The Limits of Common Coins: Further Results

Hemanta K. Maji[⋆1] and Manoj Prabhakaran[⋆⋆2]

[1] University of California, Los Angeles
[2] University of Illinois, Urbana-Champaign

Abstract. In [8] it was shown that the coin-tossing functionality $\mathcal{F}_{\mathsf{coin}}$ has limited use in 2-party secure function evaluation (SFE) in the computationally unbounded (a.k.a information-theoretic) setting. Further it was shown that for $\mathcal{F}_{\mathsf{coin}}$ to be useful in securely realizing any one in a *a large class* of symmetric SFE (SSFE) functionalities, a certain computational assumption (namely the existence of a semi-honest secure OT protocol) is necessary and sufficient. In this work, we close a gap in the class of SSFE functionalities for which this result was proven in [8]: we show that $\mathcal{F}_{\mathsf{coin}}$ can be used to securely realize *any* SSFE functionality that cannot be realized in the computationally unbounded setting, if and only if there exists a semi-honest secure OT protocol.

1 Introduction

Multi-party computation is a central problem in modern cryptography. An important question regarding secure function evaluation has been to understand the relative "cryptographic complexity" of the different functions that are evaluated – i.e., secure evaluation of which functions can be reduced to that of which other functions. While several aspects of this problem have been well-studied (e.g. [4,5,9,7,6] to name a few) a large number of problems remain open. In particular, we do not fully understand how, in the probabilistic polynomial time (PPT) setting, the cryptographic complexity of various functionalities relates to computational intractability assumptions, though again several works have resolved this question in various special cases (see for e.g. [1,10]).

In this work, we follow up on [8], which studied the power of the coin-tossing functionality $\mathcal{F}_{\mathsf{coin}}$ in the 2-party setting. Our focus will be on randomized symmetric secure function evaluation (SSFE) functionalities, in which both parties get the same output. In [8], the following was shown (along with other results):

– A 2-party SSFE functionality with "bi-directional influence" reduces to $\mathcal{F}_{\mathsf{coin}}$ in the PPT setting if and only if there exists a semi-honest secure OT protocol.

⋆ This work was carried out when the author was at University of Illinois, Urbana-Champaign. e-mail: hmaji2@illinois.edu
⋆⋆ e-mail: mmp@illinois.edu

The assumption that there exists a semi-honest secure OT protocol (in the PPT setting) is a classic assumption in the context of secure function evaluation, comparable in generality to the existence of a one-way function, but stronger. The notion of reduction here is that of secure reduction in the Universal Composition framework.

Here, "bi-directional influence" refers to the case that the output distribution depends on both parties' inputs. The class of SSFE functionalities without bi-directional influence can be "less complex," and may indeed reduce to $\mathcal{F}_{\mathsf{coin}}$ unconditionally. On the other hand, some other SSFE functionalities without bi-directional influence may be complex enough that they require some computational complexity assumption to reduce to $\mathcal{F}_{\mathsf{coin}}$. This raises the question if such reductions are equivalent to *other* computational intractability assumptions. In this paper we answer this question in the negative:

- A 2-party SSFE functionality $\mathcal{F}$ reduces to $\mathcal{F}_{\mathsf{coin}}$ in the PPT setting, if and only if,
  - either $\mathcal{F}$ reduces to $\mathcal{F}_{\mathsf{coin}}$ in the computationally unbounded setting, or
  - there exists a semi-honest secure OT protocol.

Since [8] explicitly characterized the class of SSFE functionalities that reduce to $\mathcal{F}_{\mathsf{coin}}$ in the computationally unbounded setting, our result completely resolves the question of which SSFE functionalities reduce to $\mathcal{F}_{\mathsf{coin}}$ under what computational assumptions. Indeed, *our result shows that there is only one relevant complexity assumption in this case, namely, the existence of a semi-honest secure OT protocol.*

1.1 Outline

The main components in proving our main result (Theorem 1) are the following:

- In Section 2 we define a new class of SSFE functionalities with "uni-directional influence" called *oblivious sampling functionalities* (Definition 1) that is crucial to our new result.
- Oblivious Sampling functionalities are essentially the ones that the result in [8] mentioned above, did not cover. This is formalized as Lemma 2.
- Then we go on to show that if an oblivious sampling functionality reduces to $\mathcal{F}_{\mathsf{coin}}$, then there must exist a semi-honest secure OT protocol (Lemma 3).

In Section 3 we provide an immediate corollary of our main result (Corollary 2) as well as a generalization (Theorem 3) which replaces $\mathcal{F}_{\mathsf{coin}}$ with a larger class of SSFE functionalities (called publicly-selectable source, originally defined in [8] and extended in Section 2).

2 Preliminaries

In this section we introduce important definitions (including some from [8]) that are required to state and prove our results.

Secure function evaluation functionalities. A (randomized) 2-party symmetric secure function evaluation (SSFE) functionality $\mathcal{F}_f$ is specified by a function $f : X \times Y \times R \to Z$.[3] The functionality takes inputs $x \in X$ from Alice, $y \in Y$ from Bob, uniformly samples $r \in R$ and outputs $f(x, y, r)$ to both Alice and Bob. (If a party is actively corrupt, it can obtain its own output first and decide whether the output should be delivered to the other party.) We shall write $f(x, y)$ to denote the *distribution* of $f(x, y, r)$ when r is sampled uniformly from R. An example is the common randomness functionality, denoted by $\mathcal{F}_{\mathsf{coin}}$, with $X = Y = \{0\}$, $R = \{0, 1\})$, $f(x, y, r) = r$.

