Exploring the Limits of Common Coins Using Frontier Analysis of Protocols

Hemanta K. Maji[1,⋆], Pichayoot Ouppaphan[1,⋆⋆], Manoj Prabhakaran[1,⋆], and Mike Rosulek[2]

[1] Department of Computer Science, University of Illinois, Urbana-Champaign. {hmaji2,pouppap2,mmp}@uiuc.edu.
[2] Department of Computer Science, University of Montana. mikero@cs.umt.edu.

Abstract. In 2-party secure computation, access to common, trusted randomness is a fundamental primitive. It is widely employed in the setting of computationally bounded players (under various complexity assumptions) to great advantage. In this work we seek to understand the power of trusted randomness, primarily in the computationally unbounded (or information theoretic) setting. We show that a source of common randomness does not add any additional power for secure evaluation of deterministic functions, even when one of the parties has arbitrary influence over the distribution of the common randomness. Further, common randomness helps only in a trivial sense for realizing randomized functions too (namely, it only allows for sampling from publicly fixed distributions), if UC security is required.

To obtain these impossibility results, we employ a recently developed protocol analysis technique, which we call the *frontier analysis*. This involves analyzing carefully defined "frontiers" in a weighted tree induced by the protocol's execution (or executions, with various inputs), and establishing various properties regarding one or more such frontiers. We demonstrate the versatility of this technique by employing carefully chosen frontiers to derive the different results. To analyze randomized functionalities we introduce a frontier argument that involves a geometric analysis of the space of probability distributions.

Finally, we relate our results to computational intractability questions. We give an equivalent formulation of the "cryptomania assumption" (that there is a semi-honest or standalone secure oblivious transfer protocol) in terms of UC-secure reduction among randomized functionalities. Also, we provide an *unconditional result* on the uselessness of common randomness, even in the computationally bounded setting.

Our results make significant progress towards understanding the exact power of shared randomness in cryptography. To the best of our knowledge, our results are the first to comprehensively characterize the power of large classes of randomized functionalities.

⋆ Partially supported by NSF grants CNS 07-47027 and CNS 07-16626.
⋆⋆ Supported by NSF grant CNS 0851957 for undergraduate research.

1 Introduction

In this work, we consider a fundamental question: *How cryptographically useful is a trusted source of public coins?*

While there are several instances in cryptography where a common random string or a trusted source of public coins is very useful (e.g. [3,5]), we show severe limitations to its usefulness[3] in secure two-party computation, without — and sometimes even with — computational intractability assumptions. In contrast, it is well known that more general correlated private random variables can be extremely powerful [2]. Given that for semi-honest security common randomness is useless (as one of the parties could sample and broadcast it), it is not surprising that it should turn out to be not as powerful as general correlated random variables. However, despite its fundamental nature, the exact power of common randomness has not yet been characterized. Here, we provide tight characterizations of what can be achieved with a source of common randomness, in various settings of 2-party computation. We show:

- For two-party secure function evaluation (SFE) of *deterministic* functions, being given a source of common randomness is useless, irrespective of any computational complexity assumptions, when considering security in the standalone setting.[4]
- Clearly a source of common randomness can be useful for realizing *randomized* functionalities. However, in the case of UC security, we show that a source of common coins can be useful only in a trivial sense (unless restricted to the computationally bounded setting, and intractability assumptions are employed). We show that any UC-secure protocol that uses common coins for evaluating a randomized function can be replaced by a protocol of the following simple form: one of the parties announces a probability distribution, based deterministically on its input, and then the two parties sample an outcome from this distribution using freshly sampled common coins. We call the resulting functionality a *publicly-selectable source.*
- We relate computational intractability assumptions to secure reductions among randomized functionalities, giving evidence that common randomness is useful only under strong computational assumptions. In particular we show that common randomness can be used to UC-securely realize a symmetric functionality with bi-directional influence (i.e., the output is influenced by both the parties' inputs) if and only if there exists a semi-honest secure protocol for oblivious transfer.

These results are actually proven for a class of sources more general than coin tossing, namely *selectable sources* – that let one of the parties (secretly) specify

[3] We say that a source of common randomness is useless in realizing some 2-party functionality $\mathcal{F}$ if either $\mathcal{F}$ could be realized without using the given source or $\mathcal{F}$ cannot be realized even given the source. Note that we consider only the feasibility question and not any efficiency issues.

[4] In the case of UC security, it follows from the results in [16] that a source of common randomness is useless except in Cryptomania, where it is a complete functionality.

which among a set of distributions should be used by the source. We highlight two aspects of these results:

Non-blackbox analysis of protocols. In deriving the impossibility results our analysis crucially relies on the communication and information structure of protocols. We build on the "frontier analysis" paradigm in [8,15,16], but significantly extend its power, among other things, to enable analyzing protocols for arbitrary randomized functionalities, and protocols using randomized functionalities.

These results (and hence proofs) are necessarily of a *non-relativizing* nature — if the protocol has access to another trusted functionality (more sophisticated than common randomness), the impossibility results no longer hold. Specifics about the common randomness functionality are (and must be) used in our proofs. Such low-level analysis of protocols, we believe, is crucial to understanding the power and complexity of multi-party computation primitives.

Understanding randomized functionalities. Secure evaluation of *randomized* functions has in general been a poorly understood area. In particular, to date it remains open to characterize which randomized functions can be securely realized even against computationally unbounded passive (honest-but-curious) adversaries — a problem that was solved for deterministic functions twenty years ago [1,13]. Much of the study of randomized functionalities has been focused on in-depth understanding of the simplest such functionality — namely generating shared fair coins (e.g., see [7,10,8,18] and references therein). Our results provide significant insight into *other* randomized functionalities as well, and their connections to computational intractability assumptions. In particular, our results involve two interesting classes of randomized functionalities, namely *selectable sources* and *publicly-selectable sources*.

1.1 Overview

Frontier analysis. The bulk of our results take the form of statements of cryptographic impossibility. That is, we show that a protocol for a given cryptographic task is impossible (or else implies a certain computational primitive like one-way functions). Such impossibility results have been a core challenge in cryptography. In this work, we present a powerful battery of techniques that we use to analyze 2-party protocols, which we broadly call "frontier analysis."

