A Zero-One Law for Cryptographic Complexity with respect to Computational UC Security*

Hemanta K. Maji[1] Manoj Prabhakaran[1] Mike Rosulek[2]

[1] Department of Computer Science, University of Illinois, Urbana-Champaign. {hmaji2,mmp}@uiuc.edu.
[2] Department of Computer Science, University of Montana. mikero@cs.umt.edu.

Abstract. It is well-known that most cryptographic tasks do not have universally composable (UC) secure protocols, if no trusted setup is available in the framework. On the other hand, if a task like fair coin-tossing is available as a trusted setup, then *all* cryptographic tasks have UC-secure protocols. What other trusted setups allow UC-secure protocols for all tasks? More generally, given a particular setup, what tasks have UC-secure protocols?

We show that, surprisingly, every trusted setup is either useless (equivalent to having no trusted setup) or all-powerful (allows UC-secure protocols for all tasks). There are no "intermediate" trusted setups in the UC framework. We prove this **zero-one law** under a natural intractability assumption, and consider the class of deterministic, finite, 2-party functionalities as candidate trusted setups.

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. Another contribution of independent interest is to optimize the hardness assumption used by Canetti et al. (*STOC* 2002) in showing that the common random string functionality is complete (a result independently obtained by Damgård et al. (*TCC* 2010)).

1 Introduction

Cryptographic tasks provide a fascinating arena to study the interplay of information, interaction and computation. Each cryptographic task has a fundamental "information-control fingerprint" that specifies how various parties involved in the task can learn and/or influence all the pieces of information in the system. This work forms part of a study that aims to systematically understand abstract cryptographic tasks, classifying them by how "cryptographically complex" their fingerprints are.

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

A crisp way to capture the information-control fingerprint of a cryptographic task is by modeling it as a program carried out by a third party. This program is simply called a **functionality**. Modeling tasks this way conveniently separates the security definition from the information-control fingerprint.[3] Understanding and classifying such functionalities has been the subject, implicitly or explicitly, of a wealth of literature in theoretical computer science.

In this work, we continue the study of cryptographic complexity of functionalities from [29,24,25], which explicitly define cryptographic complexity classes using various notions of reductions among functionalities. Our focus in this work is on cryptographic complexity defined using security in the universal composition (UC) framework.[4] We show that under standard computational intractability assumptions, the universe of functionalities collapse to just two distinct levels of cryptographic complexity.

For simplicity, first we describe our result in terms of standard UC security terminology, and then summarize its implications for the theory of cryptographic complexity developed in [29,24,25].

Main Result. The standard UC framework defines security in a *plain* model in which protocols are allowed access only to communication channels. However, the framework also allows protocols to use access to a trusted *setup functionality*, in what is called a hybrid model. It is known that in the plain model, very few tasks admit UC-secure protocols [5,29]. On the other hand, in certain hybrid models (say, those corresponding to oblivious transfer or fair coin-tossing functionalities being used as a trusted setup) *all* tasks admit UC-secure protocols [6,16].

In this work we classify the strength of a functionality in terms of how useful it is as a trusted setup in a UC hybrid model. We prove the following surprisingly simple classification that was conjectured in [29]

[3] For instance, the commitment functionality is specified the same way for various security settings, independently of considerations in defining security; in contrast, traditional specification of commitment cannot be divorced from how binding and hiding are defined, and results in different notions (like statistically binding commitment, or statistically hiding commitment) which are not formally captured as the same functionality.

[4] While using UC security provides a fine resolution picture of cryptographic complexity, weaker security notions also yield meaningful complexity classes. For the computationally unbounded setting, semi-honest and standalone (as well as UC) security notions were considered in [24]. For the computationally bounded setting, results analogous to ours for these weaker security notions follow from the classical results in [30] and [11]. See related work.

— under a natural intractability assumption, every deterministic, finite, 2-party functionality is either:

Trivial: These functionalities already have secure protocols in the *plain* model. As such, they are useless as trusted setups; they admit no more tasks to have UC-secure protocols than in the plain model. Or,

Complete: When any of these functionalities is used as a trusted setup, *all tasks* have UC-secure protocols.[5]

We call this classification our **zero-one law**. In other words, every such functionality is at the *extremes* of usefulness: either trivial or complete. If a functionality is unrealizable in the plain model, then it is all-powerful if used as a trusted setup.

We consider UC security against probabilistic, polynomial-time adversaries who corrupt parties statically (non-adaptively). In sharp contrast, when considering the computationally unbounded setting, [24] shows that there are infinitely many setups that allow realization of infinitely many distinct classes of functionalities.

The intractability assumption we use is the existence of a protocol for oblivious transfer secure against standalone semi-honest PPT adversaries (sh-OT assumption). Interestingly, this intractability assumption is both necessary and sufficient. Then, our main result is formally stated as:

> **Main Theorem:** The sh-OT assumption is true if and only if every deterministic, finite, 2-party functionality is either trivial or complete.