We shall partition SSFE functionalities into three classes. We say that in $\mathcal{F}_f$ Alice has influence on the output if there exist $x', x'' \in X$, and $y \in Y$ such that the distributions $f(x', y) \neq f(x'', y)$ (and similarly we define Bob having influence on the output).

- *Uninfluenced functionalities.* In this case neither Alice nor Bob has influence on the output; i.e., the output is from a constant distribution and hence we can set $f(x, y, r)$ to be $f(r)$.
- *Functionalities with unidirectional influence.* In this case exactly one party, say Alice (or Bob), has influence on the output. We shall denote the output distribution as $f(x)$ (or $f(y)$) in this case. By abuse of notation, we will use the same notation to denote a random variable drawn from this distribution.
- *Functionalities with bidirectional influence.* In this case both parties have influence on the output.

Oblivious Sampling. In order to prove our results, we define a new class of SSFE functionalities with unidirectional influence called oblivious sampling functionalities. Intuitively, in these functionalities, Alice decides the distribution from which the common output is sampled, but Bob does not fully learn which distribution the output comes from.

Towards formally defining this class, we require the notion of a non-redundant input. Let $\mathcal{F}_f$ be a 2-party SSFE functionality with unidirectional influence. Then x is a *redundant input* if there exists a set $X' \subseteq X$ such that for each $x' \in X'$, $f(x') \neq f(x)$, but $f(x)$ is a convex combination of $\{f(x') | x' \in X'\}$ (i.e., $f(x) = \sum_{x' \in X'} \alpha_{x'} f(x')$ for $\alpha_{x'} > 0$ such that $\sum_{x' \in X'} \alpha_{x'} = 1$).

Definition 1. *A 2-party SSFE functionality $\mathcal{F}_f$ with unidirectional influence (say Alice has influence) is called an* Oblivious Sampling functionality *if there exist two non-redundant inputs x_0, x_1 such that*

[3] In this work, unless otherwise specified, we allow functionalities to be randomized by default. A "symmetric" functionality is one which gives the same output to both parties, and we restrict our general definition to such functionalities. Finally, as in [8] and related works, we shall always consider X, Y, R, Z to be finite and of constant size independent of the security parameter that appears in the definition of security; also the probabilities involved are constant. However, the results do extend to the case when these sets are polynomially large in the security parameter, and the probabilities are such that the outcome for a "non-redundant" input (x, y) cannot be approximated as a convex combination of other inputs within an inverse polynomial statistical distance.

– $\exists z' \Pr[f(x_0) = z'] \neq \Pr[f(x_1) = z']$ *(i.e., $f(x_0) \neq f(x_1)$), and*
– $\exists z \Pr[f(x_0) = z], \Pr[f(x_1) = z] > 0$ *(i.e., supports of $f(x_0)$ and $f(x_1)$ intersect).*

From a characterization in [8], it follows that Oblivious Sampling functionalities are not UC-securely reducible to $\mathcal{F}_{\mathsf{coin}}$ in the computationally unbounded setting (publicly-selectable sources, as defined below, are the only SSFE functionalities which are). A modification of the proof there could be used to show that one-way functions must exist for such a reduction to be possible in the PPT setting. What we shall show is that, in fact, a semi-honest OT protocol must exist for such a reduction.

We also note that Oblivious Sampling functionalities are not complex enough to be complete in the computationally unbounded setting. Since the characterization in [5] of complete SSFE functions in the *semi-honest security setting* requires both parties to have more than one input value,[4] there is no semi-honest secure reduction of say, OT to an Oblivious Sampling functionality; and therefore there is no UC-secure of reduction of OT to an Oblivious Sampling functionality (as OT is deviation-revealing [12]). This makes it non-trivial to prove that such a reduction in the PPT setting implies the existence of a semi-honest OT protocol.

Publicly-selectable sources. [8] defined an SSFE functionality to be a publicly-selectable source if it is of the form $\mathcal{F}_f$ where $f(x, y, r) = (g(x), h(g(x), r))$ (possibly relabeled using an output alphabet), for some functions g and h (or with Alice's and Bob's roles interchanged). That is, the function's output distribution for different values of x must either be identical (when $g(x) = g(x')$) or have disjoint supports (when $g(x) \neq g(x')$). We slightly extend this definition so that $\mathcal{F}_f$ is called a publicly-selectable source if $\mathcal{F}'_f$ obtained by restricting $\mathcal{F}_f$ to non-redundant inputs has the same property.

Definition 2. *A 2-party SSFE functionality $\mathcal{F}_f$ with unidirectional influence (say Alice has influence) is called a* publicly-selectable source *if, for every two non-redundant inputs x_0, x_1 such that $f(x_0) \neq f(x_1)$, supports of $f(x_0)$ and $f(x_1)$ are disjoint (i.e., $\nexists z \Pr[f(x_0) = z], \Pr[f(x_1) = z] > 0$).*

Note that with this modification in the definition, an oblivious sampling functionality could alternately be defined as an SSFE functionality with unidirectional influence that is *not* a publicly-selectable source (as redefined here).