The basic outline of a frontier analysis is as follows. We first interpret a protocol as a tree of possible transcripts, with weights corresponding to the probability that the protocol assigns to each message, based on the parties' inputs. Within this tree, we identify "frontiers", which are simply a collection of nodes (partial transcripts) that form a cut and an independent set. Intuitively, these frontiers correspond to points in the protocol when some condition is satisfied for the first time, where the condition in question depends on the kind of analysis needed: for example, the first place the transcript leaks "significant" information about a party's input, or the first place that common coins have made a "significant" influence on the protocol's output.

Impossibility proofs using frontier analysis proceed by showing that frontiers of certain kind exist, often showing that multiple frontiers must be encountered in a specific order, and then showing that an adversary can effect an attack by exploiting the properties of these frontiers. The interested reader can find a high level discussion on frontier analysis as a tool for protocol analysis in the full version of this paper [14].

Common coins are not useful in SFE protocols. We show that against computationally unbounded adversaries (more precisely, against adversaries that can break one-way functions), any 2-party deterministic SFE (in which both parties receive the same output) functionality that can be securely realized given a trusted coin-tossing functionality can in fact be securely realized without it. This is most interesting for the standalone setting, because if one-way functions do exist then a standalone-secure coin-tossing protocols exist, so again access to a trusted coin-tossing functionality is redundant.[5]

We start off by showing that there is no secure protocol for evaluating boolean XOR given a coin-tossing functionality. In many ways these functionalities have similar "complexity" (in particular, neither is complete, and both are trivial to realize against passive adversaries), so establishing a qualitative separation between them is interesting in itself. In a protocol for XOR, either party may be the first to reveal information about their input, and the two parties can even gradually reveal more and more information about their input in an interleaved fashion. We define a frontier corresponding to the first point at which some party has revealed "significant" information about its input. Then we define an attack that can be carried out when the protocol crosses this frontier. Since a large class of SFE functionalities can be used to securely realize XOR, the impossibility extends to these functionalities as well.

We then use the combinatorial characterizations of Symmetric Secure Function Evaluation (SSFE) functionalities (obtained using frontier analysis) from [15] to extend the result to arbitrary SSFE functionalities (instead of just XOR). Further, using an extension of a result in [11], we extend this to arbitrary SFE functionalities by associating a symmetric SFE with every general SFE that has a secure protocol using a source of common randomness.

For randomized SFE, common coins help only in a trivial sense. We show that common coins are useful in constructing UC-secure protocols for randomized SFE functionalities only for the class of publicly-selectable sources (Theorem 2). For this result, we exploit the versatility of the frontier analysis and also employ a geometric analysis of the space of effective probability distributions.

[5] A recent result in [16] gives a sharp result for the case of UC security: the coin-tossing functionality is useful in realizing further deterministic SFE functionalities if and only if there exists a semi-honest oblivious transfer protocol. However neither the result nor the approach in [16] extends to the standalone setting. Also, our result is applicable to not just symmetric functionalities and coin-tossing, but extends to general SFE functionalities and all selectable sources.

The frontier analysis is carried out for an SSFE functionality, and then the result is extended to general SFE functionality separately. For a randomized SSFE functionality, for each pair of inputs, the output is specified by a distribution (over a finite output alphabet). This distribution can be represented as a vector in d-dimensional real space where d is the size of the output alphabet. By considering all possible inputs, we obtain a set of points in this space as legitimate output distributions. But since the parties can choose their input according to any distribution they wish, the entire convex hull of these points is the set of legitimate output distributions. Note that the vertices of this polytope correspond to the output distributions for various specific input choices.

In analyzing a protocol for such a functionality, we define *two* very different frontiers: one intuitively captures the last point in the protocol where the parties' inputs have any noticeable influence over the output distribution. The other intuitively captures the first point where the common coins have had a non-trivial influence on the output distribution.

Defining these frontiers is a delicate task, but once they are defined, we can show that, for the protocol to be UC-secure, the two frontiers must be encountered in the order listed above. Thus there is always a point within the protocol where the parties' inputs have stopped influencing the output, yet the public coins have not yet started influencing the output in a non-trivial way. At this point, we can show that the output distribution is uniquely determined, and that the subsequent coins are simply used to sample from this publicly-chosen distribution.

Then, on each node in the first frontier the conditional output distribution is still within the polytope. On the other hand, since the input influence has ceased at this point, for any fixed input, its output distribution must be determined by this frontier: i.e., it must be a convex combination of the conditional output distributions at the nodes on the frontier. That is, the output distribution for this input is a convex combination of conditional output distributions which are all themselves within the polytope. Now, (without loss of generality, as it turns out) we can consider inputs whose output distributions are vertices of the polytope. Then, *for all nodes in the frontier* the conditional output distribution must coincide with the final distribution itself. Thus on reaching this frontier in the protocol, the output distribution is revealed (as a deterministic function of the inputs) and the rest of the protocol simply samples from this distribution.

Finally, we extend this result also to general SFE (instead of just symmetric SFE) functionalities, in the same way as for deterministic functionalities.

Selectable sources. Selectable sources are an interesting class of randomized functionalities with an intermediate level of complexity: they can be more complex than a (fixed) source of common randomness, yet they are simple enough that we can show that they are as useless as common randomness when it comes to securely realizing deterministic functionalities. The extension is observed by following the analysis for the case of the source of common randomness, and identifying the properties that it relies on. We do not know at this point whether

these are exactly all the functionalities which are useless for realizing SFE functionalities, but based on our understanding so far, we conjecture that they are.

Connections to Computational Intractability. Finally, we relate our results to computational intractability questions. The attacks based on frontier analysis can often be extended to the computationally bounded setting, if one-way functions do not exist (as was pointed out in [16]). We show that this is indeed the case for our attacks. In fact, our first application of such an extension is to obtain an *unconditional* result about the uselessness of selectable sources in realizing deterministic secure function evaluation with standalone security. For this we use the fact that if one-way functions do not exist we can attack any given protocol, whereas if one-way functions do exist then we can realize any selectable source functionality (with standalone security) and then again they are useless.

We also generalize a result in [16] to the setting of randomized functionalities. There it was shown that if any non-trivial deterministic functionality is UC-securely realizable using access to common randomness, then there exists an oblivious transfer protocol secure against semi-honest adversaries. We generalize common randomness to any selectable source, and also generalize non-trivial deterministic functionalities to randomized SSFE functionalities with both parties' inputs having an influence on the output.