The class of deterministic, finite, 2-party functionalities is defined formally in Section 3. Most notably, this class includes *reactive* functionalities, which receive inputs, give outputs, and keep internal state over many rounds of interaction with the parties. An important contribution in this work is to initiate an "automata-theoretic" study of reactive functionalities. Previous works on multi-party computation are almost exclusively restricted to SFE functionalities, except for the positive (i.e., triviality) results like in [11], which give secure realization of reactive functionalities.[6] In contrast, we develop techniques to *use* an arbitrary reactive functionality in a cryptographic protocol.

[5] Only *well-formed* functionalities are considered here, without any fairness requirement.

[6] An important exception is [29] which gives a characterization of trivial functionalities which is applicable to reactive functionalities as well; however, [29] does not offer an explicit combinatorial or automata theoretic interpretation of their characterization.

Cryptographic Complexity and Intractability. As alluded to above, one way to think about our main result is by taking a complexity-theoretic view of secure multi-party computation. Say that a functionality $\mathcal{F}$ *reduces to* another functionality $\mathcal{G}$ (written $\mathcal{F} \sqsubseteq_{\text{PPT}} \mathcal{G}$) if there is a UC-secure protocol for $\mathcal{F}$ in the $\mathcal{G}$-hybrid model. This reduction is reflexive and transitive (for standard notions of secure reductions), and is a natural complexity-theoretic reduction to compare the relative "cryptographic complexities" of cryptographic tasks. Throughout this work, we use this convenient $\sqsubseteq_{\text{PPT}}$ notation.

Under this interpretation, "completeness" (as defined above) indeed refers to $\sqsubseteq_{\text{PPT}}$-completeness in the complexity-theory sense. The zero-one law shows that there are only two *degrees* of the $\sqsubseteq_{\text{PPT}}$ reduction.[7]

While we show that the sh-OT assumption is both necessary and sufficient for the entire zero-one law, the sh-OT assumption may not be necessary for all individual reductions of the form $\mathcal{F} \sqsubseteq_{\text{PPT}} \mathcal{G}$. In a companion paper [25], we also classify which intractability assumptions are *necessary* for reductions of the form $\mathcal{F} \sqsubseteq_{\text{PPT}} \mathcal{G}$. Every reduction of the form $\mathcal{F} \sqsubseteq_{\text{PPT}} \mathcal{G}$ that we classify turns out to be unconditionally true or false, or else *exactly equivalent* to a well-known computational assumption (the sh-OT assumption or the existence of one-way functions). This suggests the possibility of *defining* intractability assumptions in terms of reductions of the form $\mathcal{F} \sqsubseteq_{\text{PPT}} \mathcal{G}$. Such assumptions are of a fundamental nature for secure multi-party computation, since they are derived directly from the definitions of functionalities themselves.

Our results in this work imply that the sh-OT assumption is the *maximal* assumption that can emerge in this framework; we conjecture that the existence of one-way functions is the *minimal* assumption. A more intriguing question is whether there are other intermediate assumptions. Put differently, one likely outcome of this line of investigation is to discover new cryptographically interesting worlds in "Impagliazzo's multiverse" [14] 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.

Related Work. There is a large body of work on complexity of 2-party functionalities in the computationally unbounded setting [17,7,23,1,18,19,20,22,24]. In the computationally bounded setting, the classical results of [30,11] imply that *all functionalities are trivial* (i.e., realizable without relying on any other functionality) *for the semi-honest and standalone security*

[7] The degree of $\mathcal{G}$ under reduction $\sqsubseteq$ is the set $\{\mathcal{F} \mid \mathcal{F} \sqsubseteq \mathcal{G}\}$.

notions respectively, under the sh-OT assumption. Our work could be considered a refinement of these early results, but for the UC security notion.

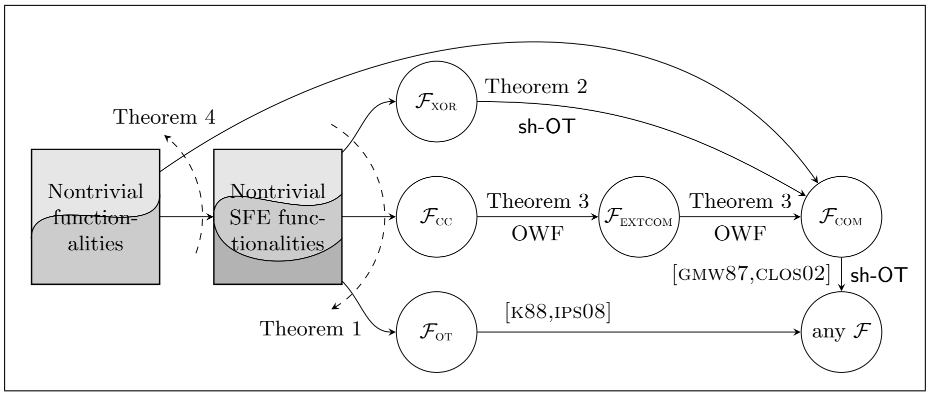
Beimel et al. [2], who showed (in the probabilistic polynomial-time setting, and for the special case of SFE functionalities in which only one party receives the output) that the sh-OT assumption is implied by the existence of a *semi-honest* secure protocol for any functionality that is not unconditionally trivial. [12] partially extends this result beyond finite functionalities, but is still restricted to the case of one-sided output. (In the full version, we show that, as in [12], there is a gap between triviality and completeness when our results are extended to unbounded-memory functionalities.)

