_B \in \{0, 1\}^{2\kappa}$ from $\mathcal{F}_{\text{COIN}}$.
2. On input (COMMIT, b), for $b \in \{0, 1\}$, Alice first commits to b using the Com protocol.
3. Alice and Bob both choose random coin shares, $r_A, r_B \leftarrow \{0, 1\}^{R(\kappa)}$ respectively, for use in the ψ_{sh} protocol. Each party commits to its respective coins using the Com protocol.
4. Both parties use a ZK proof of knowledge protocol to prove knowledge of the values underlying their commitments (for Alice, the commitments to b and r_A; for Bob, the commitment to r_B).
5. Alice chooses random coins $r'_B \leftarrow \{0, 1\}^{R(\kappa)}$ and sends them to Bob. Bob chooses random coins $r'_A \leftarrow \{0, 1\}^{R(\kappa)}$ and sends them to Alice.
6. Both parties engage in the OT protocol ψ_{sh}, with Alice acting as the sender with inputs $x_0 = 0$ and $x_1 = b$, and Bob acting as the receiver with input (choice bit) $y = 0$.
   (a) For each of Alice's turns in the ψ_{sh} protocol, suppose that the protocol instructs her to send message m. She sends m to Bob and then proves using a witness-indistinguishable proof protocol the following statement: Either there exists d such that $G(d) = \sigma_A$, or else there exists values r_A and b that are consistent with the commitments to r_A and b, and the ψ_{sh} protocol prescribes

that the sender send m when running on input $(x_0 = 0, x_1 = b)$ and random coins $r_A \oplus r'_A$, given the ψ_{sh} transcript so far.

(b) For each of Bob's turns in the ψ_{sh} protocol, suppose that the protocol instructs him to send message m. He sends m to Alice and then proves using a witness-indistinguishable proof protocol the following statement: Either there exists d such that $G(d) = \sigma_B$, or else there exists coins r_B that are consistent with his commitment to r_B, and the ψ_{sh} protocol prescribes that the receiver send m when running on input $y = 0$ and random coins $r_B \oplus r'_B$, given the ψ_{sh} transcript so far.

7. If any of the interactive proofs in the previous steps fail, then the parties abort. Otherwise Bob outputs COMMITTED.

8. (Reveal phase.) On input REVEAL, Alice sends b to Bob. She then proves using a witness-indistinguishable proof that either there exists a non-interactive opening of Alice's Com-commitment from step (2) to the value b, or else there exists d such that $G(d) = \sigma_A$. Bob outputs (REVEAL, b) if this proof verifies.

Overview and Motivation. Intuitively, the protocol is for Alice to commit to b in a standalone-secure commitment scheme, and then use b as an input to the ψ_{sh} OT protocol. However, Bob will choose not to pick up b in the OT subprotocol, so that he is still oblivious to its value.

The rest of the protocol (most importantly steps 3–6) essentially "compiles" the ψ_{sh} protocol using the standard GMW paradigm to enforce that Alice indeed uses b as an input to ψ_{sh}, and that Bob indeed chooses not to pick it up. However, for technical reasons we deviate slightly from the GMW paradigm in the following ways:

- In step (5), the parties prove knowledge of the commitments to their private inputs to ψ_{sh} and their random-tape shares. This is important because we eventually reduce an adversary running ρ to a semi-honest adversary running ψ_{sh}. For technical reasons, we need to extract inputs and the random tape share in this step, but a rewinding extraction is sufficient.
- Bob does not prove the standard GMW-style statement in step (6a). Instead, he proves a statement containing a trapdoor clause related to the public coins σ. This extra trapdoor allows a straight-line simulator for a corrupt Bob to prove false statements about its interaction using ψ_{sh} (by choosing σ from the range of G and using the trapdoor witness), which is crucial in our subsequent security reductions.

Finally, in the reveal phase Alice does not simply reveal the standalone-secure commitment to b made in step (2). Instead, she proves it with a witness-indistinguishable proof that has a "trapdoor" clause regarding the public coins σ. This extra clause allows a straight-line simulator for a corrupt Alice to open a commitment to either value.

Theorem 2 (restated). *The protocol ρ is a secure realization of $\mathcal{F}_{\text{COM}}$ in the $\mathcal{F}_{\text{COIN}}$-hybrid model.*

Proof. We must demonstrate a suitable simulator for any adversary. The construction is trivial in the cases where both or neither of the parties are corrupt. The correctness of the protocol follows straight-forwardly from the security properties of the various components used in ρ. We focus on the other two cases, in which only one party is corrupt.

