Complexity of Multi-party Computation Problems: The Case of 2-Party Symmetric Secure Function Evaluation*

Hemanta K. Maji, Manoj Prabhakaran, and Mike Rosulek

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

Abstract. In symmetric secure function evaluation (SSFE), Alice has an input x, Bob has an input y, and both parties wish to securely compute $f(x, y)$. We show several new results classifying the feasibility of securely implementing these functions in several security settings. Namely, we give new alternate characterizations of the functions that have (statistically) secure protocols against passive and active (standalone), computationally unbounded adversaries. We also show a strict, infinite hierarchy of complexity for SSFE functions with respect to universally composable security against unbounded adversaries. That is, there exists a sequence of functions $f_1, f_2, \ldots$ such that there exists a UC-secure protocol for f_i in the f_j-hybrid world if and only if $i \leq j$.

The main new technical tool that unifies our unrealizability results is a powerful protocol simulation theorem, which may be of independent interest. Essentially, in any adversarial setting (UC, standalone, or passive), f is securely realizable if and only if a very simple (deterministic) "canonical" protocol for f achieves the desired security. Thus, to show that f is unrealizable, one need simply demonstrate a single attack on a single simple protocol.

1 Introduction

In the classical setting of secure two-party computation, Alice and Bob have private inputs x and y respectively, and they want to to jointly compute a common value $f(x, y)$ in a secure way. Starting from Yao's millionaire's problem [21], such *symmetric secure function evaluation* (SSFE) problems have remained the most widely studied multi-party computation problems, in many security models. SSFE problems are fully specified by their associated function tables (i.e., a matrix M with $M_{x,y} = f(x, y)$); studying this matrix can tell us everything about the corresponding SSFE problem. Despite this apparent simplicity, and several works carefully exploring SSFE problems, the landscape of such problems has remained far from complete.

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-3-642-00457-5_36

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

O. Reingold (Ed.): TCC 2009, LNCS 5444, pp. 256–273, 2009.

On "cryptographic complexity." One expects different cryptographic tasks to have different levels of cryptographic sophistication. For instance public-key encryption is more complex than symmetric-key encryption. One indication of this is that in a computationally unbounded setting one-time pads provide (a limited version of) symmetric-key encryption, but public-key encryption is simply impossible. Impagliazzo and Rudich [8] provide a separation between the complexity of these two primitives by demonstrating another primitive (namely random oracles) which is sufficient to realize (full-fledged) symmetric-key encryption, but not enough for public-key encryption (in, say, a computationally unbounded setting). Our goal in this work is to understand such complexity separations among 2-party SSFE functionalities (and more generally, among multi-party computation functionalities).

The natural tool for comparing qualitative complexity of two tasks is a *reduction*. In the context of cryptographic feasibility, the most natural reduction is a black-box security reduction — we say that an SSFE functionality f reduces to another functionality g if it is possible to securely realize f using calls to a trusted party that securely implements g. As in computational complexity, the most fine-grained distinctions in complexity are made by considering the most restricted kinds of reductions. In fact, fine-grained complexity distinctions often disappear when using more generous reductions. In this work, we consider a very strong formulation of black-box security reduction: universally composable security in computationally unbounded environments against active (and adaptive) adversaries.

Our goal is to identify broad complexity classes of SSFE functionalities under this notion of reduction. This involves identifying various functionalities that reduce to each other, and — this is typically the more difficult part — establishing separations (non-reducibility) between functionalities. A complexity class can be *understood* by providing alternate (say combinatorial) characterizations of the functionalities in the class. Another approach to understanding a class is to identify a *complete* functionality, which provides a concrete embodiment of the complexity of all functionalities in that class. Conversely, the inherent cryptographic qualities of a given functionality can be understood by studying its "degree," namely, the class of all functionalities that can be reduced to it. We pursue all these approaches in this paper.

Finally, a systematic study of multi-party computation functionalities, under a stringent notion of reduction, unifies several prior advances in different security models. In particular, the two main classes that we identify and combinatorially characterize in this paper, which are downward-closed under this reduction, correspond to realizability in weaker security models — standalone security and passive security (a.k.a, semi-honest or honest-but-curious security). We emphasize that in plotting these classes in our complexity map, we do not change our notion of reduction.