We point out that any publicly-selectable source functionality UC-securely reduces to $\mathcal{F}_{\mathsf{coin}}$ by a protocol in which the party with the influence specifies one of the non-redundant inputs, and then the two parties use $\mathcal{F}_{\mathsf{coin}}$ to sample an outcome from that distribution. Indeed, it follows from the characterization in [8] that these are the only SSFE functionalities which can be UC-securely reduced to $\mathcal{F}_{\mathsf{coin}}$.

[4] [5] showed that an SSFE functionality is semi-honest complete iff there exist x_0, x_1, y_0, y_1, z, such that $\Pr[z|x_0, y_0], \Pr[z|x_0, y_1] > 0$ and $\Pr[z|x_0, y_0] \Pr[z|x_1, y_1] \neq \Pr[z|x_0, y_1] \Pr[z|x_1, y_0]$.

Secure Reductions. We use standard security notions, that are summarized in Appendix A. We say that a functionality $\mathcal{F}$ *UC-securely reduces* (or simply, reduces) to a functionality $\mathcal{G}$ if there exists a universally composable protocol that securely realizes $\mathcal{F}$, in which the parties are allowed access to ideal instances of the functionality $\mathcal{G}$. We distinguish between security in the probabilistic polynomial time (PPT) setting — in which the adversaries and the environment — are restricted to be PPT, and the computationally unbounded setting.

A functionality is called *trivial* if it can be UC-securely realized by a protocol in which the parties use only a plain communication functionality to interact with each other. For finite functionalities as we consider, the class of trivial functionalities remains the same in the PPT and computationally unbounded settings. In particular, based on the characterization in [12] it is easy to see that the 2-party SSFE functionalities that are trivial are unidirectional functionalities in which one party, say Alice, can determine the outcome as a deterministic function of its input (and may in addition have redundant inputs).

Oblivious Transfer. We shall refer to the oblivious transfer or OT functionality which takes two bits (x_0, x_1) as input from Alice, a single bit b from Bob as input, and outputs x_b to Bob, but nothing to Alice. (Note that this is not an SSFE functionality, because it is not symmetric.) The only computational intractability assumption that is referred to by our results is that there exists a protocol that is a secure realization of the OT functionality against semi-honest adversaries in the PPT setting. It is known that this is equivalent to the existence of an OT protocol that is secure against active adversaries as well, if restricted to the standalone security (as opposed to UC security) case (and even in a "black-box" sense [3]).

3 Our Results

In this section we describe our main results, which are proven in the subsequent sections.

Theorem 1. *If an SSFE functionality $\mathcal{F}$ reduces to $\mathcal{F}_{\mathsf{coin}}$ in the PPT setting, then either $\mathcal{F}$ is reducible to $\mathcal{F}_{\mathsf{coin}}$ in the computationally unbounded setting or there exists a semi-honest secure OT protocol.*

Our main contribution is to recognize the gap left behind by the results in [8] which only addressed the case of SSFE functionalities $\mathcal{F}$ with bi-directional influence. We fill this gap by identifying *oblivious sampling functionalities* (Definition 1) as an interesting class of SSFE functionalities, and showing that if such a functionality reduces to $\mathcal{F}_{\mathsf{coin}}$, then there exists a semi-honest secure OT protocol (Lemma 3).

We note that if a semi-honest secure OT protocol exists, then it is known that $\mathcal{F}_{\mathsf{coin}}$ is complete in the PPT setting and a hence $\mathcal{F}_{\mathsf{coin}}$ is useful for realizing a functionality like OT [11]. Hence we have the following corollary of the above theorem.

Corollary 2 *The following statements are equivalent:*

1. *Some 2-party SSFE functionality $\mathcal{F}$ that is not UC-reducible to $\mathcal{F}_{\mathsf{coin}}$ in the computationally unbounded setting, reduces to $\mathcal{F}_{\mathsf{coin}}$ in the PPT setting.*
2. *There exists a semi-honest secure OT protocol (in the PPT setting).*
3. *Every 2-party SSFE functionality reduces to $\mathcal{F}_{\mathsf{coin}}$ in the PPT setting.*

Finally, we can extend the above results to a wider class of functionalities than just $\mathcal{F}_{\mathsf{coin}}$. In particular, we show the following.

Theorem 3. *If $\mathcal{G}$ is a publicly-selectable source, and an SSFE functionality $\mathcal{F}$ reduces to $\mathcal{G}$ in the PPT setting, then either $\mathcal{F}$ is reducible to $\mathcal{G}$ in the computationally unbounded setting or there exists a semi-honest secure OT protocol.*

To prove this we show that any non-trivial publicly-selectable source $\mathcal{G}$ is "equivalent" to $\mathcal{F}_{\mathsf{coin}}$ in that either functionality can be reduced to the other (Lemma 1).

4 Proofs

In this section we prove Theorem 1 and Theorem 3. But first we prove the following simple lemma that will be useful in both proofs.