The above results do not apply in a security setting with an arbitrary environment. Since Canetti introduced the Universal Composition (UC) framework [3], there have been several works on cryptographic complexity of functionalities in this setting. In particular, [4,5,29] characterize trivial functionalities. (For finite functionalities this class remains the same in computationally bounded and unbounded settings.)

Less was known about which functionalities were *complete* under UC-secure reductions. Results in [17,21,16] establish the completeness of oblivious transfer and many other non-reactive functionalities, for computationally unbounded adversaries. In the polynomial-time setting, the well-known CLOS construction [6] demonstrates the completness of the "coin-tossing" functionality, assuming enhanced trapdoor permutations and dense cryptosystems. Our result improves this by using the minimal sh-OT assumption, but more significantly by showing the completeness of *every* non-trivial deterministic functionality. (However, [6] proves completeness against adversaries which corrupt parties *adaptively*, whereas we consider only *static* security.) Independently of our work, Damgård et al. [8] also show the completeness of the coin-tossing functionality under the minimal sh-OT 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. Our current protocol is the result of a simplification to a protocol in an earlier draft of this work, motivated by their recent result.

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 secure protocols which demonstrate the completeness of every non-trivial functionality, proving security using only the $\mathsf{sh\text{-}OT}$ assumption. We do this in a series of steps, outlined in Figure 1.



Theorem 4
Nontrivial functionalities
Nontrivial SFE functionalities
$\mathcal{F}_{\text{XOR}}$
Theorem 2
sh-OT
$\mathcal{F}_{\text{CC}}$
Theorem 3
OWF
$\mathcal{F}_{\text{EXTCOM}}$
Theorem 3
OWF
$\mathcal{F}_{\text{COM}}$
[GMW87,CLOS02]
sh-OT
Theorem 1
$\mathcal{F}_{\text{OT}}$
[K88,IPS08]
any $\mathcal{F}$

Fig. 1. Overview of protocol constructions. An arrow from functionality $\mathcal{G}$ to $\mathcal{F}$ denotes a secure protocol for $\mathcal{F}$ using ideal access to $\mathcal{G}$ (that is, $\mathcal{F} \sqsubseteq \mathcal{G}$). Arrows not labeled by a computational assumption indicate unconditionally secure protocols.

Approach: Behavioral Components of Functionalities. Our approach of proving the zero-one law centers around identifying four distinct behaviors of functionalities that lead to non-triviality. For each behavior we associate a familiar "canonical" functionality which is non-trivial for only that reason:

– Allowing simultaneous exchange of information, exemplified by the boolean XOR functionality $\mathcal{F}_{\text{XOR}}$. In this functionality, one party's output completely determines the other's input. Thus its cryptographic non-triviality stems not from *hiding* information, but ensuring that both party's inputs are chosen independent of the other's.
– Selectively hiding one party's inputs from the other, exemplified by a simple SFE functionality we introduce called *simple cut-and-choose* ($\mathcal{F}_{\text{CC}}$). In this functionality, Alice gives a bit as input, and Bob gives an input indicating whether he wants to learn Alice's bit or not. Furthermore, Alice is told whether Bob learned her bit. Thus $\mathcal{F}_{\text{CC}}$ embodies selective hiding of Alice's input alone.
– Selectively hiding both party's inputs simultaneously, exemplified by the $\binom{2}{1}$-oblivious transfer functionality $\mathcal{F}_{\text{OT}}$. Recall that $\mathcal{F}_{\text{OT}}$ hides meaningful information about both parties from the other.
– Holding meaningful information hidden in internal memory between rounds, exemplified by the commitment functionality $\mathcal{F}_{\text{COM}}$. This func-

tionality holds the sender's data in memory between the commit phase and reveal phase. This component can appear only in a reactive functionality.