Our results only start to unveil the rich landscape of cryptographic complexity of multi-party computation functionalities. We believe it will be of great theoretical — and potentially practical — value to further uncover this picture.

1.1 Previous Work

Cryptographic complexity of multi-party computation functionalities (though not necessarily under that name) has been widely studied in various security models. We restrict our focus mostly to work on 2-party SSFE functions in computationally unbounded settings. Complexity questions studied were limited to realizability (least complex), completeness (most complex) and whether there exist functions of intermediate complexities.

Realizability. The oldest and most widely studied model for SSFE is security against passive (honest-but-curious) adversaries. Chor and Kushilevitz [6] characterized SSFE functions with *boolean output*. Beaver [2] and Kushilevitz [17] independently extended this to general SSFE function, but restricted to the case of *perfect* security (as opposed to statistical security). These characterizations were given in the standalone security model, but do extend to the the universal composition (UC) framework [4] that we use.

However, in the case of security against active (a.k.a malicious) adversaries, demanding composability does affect realizability. The following hold for both computationally bounded and unbounded settings. In the UC-setting, Canetti et al. [5] characterized securely realizable SSFE functions as those in which the function is insensitive to one party's input. Lindell [18] showed that UC security is equivalent to concurrent self-composable security, for a class of functionalities that includes SSFE. But Backes et al. [1] gave an example of a function that is realizable in the standalone setting, but not in the UC-setting. The problem of identifying all such functions remained open.

Completeness. The question of completeness for SSFE was essentially settled by Kilian [12], who showed that a function is complete if and only if it contains a generalized "OR-minor." This relies on the completeness of the SFE functionality of oblivious transfer, a result originally proven in [10], and proven in the UC setting in [9].[1] The reduction in [12] was reconsidered in the UC setting by [15].

Intermediate Complexities. In some security settings, there are only two distinct levels of complexity: the realizable tasks and the complete tasks, with nothing in between. Indeed, such a dichotomy holds in the case of *asymmetric* SFE (in which only one party receives any output), both for passive [3] and active security [13],[2] and also in the case of passive security for *boolean output* SSFE [14]. In [20] it is conjectured that such a dichotomy holds for general functionalities *in a computationally bounded setting*. However, there is no such simple dichotomy in the setting of SSFE. Indeed the characterizations of complete and realizable

[1] The protocol in [10] is not UC-secure, but an extension presented in [11] is likely to be.

[2] [3] also considers a notion of active security for computationally bounded setting, and extends their dichotomy using a stronger notion of realizability and a weaker, non-black-box notion of completeness; this result draws the line between realizability and completeness differently from [13]. The dichotomy does not extend to the UC-setting.

SSFE functions [12,5] leaves much gap between them. Further, [2,17,14] give an example SSFE function which is neither complete nor even passively realizable.

1.2 Our Results

A visual overview of our results is given in Figure 1.

First, we show that SSFE functions with *perfect* passive-secure protocols (as characterized by Beaver and Kushilevitz) are exactly those with *statistically secure*, passive-secure protocols (Theorem 3). They are also exactly the functions that are UC-realizable in the $\mathcal{F}_{\mathsf{com}}$-hybrid world — i.e., realizable against active adversaries in the UC framework, using calls to an ideal commitment functionality (Theorem 4). Thus, $\mathcal{F}_{\mathsf{com}}$ exactly captures the difficulty of passively realizing SSFE functions (it cannot be said to be complete, since it is not SSFE itself). We also show that the perfectly secure deterministic protocols used by Kushilevitz achieve optimal round complexity, even among randomized, statistically secure protocols (Corollary 2).