Lemma 1. *For any publicly-selectable source $\mathcal{G}$, in the computationally unbounded (as well as PPT) setting, $\mathcal{G}$ reduces to $\mathcal{F}_{\mathsf{coin}}$; also, $\mathcal{F}_{\mathsf{coin}}$ reduces to $\mathcal{G}$, unless $\mathcal{G}$ is trivial.*

Proof. W.l.o.g, let Alice be the party whose input may have influence on the output in $\mathcal{G}$. Let $\mathcal{D}$ denote the set of output distributions for non-redundant inputs for Alice. Note that since $\mathcal{G}$ is a publicly-selectable source, the distributions in $\mathcal{D}$ have disjoint supports.
$\mathcal{G}$ reduces to $\mathcal{F}_{\mathsf{coin}}$: this follows from a simple protocol for $\mathcal{G}$ as follows (we omit the routine security analysis):

- On input x, Alice determines the unique convex combination of distributions in $\mathcal{D}$ that equals the output distribution for x. The uniqueness is a consequence of those distributions having disjoint supports.
- Alice samples an element from $\mathcal{D}$ according to its weight in the above convex combination, and announces it. (We remark that a cheating Alice could use any strategy to choose an element from $\mathcal{D}$; however it can be mapped to simply choosing an input and then following the protocol honestly.)
- Alice and Bob obtain coins from $\mathcal{F}_{\mathsf{coin}}$, and use them to sample an outcome from the announced distribution.

$\mathcal{F}_{\mathsf{coin}}$ reduces to $\mathcal{G}$, unless $\mathcal{G}$ is trivial: $\mathcal{G}$ is trivial iff every distribution in $\mathcal{D}$ has zero entropy. Otherwise the following is a secure protocol[5] for $\mathcal{F}_{\mathsf{coin}}$ using $\mathcal{G}$

[5] Note that in a secure realization for $\mathcal{F}_{\mathsf{coin}}$ (without guaranteed output delivery) either party is allowed to abort the protocol, possibly after seeing the outcome of the protocol. This is the standard UC security guarantee for 2-party functionalities (and more generally, when there is no honest majority assumption).

(again, we omit the standard security analysis). Briefly, in this protocol the parties apply a von Neumann extractor to the outcome sampled from $\mathcal{G}$, to obtain a fair coin.

- Let x be a fixed non-redundant input for Alice such that the output distribution for x is in $\mathcal{D}$ and has positive entropy. Let $Z_0 \subseteq Z$ be a subset of the outcomes so that for input x, the probability that the outcome is in Z_0 is p, $0 < p < 1$. Let $Z_1 = Z \backslash Z_0$.
- Alice and Bob repeat the following until they are "satisfied":
  - Alice sends x to $\mathcal{G}$ twice.
  - If in either instance, the output from $\mathcal{G}$ is not from the support of the distribution corresponding to x, Bob aborts the protocol. Note that since $\mathcal{G}$ is a publicly-selectable source, this essentially forces Alice to either send an input equivalent to x or probabilistically abort.
  - Else, if exactly one of the outputs is from Z_0 and one from Z_1 then the parties are satisfied
- If the first and second outputs in the last pair of invocations of $\mathcal{G}$ were in Z_0 and Z_1 respectively, the common output is 0; else (the outputs where in Z_1 and Z_0 respectively) the common output is 1.

Since p is a constant independent of the security parameter, this protocol runs in expected constant number of rounds, and except with negligible probability, ends in a polynomial number of rounds. □

4.1 Proof of Theorem 1

We start with the following classification of 2-party (randomized) SSFE functionalities.

Lemma 2. *Every 2-party SSFE functionality falls into one of the following categories.*

1. *UC-reduces to $\mathcal{F}_{\mathsf{coin}}$ (in the computationally unbounded setting).*
2. *An oblivious sampling functionality.*
3. *A functionality with bi-directional influence.*

Proof sketch: This follows from the partitioning of SSFE functionalities into (a) uninfluenced functionalities, (b) functionalities with unidirectional influence, and (c) those with bidirectional influence (see Section 2). As noted in Section 2, an uninfluenced SSFE functionality amounts to sampling from a fixed distribution, and this readily UC-reduces to $\mathcal{F}_{\mathsf{coin}}$. Hence functionalities of type (a) fall into Category 1. For a functionality $\mathcal{F}$ of type (b), if $\mathcal{F}$ is a publicly-selectable source, then again it falls into Category 1, by the first part of Lemma 1. On the other hand if $\mathcal{F}$ of type (b) is not a publicly-selectable source, then as pointed out after Definition 2, it is an oblivious sampling functionality and hence falls into Category 2. Finally (c) is the same as Category 3. □

Given the above classification we prove Theorem 1 by considering functionalities in each of the above categories separately.

- *Category 1.* Since $\mathcal{F}$ in this category is UC-reducible to $\mathcal{F}_{\mathsf{coin}}$ in the computationally unbounded setting, the condition in the theorem is satisfied.
- *Category 2.* If $\mathcal{F}_{\mathsf{coin}}$ is useful for UC-securely realizing a functionality $\mathcal{F}$ in this category, and therefore in particular $\mathcal{F}$ UC-securely reduces to $\mathcal{F}_{\mathsf{coin}}$, then below we shall give a semi-honest secure OT protocol.
- *Category 3.* In [8] it has already been shown that if a functionality in this category reduces to $\mathcal{F}_{\mathsf{coin}}$, then there exists a semi-honest secure OT protocol.