To show the zero-one law, we do the following: First, we formally define what it means for a functionality to exhibit each of these four fundamental behaviors. Next, we show that these four 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 $\mathcal{F} \in \{\mathcal{F}_{\text{XOR}}, \mathcal{F}_{\text{CC}}, \mathcal{F}_{\text{OT}}, \mathcal{F}_{\text{COM}}\}$.[8] In other words, every non-trivial functionality must exhibit at least one of the above four behaviors. Finally, we show that each of the four canonical functionalities ($\mathcal{F}_{\text{XOR}}$, $\mathcal{F}_{\text{CC}}$, $\mathcal{F}_{\text{OT}}$, $\mathcal{F}_{\text{COM}}$) is complete under the $\mathsf{sh\text{-}OT}$ assumption.

Since our definitions of these four component behaviors are all combinatorial, we are able to give the first complete *combinatorial* characterization of non-triviality (and consequently completeness) for *reactive* functionalities. Further, this characterization holds even with respect to computationally unbounded adversaries.

Non-Reactive Behaviors (Section 4). Of the four behaviors enumerated above, only the $\mathcal{F}_{\text{COM}}$ behavior is exclusive to reactive functionalities. For the other three, which can apply to non-reactive functionalities, we give formal combinatorial definitions in terms of the input/output function table. Then it suffices to show that any non-reactive functionality not meeting one of these three criteria is in fact trivial (Theorem 1).

Next, we show that $\mathcal{F}_{\text{XOR}}$, $\mathcal{F}_{\text{CC}}$, and $\mathcal{F}_{\text{OT}}$ are each complete. It is well-known that $\mathcal{F}_{\text{OT}}$ is (unconditionally) complete, even under the strong notion of reduction that we consider [17,16]. For the other two cases, we use the fact that the commitment functionality $\mathcal{F}_{\text{COM}}$ is complete in the UC framework under the $\mathsf{sh\text{-}OT}$ assumption. This follows directly from the well-known CLOS construction [6]. Thus, to complete our claim, it suffices to show that the $\mathsf{sh\text{-}OT}$ assumption implies $\mathcal{F}_{\text{COM}} \sqsubseteq_{\text{PPT}} \mathcal{F}_{\text{XOR}}$ and $\mathcal{F}_{\text{COM}} \sqsubseteq_{\text{PPT}} \mathcal{F}_{\text{CC}}$.

We give new commitment protocols in the $\mathcal{F}_{\text{XOR}}$- and $\mathcal{F}_{\text{CC}}$-hybrid models (Theorems 2 and 3), secure under the $\mathsf{sh\text{-}OT}$ and OWF assumptions, respectively. We note that [6] shows (implicitly) that $\mathcal{F}_{\text{XOR}}$ is complete;[9]

[8] 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 we prove the complete of each one differently.

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

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 new protocol and under the minimal sh-OT assumption.

Reactive Behaviors (Section 5). To complete the classification of reactive functionalities, we show that every reactive functionality is either trivial, contains a non-reactive behavioral component ($\mathcal{F}_{\text{XOR}}$, $\mathcal{F}_{\text{CC}}$, $\mathcal{F}_{\text{OT}}$), or else can be used for a commitment ($\mathcal{F}_{\text{COM}}$) protocol (Theorem 4). As mentioned above, $\mathcal{F}_{\text{COM}}$ is complete under the sh-OT assumption, thus we establish the exhaustiveness of the four behavioral components, as well as the completeness of their respective canonical functionalities.

The bulk of our technical contributions for reactive functionalities involves formally defining this fourth behavioral component; namely, defining when an arbitrary functionality keeps meaningful information about a party's input hidden in memory between rounds. We model reactive functionalities as finite-state automata, and initiate an automata-theoretic analysis of their input/output behavior. This classification involves identifying states and transitions of an automaton which have specific cryptographic consequences, and then showing how such features can be leveraged to give a protocol for $\mathcal{F}_{\text{COM}}$.

3 Preliminaries

Model and Security Definition. Our security definitions are grounded in the framework of Universal Composable (UC) security [3], with which we assume the reader has slight familiarity. For concreteness we consider the model used in [29], which in turn is based on that in [28]. 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 note that the functionalities we consider 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 [3] or [28] 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.[10]

By default, we also allow protocols access to a communication channel. Following [29], 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.

Classes of functionalities. In this work we restrict our attention to **finite, deterministic, 2-party, reactive** functionalities. We formally model such 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. 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'. 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). We consider an asynchronous network setting in which the adversary has control over the timing of input/output delivery. In Figure 2 we give an example of how the (reactive) commitment functionality $\mathcal{F}_{\text{COM}}$ can be expressed in such a way.