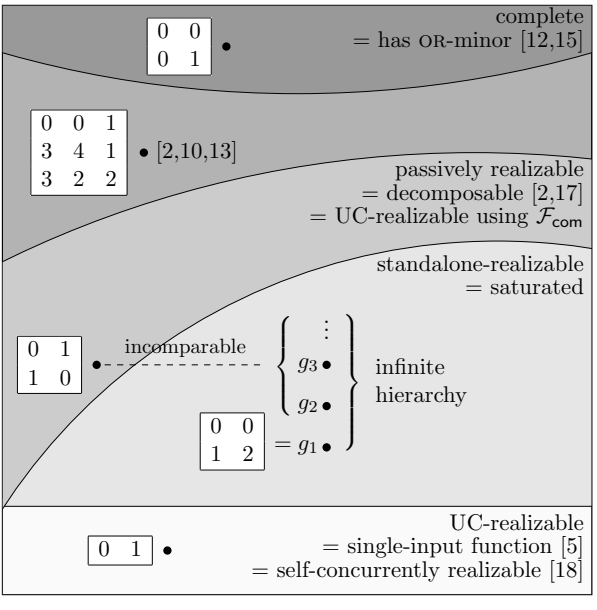

Fig. 1. Cryptographic complexity of 2-party SSFE

Next, we give an explicit and simple combinatorial characterization of the standalone-realizable SSFE functions, as the uniquely decomposable functions which are "maximal" (Theorem 5). We call such functions *saturated* functions. We also show that every SSFE function which is standalone-realizable but not UC-realizable also has no protocol secure under concurrent self-composition (Theorem 6), strengthening a negative result from [1], and yielding a much simpler proof of Lindell's characterization [18] for the special case of SSFE.

Finally, we focus our investigation on the vast area between the two complexity extremes of completeness and UC-realizability. We leverage ideas from passive security and standalone security to obtain a new technique for obtaining impossibility results in the UC framework. Namely, we describe a purely combinatorial criterion of two functions f, g (which has to do with the round complexity of realizing f and g) which implies that there is no UC-secure protocol for f that uses calls to an ideal functionality for g. (Theorem 7).

We apply this new separation technique to obtain several new results. We first demonstrate an infinite hierarchy of complexity — that is, SSFE functions $g_1, g_2, \ldots$ such that there is a secure protocol for g_i using calls to an ideal g_j if and only if $i \leq j$ (Corollary 8). We also show that there is no complete function (complete under our strong notion of reduction) for the class of standalone-realizable or passively realizable SSFE functions (Corollary 9). Finally, we show that there exist SSFE functions with *incomparable* complexities (Corollary 10), answering an open problem from [20].

We note that our characterizations of passive (Theorem 3) and standalone security (Theorem 5) were also independently discovered by Künzler, Müller-Quade, and Raub [16]. They also extend these results to a multi-party setting, and beyond the symmetric-output case.

About our techniques. The characterization of Beaver and Kushilevitz for passive security gives very simple deterministic protocols for SSFE functions, which we call *canonical protocols.* Our results demonstrate the special privilege of canonical protocols in the universe of SSFE. Our main technical tool is Theorem 1, which strongly formalizes the intuition that *every* protocol for f must disclose information in essentially the same order as the canonical protocol for f. Stated more formally, for any secure protocol π for an SSFE function f, if the canonical protocol for f is unique (up to some isomorphism), then the canonical protocol is "as secure as" π, in the UC simulation sense. That is, for any adversary in the canonical-protocol-real-world, there is an adversary in the π-real-world that achieves the same effect for all environments. Note that standalone security and passive security can both be expressed as restrictions of the UC definitions, so our theorem is applicable to a wide variety of security settings.

Using this powerful theorem, it is quite simple to demonstrate impossibility results for secure realizability. Roughly speaking, an SSFE function f satisfying the above condition is realizable in a given security model if and only if its canonical protocol achieves the desired security. Thus, to show f is unrealizable, we simply describe a feasible attack against the (simple) canonical protocol.

We crucially use Theorem 1 as the unifying component when proving impossibility of secure realization in the standalone (Theorem 5), concurrent self-composition (Theorem 6), and even UC hybrid world settings (Theorem 7).

2 Preliminaries

In this section we present some standard notions, and also introduce some concepts and definitions used in our constructions and proofs.

MPC Problems and Secure Realization. Following the Universal Composition [4] framework, a multi-party computation problem is defined by the code of a trusted (probabilistic, interactive) entity called a *functionality.* A protocol π is said to securely realize an MPC functionality $\mathcal{F}$, if for all (static) active adversaries, there exists a simulator such that for all environments, $|\Pr[\text{EXEC}_{\mathcal{F}} = 1] - \Pr[\text{EXEC}_{\pi} = 1]| \leq \epsilon(k)$, for some negligible function ϵ. Here k is the *security parameter* that is

- Wait for (and record) input x from Alice and input y from Bob.
- When both x and y have been received, if either party is corrupt, then send $(\text{OUTPUT}, f(x, y))$ to the adversary.
- On input DELIVER from the adversary, or if neither party is corrupt, send $f(x, y)$ to both Alice and Bob.