Related Results. Frontier analysis is possibly implicit in previous works on proving impossibility or lower bounds for protocols. For instance, the analysis in [8] very well fits our notion of what frontier analysis is. The analysis of protocols in [6,1,13] also have some elements of a frontier analysis, but of a rudimentary form which was sufficient for analysis of perfect security. In [15] frontier analysis was explicitly introduced and used to prove several protocol impossibility results and characterizations. [12] also presented similar results and used somewhat similar techniques (but relied on analyzing the protocol by rounds, instead of frontiers, and suffered limitations on the round complexity of the protocols for which the impossibility could be shown).

2 Preliminaries

We say that a function $\nu : \mathbb{N} \to \mathbb{R}$ is *negligible* if for every polynomial p, $\nu(k) < 1/p(k)$ for sufficiently large k. If $\mathcal{D}, \mathcal{D}'$ are discrete probability distributions with support S, we write $\mathrm{SD}(\mathcal{D}, \mathcal{D}')$ to denote the statistical distance of the distributions, defined as $\mathrm{SD}(\mathcal{D}, \mathcal{D}') = \frac{1}{2} \sum_{s \in S} |\mathcal{D}(s) - \mathcal{D}'(s)|$.

Security. We use standard conventions and terminology for the security of protocols for multi-party computation tasks. A protocol is secure if for every adversary in the real world (in which parties execute a protocol), there is an adversary, or *simulator*, in the ideal world (in which the task is carried out on behalf of the parties by a trusted third party called a *functionality*) that achieves the same effect. A *semi-honest* or *passive* adversary is one which is not allowed to deviate

from the protocol. *Standalone* security is achieved if the simulator is allowed to rewind the adversary; *Universally composable (UC)* security [4] is achieved if the simulation is straight-line (i.e., never rewinds the adversary). In this work, we exclusively consider *static* adversaries, who do not adaptively corrupt honest parties during the execution of a protocol.

The *plain model* is a real world in which protocols only have access to a simple communication channel; a *hybrid model* is a real world in which protocols can additionally use a particular trusted functionality. While hybrid worlds are usually considered only for UC security, we also use the terminology in the setting of standalone security. We note that protocols for *non-reactive* functionalities (i.e., those which receive input from all parties, then give output, and then stop responding) do securely compose even in the standalone security setting.

2.1 Functionalities

We focus on classifying several important subclasses of functionalities.

Secure function evaluation (SFE). A 2-party secure function evaluation (SFE) functionality is specified by two functions $f_1 : X \times Y \to Z$ and $f_2 : X \times Y \to Z$, where X and Y are finite sets. The functionality waits for input $x \in X$ from Alice and $y \in Y$ from Bob, then delivers $f_1(x, y)$ and $f_2(x, y)$ to them, respectively. There is no fairness guarantee: if a party is corrupt, it can obtain its own output first and decide whether the output should be delivered to the other party.

If $f_1 = f_2$ are identical we call it a *symmetric* SFE (or SSFE) functionality. SSFE functionalities are the most fundamental, and have been studied since Yao first introduced the concept of multi-party computation [20]. We can specify an SSFE function by simply giving its function table, where the rows correspond to an input of Alice, and columns correspond to an input of Bob. For instance, the XOR functionality has function table $\left[\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right]$.

Randomized functionalities. A randomized SFE functionality is specified by functions $f_1, f_2 : X \times Y \times R \to Z$. The functionality takes inputs $x \in X$ from Alice, $y \in Y$ from Bob, uniformly samples $r \in R$ and outputs $f_1(x, y, r)$ and $f_2(x, y, r)$ to Alice and Bob, respectively. An important example is the common randomness functionality, denoted by $\mathcal{F}_{\mathsf{coin}}$ (with $X = Y = \{0\}$, $R = \{0, 1\}$, and $f_1(x, y, r) = f_2(x, y, r) = r$). Note that for a given pair of inputs, the outputs to Alice and Bob could be correlated as the same value r is used in both.

We identify two important subclasses of randomized SSFE functionalities:

Selectable sources: One in which one party's input does not affect the output. That is, functions which can be written as $f(x, y, r) = h(x, r)$ for some function h. Note that for different values of x, the function's output distribution may be arbitrary.

Publicly-selectable sources: Those functions which can be written as $f(x, y, r) = (g(x), h(g(x), r))$, for some functions g and h. In this case, the function's output distribution for different values of x must be either identical (when

$g(x) = g(x')$) or have disjoint supports (when $g(x) \neq g(x')$, which is included in the function's output). Intuitively, the function's output determines the identity of the random distribution $h(g(x), \cdot)$ that was used.

In these two classes of functionalities, only one party can influence the output, so we say they have *uni-directional influence*. If there exists inputs x, x', x'' for Alice and y, y', y'' for Bob so that $f(x, y') \not\equiv f(x, y'')$, and $f(x', y) \not\equiv f(x'', y)$, then both parties can potentially influence the output, and we say that the functionality has *bi-directional influence*.

Isomorphism. $\mathcal{F}$ and $\mathcal{G}$ are isomorphic[6] if either functionality can be UC-securely realized using the other functionality by a protocol that is "local" in the following sense: to realize $\mathcal{F}$ given $\mathcal{G}$ (say), each party maps its input (possibly probabilistically) to inputs for the functionality $\mathcal{G}$, calls $\mathcal{G}$ once with that input and, based on their private input, the output obtained from $\mathcal{G}$, and possibly private random coins, locally computes the final output, without any other communication. It is easy to see that isomorphism is an equivalence relation.

Usefulness of a source. We say that a source of common randomness $\mathcal{G}$ is **useless** in realizing a 2-party functionality $\mathcal{F}$ if either $\mathcal{F}$ could be securely realized in the plain model (i.e., without using $\mathcal{G}$) or $\mathcal{F}$ cannot be securely realized even in the $\mathcal{G}$-hybrid model. Note that we consider only the feasibility question and not any efficiency issues.

2.2 Frontier Analysis

Protocols and transcript trees. We view a 2-party protocol as a weighted tree of possible transcripts. The leaves of the tree correspond to completed transcripts, on which both parties give output. The tree's internal nodes alternate between "Alice" and "Bob" nodes, corresponding to points in the protocol (identified by partial transcripts) at which Alice and Bob send messages, respectively. Given a party's private input and the transcript so far (i.e., a node in the tree), the protocol assigns probabilities to the outgoing edges (i.e., possible next messages). In some settings we also consider nodes corresponding to invocations of ideal functionalities (like $\mathcal{F}_{\mathsf{coin}}$), when appropriate. For these the protocol tree assigns probabilities to the outputs of the functionality (the corresponding "messages" included in the transcripts for these steps) according to the probabilities of parties' inputs and the functionality's internal randomness. An execution of the protocol corresponds to a traversal from root to leaf in the tree.