We say that a functionality is a **secure function evaluation** (SFE; or non-reactive) functionality if it engages only one round of interaction; that is, all transitions leading from the start state lead to a dead state with

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

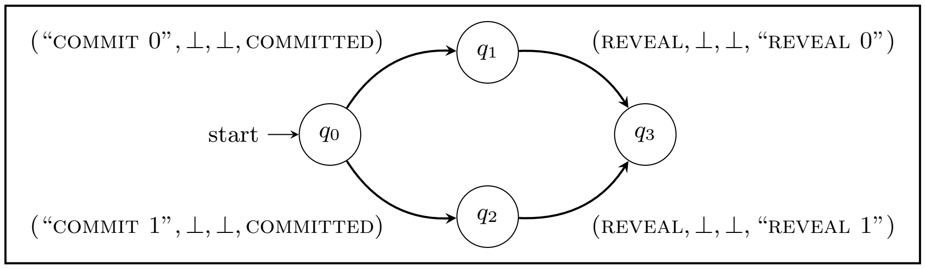

("COMMIT 0", ⊥, ⊥, COMMITTED)
q_1
(REVEAL, ⊥, ⊥, "REVEAL 0")
start
q_0
q_3
("COMMIT 1", ⊥, ⊥, COMMITTED)
q_2
(REVEAL, ⊥, ⊥, "REVEAL 1")

Fig. 2. Commitment functionality $\mathcal{F}_{\text{COM}}$ modeled as a deterministic finite functionality

no transitions. Alternatively, an SFE functionality is completely specified by a pair of functions (f_A, f_B), where Alice's output is $f_A(x,y)$ and Bob's output is $f_B(x,y)$.

The sh-OT *assumption.* The primary intractability 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 [26,13,9]. One-way functions are in turn implied by the sh-OT assumption [15].

4 Zero-One Law for Non-Reactive Functionalities

Three "Canonical" Non-Reactive Functionalities. The following three SFE functionalities exemplify the three different behaviors that lead to cryptographic non-triviality for non-reactive functionalities:

$\mathcal{F}_{\text{XOR}}$ (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. Thus, Bob decides whether to learn Alice's bit, while Alice always 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 output $\perp$.

We show that these three fundamental behaviors completely characterize non-triviality (for non-reactive functionalities), 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). ($\Leftarrow$) Each of $\mathcal{F}_{\text{XOR}}$, $\mathcal{F}_{\text{CC}}$, and $\mathcal{F}_{\text{OT}}$ is unconditionally non-trivial, from the characterization of trivial SFE functionalities in [29].

($\Rightarrow$) Kraschewski and Müller-Quade [21] 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 $\mathcal{F}$ can be used to construct an unconditionally UC-secure protocol for $\mathcal{F}_{\text{OT}}$ if and only if $\mathcal{F}$ contains such a minor.

Similarly, we also identify another important 2×2 minor called a *generalized-CC minor.* A minor a generalized-CC if, when restricted to the minor, one party can choose whether to learn the other's input, and this choice is revealed to the other party 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 $\mathcal{F}_{\text{CC}}$.[11]

Finally, it is easy to see that if $\mathcal{F}$ does not have either kind of 2×2 minor mentioned above, then $\mathcal{F}$ 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 protocol for $\mathcal{F}$ (without any set-up).

Completeness of the Three Canonical Non-Reactive Functionalities. Since $\mathcal{F}_{\text{OT}}$ is unconditionally complete (even with respect to UC secure protocols) [17,16], and the commitment functionality $\mathcal{F}_{\text{COM}}$ is complete under the sh-OT assumption [6], it suffices to prove the following two theorems:

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

[11] 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.

Proof (Sketch). We first observe that the coin-tossing functionality $\mathcal{F}_{\text{COIN}}$ [12] 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 construction [6] proves exactly this; however, their focus was on achieving *adaptive* security, and their protocol relied on a stronger computational assumption than the $\mathsf{sh\text{-}OT}$ assumption. Thus we must use an entirely different approach for achieving $\mathcal{F}_{\text{COM}}$ (with static security) from $\mathcal{F}_{\text{COIN}}$. We sketch an overview of our protocol below:

Suppose ψ_{sh} is the semi-honest protocol for $\mathcal{F}_{\text{OT}}$ guaranteed by the $\mathsf{sh\text{-}OT}$ 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 [11]. 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 $x_1 = b$.

In the real interaction, σ is generated honestly using $\mathcal{F}_{\text{COIN}}$ and is therefore in the pseudorandom distribution with negligible probability. Thus the GMW-style compilation ensures that both parties are executing the ψ_{sh} subprotocol honestly as stated. Then applying the semi-honest security of ψ_{sh}, we see that 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 play the role of Bob using input $y = 1$ to the ψ_{sh} subprotocol, while still giving convincing GMW proofs. By the correctness of the ψ_{sh} protocol, the simulated Bob obtains $x_1 = b$ from ψ_{sh} (i.e., the simulator *extracts* b), and

[12] $\mathcal{F}_{\text{COIN}}$ is a functionality which, upon activation, samples an unbiased coin $b \leftarrow \{0,1\}$ and outputs it to both parties. It does not fall in our class of deterministic finite functionalities, but we use it as an intermediate step in our protocol construction.