Fig. 2. Functionality $\mathcal{F}_f$: Symmetric Secure Function Evaluation of f with abort

an input to the protocol and simulator. $\text{EXEC}_{\mathcal{F}}$ denotes the environment's output distribution in the "ideal" execution: involving $\mathcal{F}$, the environment, and the simulator; EXEC_{π} denotes the environment's output in the "real" execution: involving an instance of the protocol, the environment, and the adversary. Throughout this paper, we consider a UC model in which all entities are computationally unbounded.

We also consider *hybrid worlds*, in which protocols may also have access to a particular ideal functionality. In a $\mathcal{G}$-hybrid world, parties running the protocol π can invoke up any number of independent, asynchronous instances of $\mathcal{G}$. Regular protocols could be considered to use the (authenticated) communication functionality. If a protocol π securely realizes a functionality $\mathcal{F}$ in the $\mathcal{G}$-hybrid world, we shall write $\mathcal{F} \sqsubseteq \mathcal{G}$. The $\sqsubseteq$ relation is transitive and reflexive, and it provides our basis for comparing the complexity of various functionalities.

We also consider two common restrictions of the security model, which will prove useful in our results. If we restrict the security definition to environments which do not interact with the adversary during the protocol execution (i.e., simulators are allowed to rewind the adversary), we get a weaker notion of security, called *standalone security.* If we restrict to adversaries and simulators which receive an input from the environment and run the protocol honestly, we get a different relaxation of security called *passive security.* Note that in this definition, simulators must behave honestly in the ideal world, which means they simply pass the input directly to the functionality.

Symmetric SFE Functionalities. In this paper we focus exclusively on classifying 2-party *symmetric secure function evaluation* (SSFE) functionalities. An SSFE functionality $\mathcal{F}_f$ is parameterized by a function $f : X \times Y \to Z$, where X, Y, Z are finite sets.[3] The functionality $\mathcal{F}_f$ simply receives the inputs from the two parties, computes f on the inputs and sends the result to both the parties. However, as is standard, we also allow the adversary to first receive the output and block delivery if desired (see Figure 2). For convenience, we shall often use f and $\mathcal{F}_f$ interchangeably.

[3] We restrict our attention to *finite* functions. In particular, the size of the domain of a function does not change with the security parameter. This is in line with previous works. Further, in the computationally unbounded setting, it is reasonable to have the ideal world not involve the security parameter. Nevertheless, all of our results can be extended to the case where f's input domain is polynomially bounded in the security parameter. We can show that this restriction is necessary for most of our results.

2.1 Structure of Functions and Protocols

We say that two functions are isomorphic if each one can be computed using a single call to the other with no other communication (with only local processing of the inputs and output). That is, Alice and Bob can independently map their inputs for f function to inputs for g, carryout the computation for g, and then locally (using their private inputs) map the output of g to the output of f (and vice-versa).

Definition 1 (Function Isomorphism, $\cong$). *Let $f : X \times Y \to D$ and $g : X' \times Y' \to D'$ be two functions. We say $f \leq g$, if there exist functions $I_A : X \to X'$, $I_B : Y \to Y'$, $M_A : X \times D' \to D$ and $M_B : Y \times D' \to D$, such that for any $x \in X$ and $y \in Y$ $f(x,y) = M_A(x, g(I_A(x), I_B(y))) = M_B(y, g(I_A(x), I_B(y)))$. If $f \leq g$ and $g \leq f$ then $f \cong g$ (f is isomorphic to g).*

For example, the XOR function $\boxed{\begin{smallmatrix}0&1\\1&0\end{smallmatrix}}$ is isomorphic to the function $\boxed{\begin{smallmatrix}1&2\\3&4\end{smallmatrix}}$.

Definition 2 (Decomposable [2,17]). *A function $f : X \times Y \to D$ is* row decomposable *if there exists a partition $X = X_1 \cup \cdots \cup X_t$ ($X_i \neq \emptyset$), $t \geq 2$, such that the following hold for all $i \leq t$:*

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

We define being column decomposable *symmetrically with respect to X and Y. We say that f is simply* decomposable *if it is either constant, row decomposable, or column decomposable.*

For instance, $\boxed{\begin{smallmatrix}0&0\\1&2\end{smallmatrix}}$ and $\boxed{\begin{smallmatrix}0&1\\1&0\end{smallmatrix}}$ are decomposable, but $\boxed{\begin{smallmatrix}0&0\\0&1\end{smallmatrix}}$ and $\boxed{\begin{smallmatrix}0&0&1\\3&4&1\\3&2&2\end{smallmatrix}}$ are not.