Simulation when Alice is corrupt. In this case the primary task of the simulator is to extract Alice's bit during the commit phase. We construct the simulator via a sequence of hybrid interactions, as follows:

Real interaction: The simulator honestly plays the role of $\mathcal{F}_{\text{COIN}}$ and Bob (who has no input) in the ρ protocol. This is exactly what happens in the real interaction with Alice.

Hybrid 1: Same as above, except that the simulator generates coins σ_B by choosing a random $d \in \{0,1\}^\kappa$ and setting $\sigma_B = G(d)$. The coins σ_A remain honestly generated. This hybrid is indistinguishable from the previous by the pseudorandomness of G.

Hybrid 2: Same as above, except that each time in step (6b), the simulator uses d as a witness to the witness-indistinguishable proofs. This hybrid is indistinguishable from the previous by the witness indistinguishability property of the interactive proof.

To obtain hybrid 3, we now define a sequence of intermediate hybrids which use rewinding simulation. This is necessary to eventually reduce the indistinguishability of consecutive hybrids to a *semi-honest* security guarantee. However, the final hybrid 3 is a straight-line simulator for corrupt Alice.

Hybrid 2a: Same as hybrid 2, except that the simulator uses the (possibly rewinding) knowledge extractor to extract Alice's commitments to r_A and b from step (4). By the soundness of the proof of knowledge protocol and the statistical binding property of Com, these values are with overwhelming probability the only values to which the commitments could be opened. This hybrid is indistinguishable from the previous by the security of the ZK proof scheme used in step (4).

Hybrid 2b: Same as hybrid 2a, except that the simulator honestly chooses random coins $R \in \{0,1\}^{R(\kappa)}$ and sends $r'_A = R \oplus r_A$ in step (5), where r_A is the value extracted in the previous step. This interaction is distributed exactly as the previous hybrid.

Hybrid 2c: Same as hybrid 2b, except that each time in step (6a), the simulator computes the next-message function of ψ_{sh} for Alice, given input $x_0 = 0, x_1 = b$, random tape R, and the ψ_{sh} transcript so far. If this next message does not match the m given by Alice in step (6a), then the simulator aborts. This hybrid is indistinguishable from the previous by the soundness property of Alice's interactive proofs in step (6a).[13] Note that hereafter we can be sure that the corrupt Alice is executing ψ_{sh} honestly.[14]

Hybrid 2d: Same as hybrid 2c, except that each time in step (6b), the simulator uses $y = 1$ instead of $y = 0$ as its input to the ψ_{sh} protocol (still executing the protocol honestly). This hybrid is indistinguishable from the previous by the *semi-honest* security of ψ_{sh}, since Alice's interactions in the ψ_{sh} protocol can be simulated by a semi-honest adversary.

Additionally, by the correctness of the ψ_{sh} protocol, the simulator obtains $x_1 = b$ as output from this subprotocol. Recall that by the statistical binding property of Com, and the soundness of the interactive proofs in step (6a), this value b is with overwhelming probability the only value to which the commitment in step (2) can be opened.[15]

Hybrid 2e: Same as hybrid 2d, except that the simulator does not extract Alice's inputs b and r_A in step (4), chooses r'_A uniformly in step (5), and does not compare Alice's messages in step (6a) to the "correct" value prescribed by ψ_{sh}. This hybrid is indistinguishable from the previous by applying the arguments in the previous three hybrids in reverse.

Hybrid 3: Exactly the same as hybrid 2e. Overall, this hybrid differs from hybrid 2 only in that the simulator uses $y = 1$ as its honest input to the ψ_{sh} protocol instead of $y = 0$. It then learns the value of b.

[13] Only with negligible probability is σ_A in the range of the pseudorandom generator G. Thus, that clause of the WI proof is false, and the soundness property implies the correctness of the other clause being proved.

[14] Here it is important that Alice is executing ψ_{sh} honestly on *honestly sampled* random tape R. In other words, it is not enough that Alice's random tape be randomly distributed, since she may be able to sample with some trapdoor. We must have Alice execute the protocol on a random tape that was sampled entirely honestly.

[15] The simulator has already extracted b previously in step (4), but subsequent hybrids will not perform such an extraction. Subsequent hybrids will, however, obtain b from the ψ_{sh} subprotocol.

Although hybrids 2a through 2d used rewinding simulation (to reduce a property to the semi-honest security of ψ_{sh}), hybrid 3 itself uses no rewinding.