by the security of ψ_{sh}, 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}. For technical reasons in this part, we require the interactive proofs to be proofs *of knowledge.*

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

Proof (Sketch). 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 then 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 $\mathcal{F}_{\mathrm{EXTCOM}}$. The complete formulation of $\mathcal{F}_{\mathrm{EXTCOM}}$ is highly nontrivial, and is defered to the full version. $\mathcal{F}_{\mathrm{EXTCOM}}$ 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 [27], we show that if one-way functions exist, then $\mathcal{F}_{\mathrm{COM}} \sqsubseteq_{\mathrm{PPT}} \mathcal{F}_{\mathrm{EXTCOM}}$.

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 whether Bob choses to see her input. 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.

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 could not supply too many incorrect values 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 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 condition (2) 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 its external behavior) and *meaningful* (may eventually influence its future external behavior). Such usage of internal memory is exemplified by the commitment functionality $\mathcal{F}_{\text{COM}}$ (between the commit and reveal phases).

Automata-theoretic Characterization. We develop three new important properties of reactive functionalities, all defined combinatorially.

Say that an input $\hat{x}$ *dominates* another input x if 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. 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 condition of $\hat{x}$ dominating x can be defined directly in terms of the UC security condition.

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. In an SFE, Bob's output may reveal information about Alice's output, but up to $\sim$-equivalence at most. However, in a *trivial* SFE, Bob's output always reveals exactly the $\sim$-equivalence class of Alice's input. We say that the start state of $\mathcal{F}$ is *simple* if: (1) its associated SFE is a trivial SFE, and (2) each equivalence class of $\sim$ (for Alice inputs and Bob inputs) contains some input that dominates all other inputs in its class.

To understand this definition, suppose the start state of $\mathcal{F}$ is simple. Then just by looking at his own output from the first round, Bob can exactly determine the $\sim$-class of the input Alice used. There is some input, say, $\hat{x}$, which dominates all Alice inputs in this $\sim$-class. No matter how the environment instructs Alice to behave in the future, she could have achieved the same effect if she had used input $\hat{x}$ in the first round. Thus, Bob can assume without loss of generality that Alice in fact used $\hat{x}$. The same is true for Alice; she can determine, given her output, an input $\hat{y}$ for Bob, and assume without loss of generality that Bob supplied $\hat{y}$ in the first round.

Thus we can assume without loss of generality that Alice and Bob only use inputs $\hat{x}$ and $\hat{y}$ of this special kind (they dominate their respective $\sim$-equivalence classes). We call the transition from the start state on such inputs $(\hat{x}, \hat{y})$ a *safe transition*. Intuitively, only safe transitions are relevant; furthermore, after a safe transition, neither party has uncertainty about the functionality's resulting state.

We can now state our main automata-theoretic characterization:

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. *There is a non-simple state in $\mathcal{F}$ that is reachable from the start state via a sequence of safe transitions.*

The automata-theoretic properties defined above, and subsequently 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). $2 \Rightarrow 1$ follows from the non-triviality of $\mathcal{F}_{\text{COM}}$.

($1 \Rightarrow 3$) Consider all the states of $\mathcal{F}$ reachable via a sequence of safe transitions; intuitively, these are the only states that matter. If all such states are simple, then $\mathcal{F}$ has the following trivial protocol: repeatedly evaluate the (trivial) SFE corresponding to $\mathcal{F}$'s current state, using that SFE's trivial protocol. Without loss of generality we can assume a safe transition was taken; thus, each party's output in the round determines the next state of $\mathcal{F}$, and the protocol can be repeated for each round.

($3 \Rightarrow 2$) Assume that one of the safely reachable states of $\mathcal{F}$ is non-simple; without loss of generality, we can take the start state to be non-simple. The definition of a simple state requires two conditions, so we consider two cases: (1) 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 (simply interact with $\mathcal{F}$ for one round only). (2) Otherwise, the start state is non-simple because there exist two inputs for (by symmetry) Alice, say x_0 and x_1, for which $x_0 \sim x_1$ (that is, these inputs always induce the same output for Bob in the first round), but no Alice input dominates *both* of $\{x_0, x_1\}$. In other words, Alice's first-round input "binds" 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 a protocol for $\mathcal{F}_{\text{COM}}$, as follows. Alice commits to b by sending x_b to $\mathcal{F}$ in the first round. The commitment is perfectly hiding since $x_0 \sim x_1$. To reveal, Alice must 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.