Note that our definition differs slightly from [2,17], since we insist that row and column decomposition steps strictly alternate. We say that a function f is *uniquely decomposable* if all of its decompositions are equivalent up to re-indexing $X_1, \ldots, X_t$ ($Y_1, \ldots, Y_t$) at each step. Thus $\boxed{\begin{smallmatrix}0&0\\1&2\end{smallmatrix}}$ is uniquely decomposable, but $\boxed{\begin{smallmatrix}0&1\\1&0\end{smallmatrix}}$ is not.

Canonical protocols [2,17]. If f is decomposable, then a *canonical protocol* for f is a deterministic protocol defined inductively as follows:

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

It is a simple exercise to see that a canonical protocol is a perfectly secure protocol for f against passive adversaries (cf. [2,17]).

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

1. At the end of the interaction, both parties include their outputs in the transcript as their last message before terminating, so that each party's output is a function of the transcript alone (i.e., not a function of their input and random tape, etc.). Since both parties should receive the same output, this is without loss of generality, even for standalone or UC security.

2. If $u_1, u_2, \ldots$ are the messages exchanged in a run of the protocol, then $(u_1, u_2, \ldots)$ can be uniquely and unambiguously obtained from the string $u_1 u_2 \cdots$.

3. The honest protocol does not require the parties to maintain a persistent state besides their private input (in particular, no random tape). Instead, the protocol is simply a mapping $P : \{0,1\}^* \times \{0,1\}^* \times \{0,1\}^* \to [0,1]$, indicating that if t is the transcript so far, and a party's input is x, then its next message is u with probability $P(t, x, u)$. In other words, randomness can be sampled as needed and immediately discarded. This requirement is without loss of generality for computationally unbounded parties.[4]

Deviation revealing. In [20] it is shown that for a class of functionalities called "deviation revealing", if a protocol π is a UC-secure realization of that functionality, then the same protocol is also secure against passive adversaries. Note that this property is not true in general for all SFE functionalities. For example, the SFE where Alice gets no output but Bob gets the boolean-OR of both parties' inputs is not passively realizable. However, the protocol where Alice simply sends her input to Bob is UC-secure, since a malicious Bob can always learn Alice's input in the ideal world by choosing 0 as its input to the functionality. It turns out SSFE functions are deviation revealing. We include an adapted version of the argument in [20]:

Lemma 1 ([20]). *Let π be a UC-secure (perhaps in a hybrid world) or a standalone-secure protocol for an SSFE f. Then π itself is a passive-secure protocol for f as well (in the same hybrid world as π).*

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

Suppose $\mathcal{S}$ is the simulator for $\mathcal{A}$. In the ideal world, both parties produce output with overwhelming probability, and so $\mathcal{S}$ must also allow the other party

[4] Note that because of this, security against adaptive corruption and static corruption are the same for this setting.

to generate output in the ideal world with overwhelming probability. Thus with overwhelming probability, $\mathcal{S}$ must receive x from the environment, send some x' to the ideal functionality f, receive the output $f(x', y)$ and deliver the output. Without loss of generality, we may assume $\mathcal{S}$ does so with probability 1.

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

3 Simulation of Canonical Protocol in a General Protocol

In this section, we develop our main new technical tool, the protocol simulation theorem. Throughout the section we fix an SSFE f with domain $X \times Y$, and fix a secure protocol π for f.

Definition 3. *We say that x, x', y, y' forms a* ⊞-minor (resp. ⊟-minor) *in f if:*

$$\begin{array}{ccc} f(x,y) & = & f(x,y') \\ \neq & & \neq \\ f(x',y) & \neq & f(x',y') \end{array} \qquad \left(\text{resp. if } \begin{array}{ccc} f(x,y) & \neq & f(x,y') \\ = & & \neq \\ f(x',y) & \neq & f(x',y') \end{array} \right)$$

In the canonical protocol for a function that is entirely a ⊞-minor, Alice must completely reveal her input before Bob reveals (anything about) his input. We show that, in general, this intuition carries through for *any* protocol, with respect to *any* embedded ⊞ or ⊟-minor. That is, there must be a point at which Alice has completely made the distinction between two of her inputs, but Bob has not made any distinction between two of his inputs.