Probabilities and frontiers. We write $\Pr[v|x, y]$ for the probability that the protocol visits node v (equivalently, generates a transcript with v as a prefix) when executed honestly on inputs x and y. Suppose $\pi_A(x, vb)$ is the probability that when Alice executes the protocol honestly with input x and the transcript so far

[6] The definition given here is a generalization for randomized functionalities of the definition from [15].

is v, her next message is b. Similarly, we define a probability π_B for Bob. Then (assuming Alice speaks first in the protocol):

$$\Pr[v|x,y] = \pi_A(x,v_1)\pi_B(y,v_1v_2)\cdots = \left[\prod_{i \text{ odd}} \pi_A(x,v_1\cdots v_i)\right]\left[\prod_{i \text{ even}} \pi_B(y,v_1\cdots v_i)\right]$$

If we define $\alpha(v,x)$ and $\beta(v,y)$ to be the two parenthesized quantities (equivalently, the product of weights from Alice nodes and Bob nodes in the transcript tree, respectively), then we have $\Pr[v|x,y] = \alpha(v,x)\beta(v,y)$. Thus, in a plain protocol, the two parties make independent contributions to the probability of each transcript. In fact, even if the protocol is allowed to use a selectable source, this property still holds (see Section 4). This property of protocols is crucially used in all frontier analysis in this work.

When S is a set of independent nodes in the transcript tree (prefix-free partial transcripts), we define $\Pr[S|x,y] = \sum_{v\in S}\Pr[v|x,y]$, as all the probabilities in the summation are for mutually exclusive events. If $\Pr[F|x,y] = 1$, then we call F a *frontier.* Equivalently, a frontier is a maximal independent set in the transcript tree. In general, a frontier represents a point in the protocol where a certain event happens, usually defined in terms of the probabilities α and β.

3 Handling General SFE Functionalities

Frontier analysis is most naturally applied to protocols realizing SSFE functionalities — that is, functionalities which give the same output to both parties. So we derive our results for such functionalities. However, we can then extend our characterizations to apply to SFE functionalities with unrestricted outputs using the following lemma (see the full version of this paper [14]):

Lemma 1. *Suppose $\mathcal{H}$ is a functionality that has a passive-secure protocol in the plain model. If $\mathcal{H}$ is useful in UC- or standalone-securely realizing a (possibly randomized) SFE functionality $\mathcal{F}$, then there exists a* **symmetric** *SFE functionality $\mathcal{F}^*$ such that $\mathcal{F}^*$ is isomorphic to $\mathcal{F}$, and $\mathcal{H}$ is useful in (respectively, UC- or standalone-) securely realizing $\mathcal{F}^*$.*

Here, being useful or not is in the sense of the definition given in Section 2.1.

Proving Lemma 1 essentially involves relating SSFE and SFE functionalities. As it turns out, relating symmetric and unrestricted functionalities is most convenient in the setting of passive security. In that setting, we associate with every SFE functionality $\mathcal{F}$ a symmetric functionality which is simply the maximal "common information" provided to the two parties by $\mathcal{F}$. (See proof in the full version [14] for a combinatorial description of this function) Following [11] it is not hard to show that if an SFE functionality $\mathcal{G}$ is not isomorphic to its (symmetric-output) common information functionality then $\mathcal{G}$ must be complete in the passive security setting.

To apply this result, however, we must be careful in relating passive security and active security. It is not necessarily the case that an actively secure protocol

implies a passively secure protocol (since in the passive security setting, the security reduction must map passively corrupt adversaries to passively corrupt simulators). In [14] we show that every SFE functionality is isomorphic to a functionality that is "deviation-revealing" [19]. Such functionalities have the property that active-secure protocols imply passive-secure protocols. Using these two results, we are able to transition from active to passive security, and then argue about generalized vs. symmetric output.

4 Selectable Sources are Useless for Deterministic SFE

In this section we will show that any selectable source is useless for securely realizing any deterministic SFE functionality against computationally unbounded adversaries. In particular this shows that $\mathcal{F}_{\mathsf{coin}}$ is useless for realizing any deterministic SFE functionality.

Theorem 1. *Suppose $\mathcal{F}$ is a 2-party deterministic SFE and $\mathcal{G}$ is a selectable source. Then $\mathcal{F}$ has a standalone-secure (resp. UC-secure) protocol in the $\mathcal{G}$-hybrid model against computationally unbounded adversaries if and only if $\mathcal{F}$ has a standalone-secure (resp. UC-secure) protocol in the plain model.*

To give an overview of our techniques, we present the result for the special case of $\mathcal{F} = \mathcal{F}_{\mathsf{xor}}$ and $\mathcal{G} = \mathcal{F}_{\mathsf{coin}}$. Then we describe the modifications necessary to consider arbitrary $\mathcal{F}$ and arbitrary selectable source $\mathcal{G}$, respectively.

The case of $\mathcal{F}_{\mathsf{xor}}$ and $\mathcal{F}_{\mathsf{coin}}$. This special case illustrates our new frontier-based attack. It is well-known that there is no standalone-secure (or UC-secure) protocol for $\mathcal{F}_{\mathsf{xor}}$ in the plain model (cf. the complete characterization of [12,15]). Also note that standalone security is a special case of UC security. Thus it suffices to show the following:

Lemma 2. *There is no standalone-secure protocol for $\mathcal{F}_{\mathsf{xor}}$ using $\mathcal{F}_{\mathsf{coin}}$, against computationally unbounded adversaries.*

Proof (Sketch). The main novelty in this proof (compared to the techniques in [15]) is the nature of the frontier we consider, in a semi-honest protocol. For semi-honest security, $\mathcal{F}_{\mathsf{xor}}$ does not have a canonical order for *what information* must be revealed by the two parties. This thwarts the analysis in [15], which depends on defining frontiers corresponding to what information is revealed in what order. Nevertheless we show that using a frontier parameterized by a threshold μ on (an appropriately defined notion of) *how much information* is revealed about a party's input, one can devise an attack on purported protocol for $\mathcal{F}_{\mathsf{xor}}$ in the $\mathcal{F}_{\mathsf{coin}}$-hybrid model.