Suppose x does not dominate x'. Then for every strategy for Alice which uses input x in the first round, there is some environment that can distinguish between Alice's strategy and one which uses input x' in the first round and thereafter runs the dummy protocol. Using an automata-theoretic characterization, we show that these quantifiers can be exchanged: there is a fixed environment such that for every x not dominating x' and every Alice strategy that uses input x in the first round, the environment has a constant probability of "catching" Alice.[13] Our commitment protocol instructs Bob to play the role of such an environment in

[13] This environment results in a protocol for $\mathcal{F}_{\text{COM}}$ whose worst-case $O(k)$ efficiency hides very large constants. However, it is usually possible to tailor such a distin-

the reveal phase, sending a sequence inputs to $\mathcal{F}$ himself and a sequence of "challenge" 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. If Alice's first-round input did not dominate x_b, she is guaranteed to be caught with constant probability. By repeating this basic protocol in parallel an appropriate number of times, Bob can be assured of catching an equivocating Alice with overwhelming probability.

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 to hold.

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

Proof. If the zero-one law holds, then $\mathcal{F}_{\text{XOR}}$ is complete, since it is unconditionally non-trivial. Thus $\mathcal{F}_{\text{OT}} \sqsubseteq_{\text{PPT}} \mathcal{F}_{\text{XOR}}$. $\mathcal{F}_{\text{OT}}$ has the property that any protocol that securely realizes $\mathcal{F}_{\text{OT}}$ (against active adversaries) is also secure against semi-honest adversaries (see [29] 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.

More generally, if $\mathcal{F}$ has an unconditionally secure protocol against semi-honest adversaries, then the $\sqsubseteq_{\text{PPT}}$-completeness of $\mathcal{F}$ implies the sh-OT assumption.

7 Extensions, Limitations, and Open Problems

We discuss several natural extensions of our main theorem, discussed in greater detail in the full version:

Strengthening the Reduction. As the definition of a reduction is strengthened, fewer functionalities reduce to one another. In the extreme, the reduction could be made so restrictive that no functionality reduces to another. On the other hand, it is relatively easy to see that the zero-one

guishing environment for a particular $\mathcal{F}$ to achieve much better efficiency bounds, resulting in a very practical commitment protocol.

law still applies as stated in this work if protocols are given only public channels instead of private channels, or if security is simultaneously required against both active and semi-honest adversaries.

If the reduction requires security against *computationally unbounded* adversaries, then the zero-one law breaks down. In fact, there exist infinitely many distinct intermediate (between trivial and complete) complexities with respect to this stronger reduction [24].

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 $\mathcal{F}_{\text{COM}} \not\sqsubseteq \mathcal{F}_{\text{CC}}$ (note that the behavior of $\mathcal{F}_{\text{CC}}$ is not symmetric with respect to the two parties), so the zero-one law does not hold 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 $\mathcal{F}_{\text{CC}}$.

We leave open the question of whether the zero-one law holds if the reduction is strengthened to require security against *adaptive* corruption.

Larger Classes of Functionalities. We restricted our attention to a class of deterministic functionalities with finite memory and inputs. In fact, the zero-one law does not extend if we relax the restriction on finiteness. 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, one can construct an f so that the resulting functionality is neither trivial nor complete.[14] The construction of this intermediate $\mathcal{F}$ is admittedly contrived, and we leave open the important problem of identifying the largest "natural" class of unbounded-memory functionalities that still satisfies the zero-one complexity 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 [23,1].

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 Theorems 2 and 3 we use interactive proofs of statements regarding various cryptographic primitives (ultimately derived from the sh-OT assumption). We do not know whether such non-black-box usage

[14] Of course, if one-way functions do *not* exist, then the sh-OT assumption, and subsequently the zero-one law, is again false.

of the assumption is necessary, although it seems that a fundamentally different approach is required to avoid the use of interactive proofs.

Acknowledgments

We acknowledge helpful discussions with Ran Canetti, Yuval Ishai, Yehuda Lindell and Amit Sahai, as well as helpful suggestions from anonymous conference referees. The protocol in Theorem 2 was simplified from its original form in an earlier manuscript, partly motivated by the recent results of [8].