Definition 4. *Let* $\Pr[u|x, y]$ *denote the probability that π generates a transcript that has u as a prefix, when executed honestly with x and y as inputs.*

Let F be a set of strings that is prefix-free[5] *Define* $\Pr[F|x, y] = \sum_{u \in F} \Pr[u|x, y]$. *We call F a* frontier *if F is maximal – that is, if* $\Pr[F|x, y] = 1$ *for all x, y. We denote as $\mathcal{D}_F^{x,y}$ the probability distribution over F where $u \in F$ is chosen with probability* $\Pr[u|x, y]$.

Lemma 2 (⊞ Frontiers). *For all $x \neq x' \in X$, there is a frontier F and negligible function ν such that, for all $y, y' \in Y$:*

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

– *if* $f(x,y) \neq f(x',y)$, *then* $SD\left(\mathcal{D}_F^{x,y}, \mathcal{D}_F^{x',y}\right) \geq 1 - \nu(k)$, *and*
– *if* x, x', y, y' *form a* $\boxplus$*-minor, then* $SD\left(\mathcal{D}_F^{x,y}, \mathcal{D}_F^{x,y'}\right), SD\left(\mathcal{D}_F^{x',y}, \mathcal{D}_F^{x',y'}\right) \leq \nu(k)$.

We defer the technical proof of Lemma 2 to the full version [19].

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

Theorem 1 (Protocol Simulation). *If f has a unique decomposition, then for any protocol π for f, the canonical protocol for f is "as secure as" π. That is, for every adversary attacking the canonical protocol, there is an adversary attacking π which achieves the same effect in every environment.*

Proof (Sketch). The proof (presented in [19]) involves a careful inductive generalization of Lemma 2. Consider a step in the decomposition of $X \times Y$, say $X = X_1 \cup \cdots \cup X_k$. Roughly, if the function is uniquely decomposable, then for each $i \neq j$, there is a witnessing $\boxplus$-minor x, x', y, y' with $x \in X_i$, $x' \in X_j$. Thus we may apply Lemma 2 to obtain a frontier with respect to these inputs. We can combine these individual frontiers to obtain a frontier representing the entire decomposition step $X = X_1 \cup \cdots \cup X_k$. We show inductively that the transcript distribution at this combined frontier statistically reveals (in this case) which X_i contains Alice's input, while at the same is nearly independent of any further distinctions among either party's inputs within $X \times Y$.

Given such frontiers, the required simulation is fairly natural. The simulator's job is to simulate the canonical protocol to an adversary $\mathcal{A}$, while interacting with the honest party in the protocol π. The simulator $\mathcal{S}$ simply keeps track of the current step in the decomposition of f, and runs π honestly with any representative input. Each time $\mathcal{S}$ reaches a frontier for a decomposition step by the honest party, the transcript distribution at the frontier statistically determines with great certainty which part of the decomposition the honest party's input belongs to. Thus $\mathcal{S}$ can simulate to $\mathcal{A}$ what the honest party would send next in the canonical protocol. Then the simulator receives the adversary's next move in the canonical protocol. $\mathcal{S}$ changes its own input for π to any input consistent with the canonical protocol transcript so far, if necessary. Since the π-transcript so far is nearly independent of such distinctions among the adversary's input, it is indeed possible to swap inputs to π at this point and maintain a sound simulation. It is also for this reason that we consider protocols to be in a normal form, so that the protocol's next-message function depends only on the (currently considered) input and the transcript so far — i.e., not on any other persistent state. □

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

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

Proof. The proof of Theorem 1 constructs for π a frontier for each step in the decomposition of f (corresponding to each step in the canonical protocol). By the required properties of the frontiers, the transcript for π must visit all the relevant frontiers in order, one *strictly* after the next, with overwhelming probability. □

4 Characterizing Passive Security

In this section, we apply Lemma 2 to extend the characterization of Beaver [2] and Kushilevitz [17] to the case of statistical security. We also show a new characterization of passive security in terms of the ideal commitment functionality.