Thus to complete the proof of Theorem 1 it remains to show the following.

Lemma 3. *If an oblivious sampling functionality $\mathcal{F}$ has a UC-secure protocol in the $\mathcal{F}_{\mathsf{coin}}$-hybrid model, then there exists a semi-honest secure OT protocol.*

Proof. Since $\mathcal{F}$ is an oblivious sampling functionality, it is an SSFE functionality $\mathcal{F}_f$ with unidirectional influence (w.l.o.g, assume that Alice's input influences Bob's output) such that there exist two non-redundant inputs $x_0, x_1 \in X$ and an output $z \in Z$, such that the distributions $f(x_0) \neq f(x_1)$ and z falls in the intersection of the supports of $f(x_0)$ and $f(x_1)$.

Suppose π is a protocol in $\mathcal{F}_{\mathsf{coin}}$-hybrid that securely realizes $\mathcal{F}$. Before we specify and analyze our protocol, we elaborate on what it means for π to securely realize $\mathcal{F}$: there exists a simulator $\mathcal{S}_\pi^A$, such that for any environment and corrupt Alice, it will be indistinguishable whether Alice is taking part in an execution of π or Alice is taking part in an execution simulated by $\mathcal{S}_\pi^A$. (Similarly, there is a simulator $\mathcal{S}_\pi^B$ for corrupt Bob.) This simulator $\mathcal{S}_\pi^A$ behaves as follows: it interacts with corrupt Alice simulating to her Bob's messages in π, while also interacting with the ideal functionality $\mathcal{F}$ playing Alice's role. At some point $\mathcal{S}_\pi^A$ would send an input to $\mathcal{F}$ on behalf of Alice, and obtain an outcome (which Bob also obtains and outputs to the environment). We use the following observation about the input that $\mathcal{S}_\pi^A$ sends to $\mathcal{F}$, when corrupt Alice follows the protocol π honestly. Here, two inputs x and x' are called *equivalent* if the distributions $f(x)$ and $f(x')$ are identical.

Claim. Consider the ideal execution involving a corrupt Alice, $\mathcal{S}_\pi^A$ and the ideal functionality $\mathcal{F}$. If corrupt Alice follows π honestly using a non-redundant input x, then the input that $\mathcal{S}_\pi^A$ sends to $\mathcal{F}$ is, except with negligible probability, equivalent to x.

Proof. Let $\alpha_{x'}$ be the probability with which $\mathcal{S}_\pi^A$ sends the input x' to $\mathcal{F}$. Then the resulting output distribution is $\sum_{x' \in X} \alpha_{x'} f(x')$. However, for the simulation to be good, we require this to be negligibly different from $f(x)$. Consider the set X' of all inputs not equivalent to x. Since x is not redundant, $f(x)$ lies outside the convex hull of the set of distributions $\{f(x') | x' \in X'\}$. Since the probabilities are constant (independent of the security parameter), the Euclidean distance between $f(x)$ (considered a point in the space $\mathbb{R}^{|Z|}$) and this convex hull is some constant, say ℓ. Then, the distribution $\sum_{x' \in X} \alpha_{x'} f(x')$ has a Euclidean distance

of at least $\ell(\sum_{x' \in X'} \alpha_{x'})$ from $f(x)$. Since this distance must be negligible (as the Euclidean distance is at most twice the statistical distance), and ℓ is constant, it must be that $\sum_{x' \in X'} \alpha_{x'}$ is negligible. In other words, except with negligible probability $\mathcal{S}_\pi^A$ sends an input equivalent to x, completing the proof of the claim. □

To show that there exists a semi-honest secure protocol for OT, we shall show that there is such a protocol for the functionality $\mathcal{F}_{\mathsf{uni\text{-}AND}}$, which takes a bit each from Alice and Bob and outputs their logical AND to Bob (Alice gets an empty output). (This is enough since it is easy to see that in the semi-honest case OT reduces to $\mathcal{F}_{\mathsf{uni\text{-}AND}}$.) Consider the following protocol for $\mathcal{F}_{\mathsf{uni\text{-}AND}}$.

Let Alice's input be $x^* \in \{0,1\}$ and Bob's input be $y^* \in \{0,1\}$. Let π be a UC-secure protocol for an oblivious sampling functionality $\mathcal{F}$, with two non-redundant inputs x_0 and x_1 such that $f(x_0) \neq f(x_1)$ and z is in the support of both $f(x_0)$ and $f(x_1)$.

For $i = 1$ to k

Until Alice and Bob are "satisfied"

Alice picks $b_i \leftarrow \{0,1\}$, and executes π with Bob, with Bob implementing $\mathcal{F}_{\mathsf{coin}}$, with input $x^i := x_{b_i}$

If $y^* = 0$, then

Bob executes the protocol π with Alice, implementing $\mathcal{F}_{\mathsf{coin}}$ himself, and obtains output $\hat{z}$.

Else ($y^* = 1$),

Bob runs the simulator $\mathcal{S}_\pi^A$ for a corrupt Alice in π, until the simulator extracts an input $\hat{x}^i$; the simulator expects a response from $\mathcal{F}$ on sending this input to it.

Bob samples $\hat{z}$ from $f(\hat{x}^i)$, and feeds this back to the simulator as the output from $\mathcal{F}$.