References

1. D. Beaver. Perfect privacy for two-party protocols. In J. Feigenbaum and M. Merritt, editors, *Proceedings of DIMACS Workshop on Distributed Computing and Cryptography*, volume 2, pages 65–77. American Mathematical Society, 1989.
2. A. Beimel, T. Malkin, and S. Micali. The all-or-nothing nature of two-party secure computation. In M. J. Wiener, editor, *CRYPTO*, volume 1666 of *Lecture Notes in Computer Science*, pages 80–97. Springer, 1999.
3. R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. Electronic Colloquium on Computational Complexity (ECCC) TR01-016, 2001. Previous version "A unified framework for analyzing security of protocols" availabe at the ECCC archive TR01-016. Extended abstract in FOCS 2001.
4. R. Canetti and M. Fischlin. Universally composable commitments. Report 2001/055, Cryptology ePrint Archive, July 2001. Extended abstract appeared in CRYPTO 2001.
5. R. Canetti, E. Kushilevitz, and Y. Lindell. On the limitations of universally composable two-party computation without set-up assumptions. In E. Biham, editor, *EUROCRYPT*, volume 2656 of *Lecture Notes in Computer Science*. Springer, 2003.
6. R. Canetti, Y. Lindell, R. Ostrovsky, and A. Sahai. Universally composable two-party computation. In *Proc. 34th STOC*, pages 494–503. ACM, 2002.
7. B. Chor and E. Kushilevitz. A zero-one law for boolean privacy (extended abstract). In *STOC*, pages 62–72. ACM, 1989.
8. I. Damgård, J. B. Nielsen, and C. Orlandi. On the necessary and sufficient assumptions for UC computation. In D. Micciancio, editor, *TCC*, volume 5978 of *Lecture Notes in Computer Science*, pages 109–127. Springer, 2010.
9. O. Goldreich. *Foundations of Cryptography: Basic Tools.* Cambridge University Press, 2001. Earlier version available on `http://www.wisdom.weizmann.ac.il/~oded/frag.html` .
10. O. Goldreich. *Foundations of Cryptography: Basic Applications.* Cambridge University Press, 2004.
11. O. Goldreich, S. Micali, and A. Wigderson. How to play ANY mental game. In ACM, editor, *Proc. 19th STOC*, pages 218–229. ACM, 1987. See [10, Chap. 7] for more details.
12. D. Harnik, M. Naor, O. Reingold, and A. Rosen. Completeness in two-party secure computation: A computational view. *J. Cryptology*, 19(4):521–552, 2006.

13. J. Håstad, R. Impagliazzo, L. A. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999. Preliminary versions appeared in STOC' 89 and STOC' 90.
14. R. Impagliazzo. A personal view of average-case complexity. In *Structure in Complexity Theory Conference*, pages 134–147, 1995.
15. R. Impagliazzo and M. Luby. One-way functions are essential for complexity based cryptography (extended abstract). In *Proc. 30th FOCS*, pages 230–235. IEEE, 1989.
16. Y. Ishai, M. Prabhakaran, and A. Sahai. Founding cryptography on oblivious transfer - efficiently. In D. Wagner, editor, *CRYPTO*, volume 5157 of *Lecture Notes in Computer Science*, pages 572–591. Springer, 2008.
17. J. Kilian. Founding cryptography on oblivious transfer. In *STOC*, pages 20–31. ACM, 1988.
18. J. Kilian. A general completeness theorem for two-party games. In *STOC*, pages 553–560. ACM, 1991.
19. J. Kilian. More general completeness theorems for secure two-party computation. In *Proc. 32th STOC*, pages 316–324. ACM, 2000.
20. J. Kilian, E. Kushilevitz, S. Micali, and R. Ostrovsky. Reducibility and completeness in private computations. *SIAM J. Comput.*, 29(4):1189–1208, 2000.
21. D. Kraschewski and J. Müller-Quade. Completeness theorems with constructive proofs for symmetric, asymmetric and general 2-party-functions, 2008. Unpublished Manuscript, 2008. `http://iks.ira.uka.de/eiss/completeness`.
22. R. Künzler, J. Müller-Quade, and D. Raub. Secure computability of functions in the it setting with dishonest majority and applications to long-term security, 2009.
23. E. Kushilevitz. Privacy and communication complexity. In *FOCS*, pages 416–421. IEEE, 1989.
24. H. K. Maji, M. Prabhakaran, and M. Rosulek. Complexity of multi-party computation problems: The case of 2-party symmetric secure function evaluation. In O. Reingold, editor, *TCC*, volume 5444 of *Lecture Notes in Computer Science*, pages 256–273. Springer, 2009.
25. H. K. Maji, M. Prabhakaran, and M. Rosulek. Cryptographic complexity classes and computational intractability assumptions. In A. C.-C. Yao, editor, *Innovations in Computer Science*, pages 266–289. Tsinghua University Press, 2010.
26. M. Naor. Bit commitment using pseudorandomness. 4(2):151–158, 1991. Preliminary version in CRYPTO' 89.
27. M.-H. Nguyen and S. P. Vadhan. Zero knowledge with efficient provers. In *STOC*, pages 287–295. ACM, 2006.
28. M. Prabhakaran. *New Notions of Security.* PhD thesis, Department of Computer Science, Princeton University, 2005.
29. M. Prabhakaran and M. Rosulek. Cryptographic complexity of multi-party computation problems: Classifications and separations. In D. Wagner, editor, *CRYPTO*, volume 5157 of *Lecture Notes in Computer Science*, pages 262–279. Springer, 2008.
30. A. C. Yao. How to generate and exchange secrets. In *Proc. 27th FOCS*, pages 162–167. IEEE, 1986.