Hybrid 3 defines our final simulator. After learning the value of b after step (6), the simulator sends (COMMIT, b) to $\mathcal{F}_{\text{COM}}$ in the ideal world. Recall that this extracted value of b is with overwhelming probability the only value to which the commitment in step (2) can be opened. Then, since σ_A is not in the pseudorandom distribution except with negligible probability, and by the soundness of the witness-indistinguishable proof in step (8), b is with overwhelming probability the only value for which Alice can make Bob output (REVEAL, b) in the real-interaction reveal phase.

We see that our simulation is indistinguishable from the real interaction, as desired.

Simulation when Bob is corrupt. In this case the primary task of the simulator is to give an equivocal commitment that can be opened to either value. We construct the simulator via a sequence of hybrid interactions, as follows:

Real interaction: We consider an interaction in the ideal world with a variant of $\mathcal{F}_{\text{COM}}$ which reveals b in the *commit phase*. When receiving $(\text{COMMITTED}, b)$ from $\mathcal{F}_{\text{COM}}$, the simulator for Bob honestly simulates $\mathcal{F}_{\text{COIN}}$ and the behavior of Alice on input b in the commit phase of ρ. When receiving (REVEAL, b) from $\mathcal{F}_{\text{COM}}$, the simulator honestly simulates the behavior of Alice in the reveal phase of ρ. The outcome of this interaction is identical to the real interaction.

Hybrid 1: Same as above, except that the simulator generates σ_A by choosing a random $d \in \{0,1\}^\kappa$ and setting $\sigma_A = G(d)$. This interaction is indistinguishable from the previous hybrid by the pseudorandomness of G.

Hybrid 2: Same as above, except that in step (8) and each time in step (6a), the simulator uses the witness d for the witness-indistinguishable proof. This interaction is indistinguishable from the previous hybrid by the witness indistinguishability of the interactive proof used in step (8).

Hybrid 3: Same as above, except that the simulator commits to 0 in step (2). However, the simulator will still use b as an input to the ψ_{sh} subprotocol. Since the opening of this commitment is never used (as a witness in any of the interactive proofs), this interaction is indistinguishable from the previous hybrid by the standalone hiding property of Com.

Hybrid 4: Same as above, except that the simulator executes step (6) using $x_0 = 0, x_1 = 0$ as inputs to the ψ_{sh} subprotocol. This interaction is indistinguishable from the previous hybrid by the semi-honest security of ψ_{sh}. We sketch the sequence of intermediate hybrids between hybrid 3 and 4; the formal details closely follow the techniques applied earlier in this proof, and are omitted:

- Starting from hybrid 3, first, let the simulator extract r_B from Bob in step (4), possibly by rewinding.
- Next, let the simulator choose $r'_B = R \oplus r_B$ (in step (5)) for an honestly sampled R.
- Next, let the simulator abort if Bob sends a message in step (6b) that is not prescribed by the ψ_{sh} protocol on input $y = 0$ and random tape R.
- Now, let the simulator change its input to ψ_{sh} from $x_1 = b$ to $x_1 = 0$. Since Bob is interacting honestly within the ψ_{sh} protocol, this difference is indistinguishable by the sender's semi-honest privacy guarantee of ψ_{sh}.
- Finally, let the simulator "roll back" these first three changes in reverse order, resulting in the final hybrid 4.

Hybrid 4 defines our final simulation. Note that since the hybrid 4 simulator does not use b until the reveal phase (i.e., the simulated commit phase is completely independent of b), this simulator is a valid simulator in the ideal interaction with $\mathcal{F}_{\text{COM}}$ (in which it does not receive b from $\mathcal{F}_{\text{COM}}$ in the commit phase). We see that the simulation is indistinguishable from the real world interaction, as desired. □

D Obtaining $\mathcal{F}_{\text{COM}}$ from $\mathcal{F}_{\text{CC}}$

We first define an intermediate functionality which we call *extractable commitment*. It captures the requirements of a statistically binding, computationally (standalone) hiding commitment protocol with a *straight-line* extracting simulation.

D.1 Extractable Commitment

We start off by introducing some convenient terminology.

Definition 6. *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 7. *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* $\mathsf{opening}(\tau, \gamma) \neq z$.

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 $\mathcal{F}_{\text{EXTCOM}}^{(\omega_L,\omega_R)}$ below, we will require a single protocol to satisfy both the security guarantees.

We define the extractable commitment functionality $\mathcal{F}_{\text{EXTCOM}}^{(\omega_L,\omega_R)}$ in Figure 3. 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 4. *If (ω_L, ω_R) form a pair of complementary statistically binding commitment protocols (and one-way functions exist), then $\mathcal{F}_{\text{COM}} \sqsubseteq_p \mathcal{F}_{\text{EXTCOM}}^{(\omega_L,\omega_R)}$.*

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 [NV06].