For simplicity, first assume that we are given a protocol π for $\mathcal{F}_{\mathsf{xor}}$ in the *plain* model (i.e., let us ignore $\mathcal{F}_{\mathsf{coin}}$ for the moment). Let α and β be defined as in Section 2. Then for every node v in the transcript tree of π, define

$$\delta_A(v, x, x') = \frac{|\alpha(v, x) - \alpha(v, x')|}{\alpha(v, x) + \alpha(v, x')} \qquad \text{and} \qquad \delta_B(v, y, y') = \frac{|\beta(v, y) - \beta(v, y')|}{\beta(v, y) + \beta(v, y')}.$$

δ_A and δ_B are well-defined after we exclude any nodes that have $\alpha(v,x) = \alpha(v,x') = 0$ or $\beta(v,y) = \beta(v,y') = 0$. Intuitively, $\delta_A(v,x,x')$ and $\delta_B(v,y,y')$ measure how much the transcript reveals about the distinction between x and x', or y and y', respectively. A δ value of 0 means that the partial transcript v is independent of the choice between the two inputs; a value of 1 means that the transcript v is only consistent with one of the two inputs.

Then given a parameter μ, we define a frontier F as follows:

$$F = \left\{ v \;\middle|\; \begin{array}{c} \max\{\delta_A(v,0,1), \delta_B(v,0,1)\} \geq \mu \\ \text{and no proper prefix of } v \text{ also satisfies this condition} \end{array} \right\}$$

Intuitively, F is the first place at which one of the parties has revealed "significant" information about its input, where significance is measured by μ.

Now we sketch an attack based on this frontier. (The actual proof and calculations in [14] follow a slightly different argument, but using the same frontier). Suppose by symmetry that on an honest execution, the protocol assigns the majority of the weight on F to transcripts v satisfying $\delta_B(v,0,1) \geq \mu$. Then, intuitively, Alice can launch an attack as follows. She runs the protocol honestly (say, with input 0) until reaching F. Then at F, the transcript is correlated with Bob's input enough so that Alice can guess Bob's input with bias roughly $\mu/2$. On the other hand, since $\delta_A(v,0,1) < \mu$ with good probability at this point of the protocol, both values for Alice's input are somewhat likely explanations for the transcript seen so far. Therefore if Alice changes her input at this point (by sampling a state consistent with the current transcript and the new input), the outcome of the protocol will change with all but negligible probability, thanks to the correctness guarantee of the protocol. Thus, Alice can significantly correlate her *effective* input with Bob's, so that Bob's output is biased significantly towards 1 (when Bob picks his input at random). But this is a behavior that is not possible in an ideal-world interaction with $\mathcal{F}_{\mathsf{xor}}$, so it constitutes a violation of the security of π.

The only difference when attacking a protocol in the $\mathcal{F}_{\mathsf{coin}}$-hybrid model is that the common coins also influence the probabilities of partial transcripts. One may consider the probability of a partial transcript v (which includes outputs of $\mathcal{F}_{\mathsf{coin}}$) as a product of $\alpha(v,x)$, $\beta(v,y)$, and a contribution $\gamma(v)$ from the combined calls to $\mathcal{F}_{\mathsf{coin}}$. However, $\gamma(v)$ does not depend on x or y, so we can absorb its contribution into (arbitrarily) $\alpha(v,x)$ and the analysis remains valid.[7]

Uselessness of $\mathcal{F}_{\mathsf{coin}}$ for any SFE $\mathcal{F}$. First we consider the case when $\mathcal{F}$ is a *symmetric* SFE functionality. We use the characterization of SSFE functionalities with standalone-secure protocols from [15] to show that if an SSFE functionality $\mathcal{F}$ has no standalone-secure protocol in the plain model, then either there

[7] Note that α and β are defined only in terms of honest behavior by the parties, so that every call to $\mathcal{F}_{\mathsf{coin}}$ delivers its output to both parties in our analysis and associated attack. (Only corrupt parties can prevent output delivery in a functionality with no output fairness guarantee.) Thus our attacks neither rely on fairness nor crucially exploit unfairness in the source of common coins; the adversaries we construct will always choose to deliver the outputs of the setup functionality.

is a standalone-secure protocol for $\mathcal{F}_{\mathsf{xor}}$ in the $\mathcal{F}$-hybrid model, or else there is a frontier-based attack that violates standalone security of every purported protocol for $\mathcal{F}$ in the plain model.

In the first case, Lemma 2 demonstrates that $\mathcal{F}$ can have no standalone-secure protocol in the $\mathcal{F}_{\mathsf{coin}}$-hybrid world. In the second case, we observe that the frontier-based attacks go through unaltered even if the protocols are allowed access to $\mathcal{F}_{\mathsf{coin}}$. This is because the frontier attack merely relies on the fact that in a protocol, given a transcript prefix v, the next message depends only on one of Alice and Bob's inputs. However, this is true even if the protocol has access to $\mathcal{F}_{\mathsf{coin}}$— the bits from $\mathcal{F}_{\mathsf{coin}}$ being independent of both parties' inputs.

This allows us to conclude that in either case, there can be no protocol for $\mathcal{F}$ in the $\mathcal{F}_{\mathsf{coin}}$-hybrid model, giving us the following lemma (see the full version [14] for more details).

Lemma 3. *If $\mathcal{F}$ is a 2-party deterministic SSFE that has no standalone-secure (resp. UC-secure) protocol against unbounded adversaries in the plain model, then $\mathcal{F}$ has no standalone-secure (resp. UC-secure) protocol in the $\mathcal{F}_{\mathsf{coin}}$-hybrid model.*

Replacing $\mathcal{G}$ with an arbitrary selectable source. Our analysis goes through with minimal modification when $\mathcal{F}_{\mathsf{coin}}$ is replaced by an arbitrary selectable source. Recall that in a selectable source functionality $\mathcal{G}$, only one party can influence the output at a time (depending on which "direction" $\mathcal{G}$ is used in). When $\mathcal{G}$ is used such that only Alice influences the output, the influence on the transcript's probability can be collected into the term $\alpha(v, x)$. Similarly, when only Bob can influence the output of $\mathcal{G}$, the influence can be collected into the term $\beta(v, y)$. Therefore, we can still write $\Pr[v|x, y] = \alpha(v, x)\beta(v, y)$ for appropriate α and β. Each invocation of $\mathcal{G}$ is an atomic event with respect to the frontiers and to the adversary's changes in behavior in our our attacks.