Bob continues executing the simulator until the end of the protocol.

If $\hat{z} = z$ then Alice and Bob are satisfied, else not.

Alice sends $w = x^* \oplus b_1 \oplus b_2 \oplus \ldots \oplus b_k$ to Bob.

If $y^* = 0$ Bob outputs 0, else he outputs $w \oplus \hat{b}_1 \oplus \hat{b}_2 \oplus \ldots \oplus \hat{b}_k$, where the bit $\hat{b}_i$ is 0 iff $\hat{x}^i = x_0$.

We shall argue that if π is a secure protocol for $\mathcal{F}$, then this protocol is a semi-honest secure protocol for $\mathcal{F}_{\mathsf{uni\text{-}AND}}$ in the PPT setting.

Firstly, we show that the protocol is correct: for any pair of inputs, the outputs of the protocol is the same as that of the ideal functionality $\mathcal{F}_{\mathsf{uni\text{-}AND}}$. Alice produces an empty output in the protocol and in the ideal execution. When $y^* = 0$, Bob's output is 0 in both cases. It only remains to analyze the case when $y^* = 1$. For this case, we argue that in the protocol, $\hat{x}^i = x^i$ for all i (except with negligible probability), so that Bob's ouput is indeed x^* as it will be in the ideal execution. This follows from the above claim regarding the correctness of the input x extracted by the simulator $\mathcal{S}_\pi^A$, and a union bound over $i = 1$ to k.

It remains to consider the case when exactly one of Alice and Bob is passively corrupt (the case when both are corrupt being trivial). In each case, we need to show that the view of the corrupt party can be simulated based on the corrupt party's input and output (and given those, independent of the input of the other party).

If Alice is corrupt, consider a simulator which simply runs our protocol with Bob's input set to (say) 0, and sends Alice's input to $\mathcal{F}_{\mathsf{uni\text{-}AND}}$. By the correctness of the protocol, we need only argue that the view of Alice is nearly the same as in the simulation for $y^* = 0$ and $y^* = 1$. Clearly this is true when $y^* = 0$. On the other hand, Alice's view is nearly identical when $y^* = 1$ and $y^* = 0$ by the indistinguishability guarantee of the simulator $\mathcal{S}_\pi^A$.

If Bob is corrupt, consider the following (semi-honest) simulation. If $y^* = 1$, then the simulator sends 1 to $\mathcal{F}_{\mathsf{uni\text{-}AND}}$ and obtains x^* in response; then it faithfully runs our protocol with Alice's input set to x^*. If $y^* = 0$, then the simulator obtains no information from $\mathcal{F}_{\mathsf{uni\text{-}AND}}$; in this case it simply picks an arbitrary input for Alice, say 0, and runs our protocol faithfully. Note that this has the effect that the last message sent in the protocol when $x^* = 1$ could be wrongly distributed. However we argue that the last message when $x^* = 1$ is nearly identically distributed as when $x^* = 0$, conditioned on Bob's view in the rest of the protocol. For this, we first replace each execution of π in our protocol as well as in our simulation with a simulation using $\mathcal{S}_\pi^B$ interacting with an instance of the ideal functionality $\mathcal{F}$. This causes a negligible change in the two distributions. Then, for an execution of π, conditioned on Bob's view (in which the only information about each b_i is the fact that the response from the ideal functionality $\mathcal{F}$ is z), $p := \Pr[b_i = 0] = \Pr[f(x_0) = z]/(\Pr[f(x_0) = z] + \Pr[f(x_1) = z])$, and $\Pr[b_i = 1] = 1 - p$ (independently for each i), for some constant (i.e., independent of k) p, with $0 < p < 1$. Then $|\Pr[\bigoplus_{i=1}^k b_i = 0] - \Pr[\bigoplus_{i=1}^k b_i = 1]| = |(p - (1-p))^k|$ is negligible, or in other words $\bigoplus_{i=1}^k b_i$ is close to a uniformly distributed bit. Thus the last message sent out by Alice is nearly identically distributed for $x^* = 0$ and $x^* = 1$. □

4.2 Proof of Theorem 3

Theorem 3 extends Theorem 1 to allow any publicly-selectable source $\mathcal{G}$ in place of $\mathcal{F}_{\mathsf{coin}}$. We show that it follows from Theorem 1 and Lemma 1. If $\mathcal{F}$ reduces to a publicly-selectable source $\mathcal{G}$ in the PPT setting, then using Lemma 1, $\mathcal{F}$ reduces to $\mathcal{F}_{\mathsf{coin}}$ in the PPT setting. By Theorem 1, then either there is a semi-honest secure protocol for OT, or $\mathcal{F}$ reduces to $\mathcal{F}_{\mathsf{coin}}$ in the computationally unbounded setting. In the former case we are done. In the latter case, if $\mathcal{G}$ is not trivial then by Lemma 1 again, we have that $\mathcal{F}$ reduces to $\mathcal{G}$ in the computationally unbounded setting. On the other hand, if $\mathcal{G}$ is trivial, then so must $\mathcal{F}$ be (since only trivial functionalities can be reduced to trivial functionalities, even in the PPT setting [12]), and then again $\mathcal{F}$ reduces to $\mathcal{G}$ in the computationally unbounded setting.