Our protocol for $\mathcal{F}_{\text{COM}}$ is as follows, with security parameter κ. It uses ideal access to 3 independent instances of $\mathcal{F}_{\text{EXTCOM}}^{(\omega_L,\omega_R)}$, 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 $\mathcal{F}_2$.

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 3: Functionality $\mathcal{F}_{\text{EXTCOM}}^{(\omega_L, \omega_R)}$: Extractable commitment, parameterized by two syntactic commitment protocols ω_L and ω_R.

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. □

D.2 Obtaining Extractable Commitment from $\mathcal{F}_{\text{CC}}$

In this section, we show how $\mathcal{F}_{\text{EXTCOM}}$ can be securely realized using $\mathcal{F}_{\text{CC}}$. We first show a protocol π in the $\mathcal{F}_{\text{CC}}$-hybrid model, and then define appropriate (π_L, π_R) protocols such that π is a secure realization of $\mathcal{F}_{\text{EXTCOM}}^{(\pi_L,\pi_R)}$.

Parameters. Let Com be a statistically binding, standalone-secure commitment scheme with a non-interactive reveal phase (for instance, Naor's commitment scheme [N91], 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 [MS83].

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 $\mathsf{opening}(\tau, \gamma) = \perp$ 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, $\mathsf{opening}(\tau, \gamma) = c \oplus \langle g, s \rangle$.

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.

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* $\mathcal{F}_{\text{EXTCOM}}^{(\pi_L, \pi_R)}$ *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 7, 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, Bob will never accept in the reveal phase and our extraction is correct. □

E Classification of Reactive Functionalities

E.1 Definitions

Definition 8. *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 defined formally in Figure 4.

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.

We emphasize that, like an SFE, a DFF provides no guarantee about output fairness — the adversary is in complete control over the delivery of outputs.

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. Set $s = f_A(q, x, y)$ and $t = f_B(q, x, y)$. If Alice is corrupt, send (OUTPUT, s) to the adversary; if Bob is corrupt, send (OUTPUT, t) to the adversary; otherwise send OUTPUT to the adversary.
4. On input DELIVER from the adversary, deliver s to Alice and t to Bob. Then update variable $q \leftarrow \delta(q, x, y)$ and repeat from step (2).

Figure 4: Semantics of the DFF functionality $\mathcal{F} = (Q, X, Y, \delta, f_A, f_B, q_0)$

E.2 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 9. *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 model, 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 7, which is crucially used in Theorem 4.

Definition 10. *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 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 11. *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 $\left|\{f_A(q', x', y) \mid \exists (q, q') \in S, y \in Y \text{ such that } f_A(q, x, y) = z\}\right| = 1$,*
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* 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 6. *$x \geq_A x'$ 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 11.
- 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 11, 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 11 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 11 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 11. 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 11. 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 11. 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 7. *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 6, 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.

E.3 Simple States & Safe Transitions

Definition 12. *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 13. *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 14. *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 1. *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 8. *If the start state of $\mathcal{F}$ is simple, then $r(\mathcal{F}) \sqsubseteq_{\text{STAT}} \mathcal{F} \sqsubseteq_{\text{STAT}} r(\mathcal{F})$. Furthermore, if q is reachable from the start state of $\mathcal{F}$ via a safe transition, then $\mathcal{F}[q] \sqsubseteq_{\text{STAT}} \mathcal{F}$.*

Proof. The protocol for $r(\mathcal{F}) \sqsubseteq_{\text{STAT}} \mathcal{F}$ 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 $\mathcal{F} \sqsubseteq_{\text{STAT}} r(\mathcal{F})$ 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 $\mathcal{F}[q] \sqsubseteq_{\text{STAT}} \mathcal{F}$, it suffices to show that $(r(\mathcal{F}))[q] \sqsubseteq_{\text{STAT}} r(\mathcal{F})$. 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. □

E.4 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 4 (restated). *Let $\mathcal{F}$ be a DFF. Then the following are equivalent:*

1. *$\mathcal{F}$ is non-trivial.*
2. *$\mathcal{F}_{\text{COM}} \sqsubseteq_{\text{STAT}} \mathcal{F}$ or $\mathcal{G} \sqsubseteq_{\text{STAT}} \mathcal{F}$ for some non-trivial SFE functionality $\mathcal{G}$.*
3. *Every state reachable from $\mathcal{F}$'s start state via a sequence of safe transitions is a safe state.*

To prove Theorem 4, 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 4 can be expressed completely combinatorially (automata-theoretically) using Lemma 6, giving the first such alternate characterization of realizability for any large class of arbitrary reactive functionalities.