Extending to general SFE functionalities. Finally, we prove Theorem 1, using Lemma 1. Note that a selectable source has a passive secure protocol (Alice samples an output and gives it to Bob). Thus if there exists any SFE functionality $\mathcal{F}$ for which some selectable source is useful in (UC- or standalone-) securely realizing, then by Lemma 1 selectable source is useful in (UC- or standalone-) securely realizing some SSFE functionality as well, contradicting Lemma 3.

5 Coins are useless for Randomized SFE

In this section, we characterize the set of randomized SFE functionalities that can be reduced to $\mathcal{F}_{\mathsf{coin}}$.

Since $\mathcal{F}_{\mathsf{coin}}$ itself is not securely realizable (in the UC or standalone model) against computationally unbounded adversaries, common randomness clearly allow more functionalities to be securely realized. In particular common randomness can be used to generate a shared sample from a publicly agreed-upon

distribution. However, we show that this is essentially the only use of common randomness, when UC security is required [8]. More precisely,

Theorem 2. *A randomized SFE functionality $\mathcal{F}$ has a UC-secure protocol in the $\mathcal{F}_{\mathsf{coin}}$-hybrid model if and only if $\mathcal{F}$ is isomorphic to the SSFE functionality $\mathcal{F}^*$ with output function $\mathcal{F}^*$ such that $\mathcal{F}^*(x, y, r) = (h(x), r)$, where h is a deterministic function.*

Note that a secure protocol for $\mathcal{F}^*(x, y, r)$ above is simple: Alice sends $h(x)$ to Bob, and then they obtain uniformly random coins r from $\mathcal{F}_{\mathsf{coin}}$. Thus, any UC secure protocol for f which uses $\mathcal{F}_{\mathsf{coin}}$ can be replaced by one of the following form: (1) one party sends a function of its input to the other party; (2) both parties access $\mathcal{F}_{\mathsf{coin}}$ to obtain coins r; (3) both parties carry out local computation to produce their outputs.

Given Lemma 1, it is enough to establish our characterization for the special case of *symmetric* SFE functionalities (for which we shall denote the common output by $f(x, y, r)$).

The first step in proving Theorem 2 for SSFE is to show that only one party's input can have influence on the outcome of the other party.

Lemma 4. *If $\mathcal{F}$ is a 2-party randomized SSFE functionality with a UC-secure protocol in the $\mathcal{F}_{\mathsf{coin}}$-hybrid model, then $\mathcal{F}(x, y)$ is distributed as $\mathcal{F}'(x)$ (or $\mathcal{F}'(y)$), where $\mathcal{F}'$ is some randomized function of one input.*

If $\mathcal{F}$ does not have the form $\mathcal{F}'(x)$ or $\mathcal{F}'(y)$, we call it an SSFE functionality with *bidirectional influence*. Using a lemma proven in the full version of this paper [14], we know that if a SSFE $\mathcal{F}$ with bidirectional influence has a UC-secure protocol in the $\mathcal{F}_{\mathsf{coin}}$-hybrid then there exists a semi-honest protocol for OT. However, this is not possible against computationally unbounded adversaries and hence, $\mathcal{F}$ can not have bidirectional influence.

Frontiers of influence. Suppose we are given a protocol π for f in the $\mathcal{F}_{\mathsf{coin}}$-hybrid model, with simulation error ϵ. Without loss of generality, we assume that the last step of π is to toss a random coin which is included in the output.[9] First, define $\mathcal{O}_v^x$ to be the output distribution of the protocol when executed

[8] The full version of this paper [14] contains examples of randomized SSFE for which $\mathcal{F}_{\mathsf{coin}}$ is useful in a more non-trivial way, but for *standalone* security. For one such example, consider the protocol to compute the minimum of the private inputs of the two parties [15,12]: Alice declares whether her input is 0 or 2; next, conditioned on Alice input being 2, Bob declares whether his input is 1 or 3. This protocol is a standalone secure protocol for the deterministic SSFE: $\left[\begin{smallmatrix} 0 & 0 \\ 1 & 3 \end{smallmatrix}\right]$. Now, we randomize the output when Alice input is 1: Based on whether Bob's input is 1 or 3 we use $\mathcal{F}_{\mathsf{coin}}$ to uniformly sample from the set $\{1, 2\}$ or $\{2, 3\}$, respectively. This is a standalone-secure protocol for the randomized SSFE: $\left[\begin{smallmatrix} (1,0,0,0) & (1,0,0,0) \\ (0,\frac{1}{2},\frac{1}{2},0) & (0,0,\frac{1}{2},\frac{1}{2}) \end{smallmatrix}\right]$, where the vectors indicate probability distribution over an output alphabet of size 4.

[9] To see that this is without loss of generality, define a randomized SSFE f' which on input x, outputs $f(x)$ as well as a random bit. Then define π' to be the protocol

honestly on (Alice) input x, *starting from partial transcript* v. We use this to define our first frontier:

$$G = \left\{ v \;\middle|\; \begin{array}{c} \forall x', x'' : \mathrm{SD}\left(\mathcal{O}_v^{x'}, \mathcal{O}_v^{x''}\right) < \sqrt{\epsilon} \\ \text{and no ancestor of } v \text{ satisfies the same condition} \end{array} \right\}$$

Intuitively, G represents the point at which Alice's input has first exhausted any "significant" influence on the final output distribution — her input can no longer change the output distribution by more than $\sqrt{\epsilon}$. Next, note that the only way to induce an output distribution in the ideal world is to choose an input x according to some distribution $\mathcal{D}$ and then send x to f, yielding the output distribution $\{f(x)\}_{x \leftarrow \mathcal{D}}$. Let $\mathcal{S}$ be the space of all possible output distributions that can be induced in this way.[10] We use this to define a collection of frontiers, one for each value of x.

$$F_x = \{v \mid \mathrm{SD}(\mathcal{O}_v^x, \mathcal{S}) > \sqrt{\epsilon} \text{ and no ancestor of } v \text{ satisfies the same condition}\}$$

Intuitively F_x represents the first time that randomness has had a "significantly" non-trivial influence on the output when Alice's input is x. Here, the influence of randomness in the protocol is considered non-trivial if the protocol has reached a point such that the conditional output distribution induced by the protocol starting from that point cannot be achieved by Alice in the ideal world.