5 Conclusions and Open Problems

This work closes a gap left in the recent work of [8], thereby characterizing the computational assumption necessary and sufficient for reducing *any* SSFE functionality to $\mathcal{F}_{\mathsf{coin}}$ (or to any publicly-selectable source). The main technical contribution in this work is to identify a new class of functionalities called oblivious sampling functionalities, and to provide a semi-honest secure protocol for OT, assuming that an oblivious sampling functionality reduces to $\mathcal{F}_{\mathsf{coin}}$.

Despite our complete understanding regarding the question of reduction to $\mathcal{F}_{\mathsf{coin}}$ and other publicly-selectable sources, several other aspects of randomized SSFE functionalities remain relatively less understood. In particular, the question of reduction to functionalities other than publicly-selectable sources remains unexplored. Also, we leave open the question of extending our current characterization to functionalities beyond SSFE functionalities. Finally, by considering reductions under other security notions (like semi-honest security), we come across more open problems for randomized functionalities. In particular, it is not known which SSFE functionalities are trivial in the semi-honest security model.

We hope that our techniques – especially, the identification of oblivious sampling functionalities, and the resulting classification of SSFE functionalities – will add to the tools that will aid in resolving these questions.

Acknowledgments

We would like to thank Pichayoot Ouppaphan for being part of this work in its initial stages. This material is based upon work supported by the National Science Foundation, USA under grant CNS 07-47027. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

1. 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.
2. 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.
3. I. Haitner. Semi-honest to malicious oblivious transfer - the black-box way. In R. Canetti, editor, *TCC*, volume 4948 of *Lecture Notes in Computer Science*, pages 412–426. Springer, 2008.
4. J. Kilian. Founding cryptography on oblivious transfer. In *STOC*, pages 20–31. ACM, 1988.

5. J. Kilian. More general completeness theorems for secure two-party computation. In *Proc. 32th STOC*, pages 316–324. ACM, 2000.
6. D. Kraschewski and J. Müller-Quade. Completeness theorems with constructive proofs for finite deterministic 2-party functions. In Y. Ishai, editor, *TCC*, volume 6597 of *Lecture Notes in Computer Science*, pages 364–381. Springer, 2011.
7. 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.
8. H. K. Maji, P. Ouppaphan, M. Prabhakaran, and M. Rosulek. Exploring the limits of common coins using frontier analysis of protocols. In Y. Ishai, editor, *TCC*, volume 6597 of *Lecture Notes in Computer Science*, pages 486–503. Springer, 2011. Full version at http://eprint.iacr.org/.
9. 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.
10. 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.
11. 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.
12. 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.

A Security Definitions

We follow standard conventions and terminology for defining security of protocols for multi-party computation tasks. For easy reference we reproduce these definitions as given in [8], whose results we improve up on.

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 in every environment. Depending on the nature or adversary/simulator and the environment, we consider three different kinds of security notions.

- A semi-honest or passive adversary (in the real or ideal execution) is one which is not allowed to deviate from the (real or ideal) protocol. *Semi-honest* or *passive security* is achieved if for every semi-honest adversary in the real world there is a semi-honest simulator in the ideal world as above.
- A standalone environment is one which does not interact with the adversary during the execution of the protocol. *Standalone security* is achieved if we restrict the security requirement to standalone environments; in this case the simulator can rewind the adversary without the environment detecting it. In this work we do not consider this notion of security.

– *Universally composable (UC) security* [2] is achieved when the security requirement is met against all adversaries (possibly active) and all environments (possibly not standalone); the simulator is allowed to be an active adversary. In this case there must exist a straight-line blackbox simulation (i.e., the simulator internally runs the adversary as a blackbox and never rewinds it).

In this work, we exclusively consider *static* adversaries, who do not adaptively corrupt honest parties during the execution of a protocol.

PPT vs. computationally unbounded setting. In the PPT setting we restrict all entities – the environment, the adversary and simulator – to probabilistic polynomial time computation. In the computationally unbounded setting all these entities can be computationally unbounded. (However, for the purpose of the results in this work, one could require the simulator in the computationally unbounded setting to be efficient (PPT) with blackbox access to the adversary. Then, if a protocol is secure in the computationally unbounded setting, it will be secure in the PPT setting too.

Hybrids. 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.

Reduction. We say that a functionality $\mathcal{F}$ *reduces* to a functionality $\mathcal{G}$ if $\mathcal{F}$ can be UC-securely realized in the $\mathcal{G}$-hybrid. In the real world protocol, that parties have access to a trusted implementation of $\mathcal{G}$ in addition to the secure point-to-point communication channel to securely realize $\mathcal{F}$. Suppose π is a UC-secure protocol for $\mathcal{F}$ in the $\mathcal{G}$-hybrid. Then, parties generate a transcript based on their local views and π can also call the trusted $\mathcal{G}$ implementation. The functionality $\mathcal{G}$ can be any arbitrary functionality, i.e. it need not be a two party function, parties need not play fixed roles while calling $\mathcal{G}$ and, in fact, both parties can provide multiple inputs while performing a call to $\mathcal{G}$.