We have that (2) ⇒ (1) of Theorem 4, by the fact that $\mathcal{F}_{\text{COM}}$ is unconditionally non-trivial. We prove (3) ⇒ (2) and (1) ⇒ (3) in the following two lemmas:

Lemma 9. *If a non-simple state in $\mathcal{F}$ is reachable via a sequence of safe transitions from $\mathcal{F}$'s start state, then either $\mathcal{F}_{\text{COM}} \sqsubseteq_{\text{STAT}} \mathcal{F}$ or $\mathcal{G} \sqsubseteq_{\text{STAT}} \mathcal{F}$ for some non-trivial SFE functionality $\mathcal{G}$.*

Proof. Without loss of generality (by Lemma 8) 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 model 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 $\mathcal{G} \sqsubseteq_{\text{STAT}} \mathcal{F}$.

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 13. 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 7, 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 10. *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.*

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 $\mathcal{F} \sqsubseteq_{\text{STAT}} R(\mathcal{F})$, since $R(\mathcal{F})$ is trivial. We prove a strengthened 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 $\mathcal{F}[q] \sqsubseteq_{\text{STAT}} (R(\mathcal{F}))[q]$. 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 $\mathcal{F}[q] \sqsubseteq_{\text{STAT}} r(\mathcal{F}[q])$ (Lemma 8). 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 1)
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. □

F Extensions of the Zero-One Law

In the Public Channel Model. 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 (as is more natural in the standard UC framework). This follows from the fact that under sh-OT assumption, the private channel securely reduces to the public channel (i.e., is a trivial functionality in the public channel model). In particular, [GKM+00] prove the security of such a construction. (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 [R90], in turn implied by sh-OT assumption.)

Simultaneous Active and Passive Security. The security definition in [BMM99] considers a protocol secure only if it is simultaneously secure against active and semi-honest adversaries. Our notion of reduction, on the other hand, used a security definition which requires security only against active corruption. It is well-known that these two definitions differ (in the computationally unbounded setting). However, we can show that under such a tighter notion of reduction too, our main result holds unaltered. To show this we need to establish, under the sh-OT assumption, the triviality of those functionalities which in the unconditional setting, are trivial for security against active corruption, but non-trivial against semi-honest corruption.

We illustrate how this is done for an asymmetric SFE, in which only Bob receives an output. First, we build a protocol for the functionality which is secure against semi-honest corruption (guaranteed by sh-OT assumption). Then we "half-compile" this protocol using standalone secure commitments and zero-knowledge proofs. That is, at every step of the protocol, Alice has to prove the correctness of her steps to Bob (but not vice versa). Finally we run this half-compiled protocol in which Bob uses his real input as the input, but Alice uses a "sanitized" version of her real input; this sanitization is prescribed by the natural protocol which establishes the triviality of the functionality in the active-corruption setting. It can be shown that this protocol remains simultaneously secure against active and semi-honest corruptions. The details and extensions to non-reactive functionalities, is deferred to the final version of this paper.

G Break-Down of the Zero-One Law

G.1 Fixed-Role Reduction

Note that the $\mathcal{F}_{\text{CC}}$ functionality is not symmetric with respect to the roles of Alice and Bob. In showing that $\mathcal{F}_{\text{CC}}$ is complete, our protocol for $\mathcal{F}_{\text{COM}}$ uses ideal access to $\mathcal{F}_{\text{CC}}$ in "both directions". Let us say that a secure protocol for $\mathcal{F}_{\text{COM}}$ is a *fixed-role reduction* to $\mathcal{F}_{\text{CC}}$ if the committer (resp. receiver) always access $\mathcal{F}_{\text{CC}}$ in the same role (Alice or Bob) throughout the $\mathcal{F}_{\text{COM}}$ protocol.

Theorem 6. *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 model 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 is continued 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. □

G.2 Unbounded-Memory Functionalities

Theorem 7. *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 $\sqsubseteq_{\text{PPT}}$ reduction.

Proof. First, $\mathcal{F}$ is not trivial. This can be seen by an appeal to the splittability characterization of [PR08]. $\mathcal{F}$ 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 model. For most values of the security parameter k, access to $\mathcal{F}$ is equivalent to a plain private communication channel, since messages of length 2^{2^n} are either superpolynomially long in k (and thus can never be sent to $\mathcal{F}$), or 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. □