We now show that in a secure protocol, Alice's input must completely exhaust its influence before the randomness from $\mathcal{F}_{\mathsf{coin}}$ can begin to influence the output distribution.

Lemma 5. *In the above setting, let $F_x < G$ denote the event that the protocol generates a transcript that encounters frontier F_x strictly before encountering frontier G. Then $\Pr[F_x < G|x]$ is negligible for all x.*

Proof (Sketch). Consider a malicious Alice that runs π honestly on input x. Whenever this adversary encounters F_x strictly before G, it reports the resulting partial transcript to the environment. Being in the frontier F_x, this transcript intuitively represents an assertion by the adversary that it can realize an output distribution $\mathcal{O}_v^x$ that is impossible to effect in the ideal world (by continuing hereafter with input x). Being before G, the transcript also indicates an assertion by the adversary that it can still induce two "significantly" different output distributions (by continuing hereafter with one of the inputs from the condition in the definition of G). The environment can choose to challenge the adversary on any of these choices, and in the real world the adversary can always succeed. However, for any simulator in the ideal world, there must be some challenge

which runs π and in the last step uses $\mathcal{F}_{\mathsf{coin}}$ to toss a coin which is included in the output. It is easy to see that if π is a secure protocol for f, then π' is a secure protocol for f', so proving the insecurity of π' establishes the insecurity of π.

[10] Note that $\mathcal{S}$ is the space of convex combinations of $\{f(x) \mid x\}$, where here $f(x)$ denotes the discrete probability distribution itself, represented by a stochastic vector.

for which the simulator must fail. Namely, if the simulator has already sent an input to ideal f at the time it makes its "assertion", then it cannot proceed to induce two significantly different output distributions on command — the output is already fixed. On the other hand, if the simulator has not sent an input to the ideal f, then it cannot proceed to realize an output distribution that is impossible in the ideal world.

Thus this adversary violates the security of f with success proportional to $\Pr[F_x < G|x]$, so we conclude that this probability must be negligible.

Using the previous two lemmas, we can now prove the special case of Theorem 2, restricted to SSFE functionalities:

Lemma 6. *A 2-party randomized SSFE functionality $\mathcal{F}$ has a UC-secure protocol in the $\mathcal{F}_{\mathsf{coin}}$-hybrid model against computationally unbounded adversaries if and only if $\mathcal{F}$ is isomorphic to the SSFE functionality $\mathcal{F}^*$ with output function f^* such that $f^*(x, y, r) = (h(x), r)$, where h is a deterministic function.*

Proof. The complete proof of the lemma is provided in the full version of this paper [14]. Lemma 5 shows that there is a frontier G that separates all of the influence of Alice's input (before G) from all of the influence of $\mathcal{F}_{\mathsf{coin}}$ (after G).

Our analysis relies on the *geometric interpretation of possible output distributions*. As before, let $\mathcal{S}$ denote the space of output distributions that can be realized in the ideal world by randomly choosing an input and sending it to f to obtain a sample of $f(x)$. $\mathcal{S}$ is the convex closure of a finite set of points $f(x)$. Call an input x *fundamental* if $f(x)$ is a corner on the convex hull of $\mathcal{S}$. Without loss of generality, we restrict our attention exclusively to fundamental inputs.[11]

Let x be a fundamental input. Since x affects the output of the protocol only negligibly after the transcript reaches G, we have that $f(x)$ is statistically close to a convex combination of $\{\mathcal{O}_v^x \mid v \in G\}$. An overwhelming weight of these distributions $\mathcal{O}_v^x$ are negligibly close (in statistical distance) to the space $\mathcal{S}$. By a geometric argument, since $f(x)$ is a *corner* in the space $\mathcal{S}$, we have that an overwhelming weight of $\mathcal{O}_v^x$ distributions are negligibly close to $f(x)$.

Thus, consider any two inputs x, x' such that $f(x) \not\equiv f(x')$. The statistical distance between these two distributions is a constant Δ. The above argument implies that x and x' must induce distributions over G that have statistical distance negligibly close to 1. In other words, executing the protocol until G unambiguously determines the distribution $f(x)$; after G, x has no more influence on the output. Then it is straight-forward to show that the following simple protocol is also secure for f: Alice sends a description of the distribution $f(x)$ to Bob (say, the lexicographically smallest x^* s.t. the distributions of $f(x)$ and $f(x^*)$ are identical). Both parties use $\mathcal{F}_{\mathsf{coin}}$ to generate random coins r and use them to compute a sample from the distribution $f(x)$. Then it is clear that f has the desired form — the output of this protocol is computed from a deterministic function of x along with independent coins.

[11] Any non-fundamental input x is redundant in f and we can remove it to obtain an isomorphic functionality (for details refer to the full version of this paper [14]).

On extending to selectable sources. Unlike our results in Section 4, Theorem 2 does **not** generalize to arbitrary selectable sources (instead of just $\mathcal{F}_{\mathsf{coin}}$). To see this, one can easily construct a selectable source f which is not of the form $f(x, y, r) = (h(x), r)$. Then trivially f has a UC-secure protocol using some selectable source (namely, itself), but f is not of the form required by Theorem 2.

Indeed, to prove Theorem 2, we made a crucial distinction between Alice's choice of input influencing the output distribution and $\mathcal{F}_{\mathsf{coin}}$ influencing the output distribution. This distinction is lost if $\mathcal{F}_{\mathsf{coin}}$ is replaced by a functionality in which Alice is allowed to influence the output.

On a common random string (CRS) vs. $\mathcal{F}_{\mathsf{coin}}$. A common random string (CRS) is a source of shared randomness in which all random bits are generated once and for all at the beginning of a protocol interaction, rather than as-needed, as with $\mathcal{F}_{\mathsf{coin}}$. Our proof of Theorem 2 states that the influence of the parties' inputs ends before the influence of the shared randomness begins. Since the influence of a CRS must happen at the start of a protocol, a CRS is useless for SSFEs except those of the form $f(x, y, r) = h(x)$ (no influence from shared randomness) or $f(x, y, r) = h(r)$ (no influence from parties' inputs), for a deterministic h.

6 Randomized Functionalities and Computational Intractability

Our results so far have been presented in the computationally unbounded setting. However, they do extend somewhat to the probabilistic polynomial time (PPT) setting (where all entities, including the adversary and the environment are PPT), and yield interesting connections with computational intractability assumptions. These results are similar in spirit to the connections established in [16], but unlike there, are applicable to randomized functionalities.