UC-security in Hybrid worlds. As mentioned earlier, we shall only consider static corruption of parties, i.e. at the beginning of an execution the adversary announces which party it wants to corrupt and cannot corrupt any further party during the execution of the protocol. To show that a protocol π is a UC-secure realization of $\mathcal{F}$ in the $\mathcal{G}$ hybrid, we need to show that for every adversarial strategy in the $\mathcal{G}$-hybrid there exists a simulator in the Ideal world such that any environment is unable to distinguish the Real execution from the Ideal execution. In this work, we shall restrict ourselves to reductions where both $\mathcal{F}$ and

$\mathcal{G}$ are (at most) two party functionalities. Henceforth, we present the definition restricted to this particular case. Suppose Alice is corrupted by the adversary and Bob is honest. The simulator S_π^A for Alice in the Ideal execution, interacts with the adversarial Alice so that no environment can distinguish the Real from the Ideal execution. The simulator also forwards communication between adversarial Alice and the environment. Note, in particular, it implies that the simulator cannot be rewinding in this case. During this execution, the calls to the $\mathcal{G}$ functionality made by the adversarial Alice is answered by the simulator S_π^A. At some point during the interaction with adversarial Alice, the simulator sends an input x to the ideal functionality $\mathcal{F}$ and receives and answer z. The simulator continues the execution with the adversarial Alice and terminates after generating a complete transcript (we can assume that the adversarial Alice strategy always completes a protocol).

If there exists an efficient S_π^A which can make the Ideal execution indistinguishable from the Real execution to any environment, then $\mathcal{F}$ is secure in the $\mathcal{G}$-hybrid when Alice is corrupt. Additionally, if there exists an efficient simulator S_π^B which shows that $\mathcal{F}$ is secure in the $\mathcal{G}$-hybrid when Bob is corrupt then π is a UC-secure protocol for $\mathcal{F}$ in the $\mathcal{G}$-hybrid. Intuitively, the existence of a simulator shows that any effect achieved by the adversarial party could be reflected in the Ideal world itself. The additional power of the simulator lies in the fact that it receives the calls to $\mathcal{G}$, i.e. it gets to see the input sent by the adversarial party to $\mathcal{G}$, and it decides the reply to this call. So, for example, when $\mathcal{G}$ is $\mathcal{F}_{\mathsf{coin}}$, the simulator can determine all the coin outcomes at the beginning of the execution and this could provide additional power to the simulator over the parties in the $\mathcal{G}$-hybrid. Another example is when $\mathcal{G}$ is an oblivious sampling functionality, so it is not possible to be always certain of the input to $\mathcal{G}$ just from the output given by $\mathcal{G}$, the simulator gets additional information when it sees the query made to $\mathcal{G}$.

B Representative Functionalities

It is instructive to consider some representative examples of SSFE functionalities to form an intuition about the classes of functionalities under consideration. The functions will be represented by a matrix where the (i, j)-th entry represents the output distribution when Alice uses input i and Bob uses input j. A distribution is represented as a vector where the k-th entry represents the probability of the k-th output symbol. In most of the examples below, Bob has a single possible input, and hence the matrix has a single column.

Influence of Inputs. Below we list three functionalities with no influence, and two functionalities with uni- and bi- directional influence respectively. The first function corresponds to $\mathcal{F}_{\mathsf{coin}}$– it is an inputless function that outputs an unbiased coin. The second function is a communication channel which sends Alice's input to Bob. Finally, the third function represents the function in which Alice and Bob each has a bit as input and the functionality provides them the XOR of the bits.

$$(\langle 1/2, 1/2\rangle) \qquad \begin{pmatrix}\langle 1,0\rangle\\ \langle 0,1\rangle\end{pmatrix} \qquad \begin{pmatrix}\langle 1,0\rangle & \langle 0,1\rangle\\ \langle 0,1\rangle & \langle 1,0\rangle\end{pmatrix}$$

Publicly-selectable sources. If the output of the function uniquely determines the input of the function (after we have already removed redundant inputs), then the function is a publicly-selectable source. In other words, in a publicly-selectable source the output distributions of the non-redundant inputs should have disjoint supports.

$$\begin{pmatrix}\langle 1,0\rangle\\ \langle 0,1\rangle\end{pmatrix} \qquad \begin{pmatrix}\langle 1/2, 1/2, 0, 0\rangle\\ \langle 0, 0, 3/4, 1/4\rangle\end{pmatrix}$$

Oblivious sampling. Following are some oblivious sampling functionalities.

$$\begin{pmatrix}\langle 1/2, 1/2\rangle\\ \langle 1/4, 3/4\rangle\end{pmatrix} \qquad \begin{pmatrix}\langle 1/2, 1/2\rangle\\ \langle 1,0\rangle\end{pmatrix} \qquad \begin{pmatrix}\langle 1/2, 1/2, 0\rangle\\ \langle 0, 1/2, 1/2\rangle\end{pmatrix}$$

Redundancies. Consider the following function $\mathcal{F}$ with unidirectional influence:

$$\begin{pmatrix}\langle 1,0\rangle\\ \langle 1/2, 1/2\rangle\\ \langle 0,1\rangle\end{pmatrix}$$

This function is not an oblivious sampling function, even though the output distributions from different inputs have overlapping support. This is because the second row of this matrix represents a redundant input as it can be expressed as a convex linear combination of the first and the last rows. So, after removal of the redundant input, the function is seen to be a publicly-selectable source.