Firstly, in the case of deterministic SFE functionalities, we obtain the following unconditional result in the PPT setting

Theorem 3. *For every 2-party deterministic SFE $\mathcal{F}$ and selectable source $\mathcal{G}$, $\mathcal{F}$ has a standalone secure protocol in the $\mathcal{G}$-hybrid model in the PPT setting, if and only if $\mathcal{F}$ has a standalone secure protocol in the plain model in the PPT setting. (for the proof, consult the full version [14]).*

Our other results for the PPT setting are conditional. An important observation in [16] was that, statements of the form "2-party functionality $\mathcal{F}$ has a UC-secure protocol in the $\mathcal{G}$-hybrid world (in the PPT setting)" are either known to be unconditionally true or false, or tend to be *equivalent* to the assumption that one-way functions exist, or the assumption that there is an oblivious transfer (OT) protocol secure against semi-honest adversaries. [16] study a large class of such statements for deterministic $\mathcal{F}$ and $\mathcal{G}$, and show that for every one of them the corresponding statement falls into one of the four classes listed above. An important problem left open is to understand whether the same pattern holds when considering *randomized* functionalities.

Our results suggest that this may be the case: the only intractability assumptions (other than being known to be unconditionally true or false) that arise among randomized functionalities still seem to be the existence of OWF and the existence of a semi-honest OT protocol. In particular we have the following two results (refer to the full version of this paper [14]):

Theorem 4. *Let $\mathcal{F}$ be any 2-party SFE functionality, possibly randomized. If one-way functions do not exist then $\mathcal{F}$ has a UC-secure protocol in the $\mathcal{F}_{\mathsf{coin}}$-hybrid model in the PPT setting, if and only if $\mathcal{F}$ is a publicly-selectable source.*

Theorem 5. *The following three statements are equivalent:*

1. *There exists a semi-honest OT protocol.*
2. *$\exists$ (possibly randomized) 2-party SSFE $\mathcal{F}$ with bidirectional influence : $\mathcal{F}$ is UC securely-realizable in $\mathcal{F}_{\mathsf{coin}}$-hybrid world.*
3. *$\forall$ (possibly randomized) 2-party SSFE $\mathcal{F}$ with bidirectional influence : $\mathcal{F}$ is UC securely-realizable in $\mathcal{F}_{\mathsf{coin}}$-hybrid world.*

The main task in proving the above is to show that (2) $\Rightarrow$ (1), which shown in the full version. (1) $\Rightarrow$ (3) follows from a result proven in [9,17] on the completeness of $\mathcal{F}_{\mathsf{coin}}$. (3) $\Rightarrow$ (2) is trivial.

7 Conclusion and Future Work

Recently, [16] made a case for "cryptographic complexity theory," trying to understand the qualitative difference between different multiparty functionalities. However, the results there were confined to deterministic functionalities; the universe of randomized functionalities is vastly more complex, and is little understood. Among other things, this work initiates a systematic study of randomized functionalities, by proving the low-complexity nature of certain classes of randomized functionalities. In this work we do not consider randomized functionalities of higher levels of complexity, nor do we seek to classify all kinds of randomized functionalities. Nevertheless, we believe that our proof techniques — both for the computationally unbounded setting and for the PPT setting — will be useful in such a study. We leave this for future work.

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. D. Beaver. Precomputing oblivious transfer. In D. Coppersmith, editor, *CRYPTO*, volume 963 of *Lecture Notes in Computer Science*, pages 97–109. Springer, 1995.
3. M. Blum, P. Feldman, and S. Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *STOC*, pages 103–112. ACM, 1988.
4. 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.

5. R. Canetti, Y. Lindell, R. Ostrovsky, and A. Sahai. Universally composable two-party computation. In *Proc. 34th STOC*, pages 494–503. ACM, 2002.
6. B. Chor and E. Kushilevitz. A zero-one law for boolean privacy (extended abstract). In *STOC*, pages 62–72. ACM, 1989.
7. R. Cleve. Limits on the security of coin flips when half the processors are faulty (extended abstract). In *STOC*, pages 364–369. ACM, 1986.
8. R. Cleve and R. Impagliazzo. Martingales, collective coin flipping and discrete control processes. Manuscript, 1993. http://www.cpsc.ucalgary.ca/~cleve/pubs/martingales.ps.
9. I. Damgård, J. B. Nielsen, and C. Orlandi. On the necessary and sufficient assumptions for UC computation. Cryptology ePrint Archive, Report 2009/247, 2009. http://eprint.iacr.org/.
10. R. Impagliazzo and M. Luby. One-way functions are essential for complexity based cryptography. In *Proc. 30th FOCS*, pages 230–235. IEEE, 1989.
11. J. Kilian. More general completeness theorems for secure two-party computation. In *Proc. 32th STOC*, pages 316–324. ACM, 2000.
12. R. Künzler, J. Müller-Quade, and D. Raub. Secure computability of functions in the it setting with dishonest majority and applications to long-term security. In O. Reingold, editor, *TCC*, volume 5444 of *Lecture Notes in Computer Science*, pages 238–255. Springer, 2009.
13. E. Kushilevitz. Privacy and communication complexity. In *FOCS*, pages 416–421. IEEE, 1989.
14. H. K. Maji, P. Ouppaphan, M. Prabhakaran, and M. Rosulek. Exploring the limits of common coins using frontier analysis of protocols. In *TCC*, 2011. Full version at http://eprint.iacr.org/.
15. 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.
16. H. K. Maji, M. Prabhakaran, and M. Rosulek. Cryptographic complexity classes and computational intractability assumptions. In A. C.-C. Yao, editor, *ICS*, pages 266–289. Tsinghua University Press, 2010.
17. H. K. Maji, M. Prabhakaran, and M. Rosulek. A zero-one law for cryptographic complexity with respect to computational UC security. In T. Rabin, editor, *CRYPTO*, volume 6223 of *Lecture Notes in Computer Science*, pages 595–612. Springer, 2010.
18. T. Moran, M. Naor, and G. Segev. An optimally fair coin toss. In *TCC 2009*, volume 5444 of *Lecture Notes in Computer Science*, pages 1–18. Springer, March 2009.
19. 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.
20. A. C. Yao. Protocols for secure computation. In *Proc. 23rd FOCS*, pages 160–164. IEEE, 1982.