Theorem 3. *f is decomposable if and only if f has a (statistically secure) passive-secure protocol.*

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

($\Leftarrow$) Suppose f is not decomposable. Without loss of generality, we may assume that f is not even row- or column-decomposable at the top level. Then for each way that Alice might start revealing information about her input, there is a $\boxplus$-minor that witnesses the fact that this information cannot be revealed first — Bob must reveal a particular distinction of his inputs before it is safe for Alice to reveal information this way. However, the converse statement holds as well, and so neither party can be the first to safely reveal any information about their input.

More formally, we suppose a secure protocol exists for f. We consider all $\boxplus$-minors, and take the upper envelope of their associated frontiers from Lemma 2. This new frontier corresponds to the first point at which Alice reveals significant information about her input. Similarly, we construct a frontier corresponding to the first point at which Bob reveals significant information. The properties of Lemma 2 imply that with overwhelming probability, the protocol must visit the first frontier after the second one, *and* visit the second frontier after the first one (i.e., neither party can be the first to significantly reveal information). This is a contradiction, since the two frontiers cannot coincide — they consist of points where different parties have just spoken. Thus no secure protocol is possible. □

Theorem 4. *f has a passive-secure protocol if and only if f has a UC-secure protocol in the $\mathcal{F}_{\mathsf{com}}$-hybrid world, where $\mathcal{F}_{\mathsf{com}}$ denotes the ideal commitment functionality defined in Figure 3.*

- On input (COMMIT, x, P_2) from party P_1, send (COMMITTED, P_1) to party P_2, and remember x.
- On input REVEAL from party P_1, if x has already been recorded, send x to party P_2.

Fig. 3. Commitment functionality $\mathcal{F}_{\mathsf{com}}$

Proof (Sketch). ($\Leftarrow$) By Lemma 1, any UC-secure protocol π for a symmetric SFE f is also passively secure. There is a trivial passive-secure protocol for $\mathcal{F}_{\mathsf{com}}$ (the committing party sends "COMMITTED" in the commit phase and sends the correct value in the reveal phase). We can compose π with the passive-secure $\mathcal{F}_{\mathsf{com}}$ protocol to obtain a passive-secure protocol for f in the plain model.

($\Rightarrow$) We will give a general-purpose "compiler" from passive-secure protocols to the UC-secure $\mathcal{F}_{\mathsf{com}}$-hybrid protocols. Suppose π is a passive-secure protocol for f, in normal form. In fact, we need to consider only the canonical protocol for f. Below we consider an arbitrary deterministic protocol π. (In the full version [19] we extend this compiler to randomized protocols as well.)

Suppose Alice's input is $x \in \{1, \ldots, n\}$, and let $\chi \in \{0,1\}^n$ be the associated characteristic vector, where $\chi_i = 1$ if and only if $i = x$. Alice commits to both $\chi_{\sigma(1)}, \ldots, \chi_{\sigma(n)}$ and $\sigma(1), \ldots, \sigma(n)$ for many random permutations σ. For each pair of such commitments, Bob will (randomly) ask Alice to open either all of $\chi_{\sigma(1)}, \ldots, \chi_{\sigma(n)}$ (and verify that χ has exactly one 1) or open all $\sigma(i)$ (and verify that σ is a permutation). Bob also commits to his own input in a similar way, with Alice giving challenges. Then both parties simulate the π protocol one step at a time. When it is Alice's turn in π, she computes the appropriate next message $b \in \{0,1\}$ and sends it. For a deterministic protocol, the next message function is a function of the transcript so far and the input. Given the partial transcript so far t, both parties can compute the set $X_b = \{x' \mid \pi(t, x') = b\}$; i.e., the set of inputs for which the protocol instructs Alice to send b at this point. Then Alice can open enough of her commitments to prove that $\chi_i = 0$ for all $i \in X_{1-b}$ to prove that the message b was consistent with her committed input and the honest protocol. □

Note that much like the well-known GMW compiler [7], we convert an *arbitrary* passive-secure protocol to a protocol secure in a stronger setting. In particular, we do not use the fact that the protocol realizes a functionality with symmetric, deterministic outputs. Thus the compiler may be of more general interest. Unlike the GMW compiler, our compiler operates in an unbounded setting, given ideal commitments. The GMW compiler relies on the existence of commitment protocols and zero-knowledge proofs, while in an unbounded setting, commitment protocols are impossible and zero-knowledge proofs are trivial.

5 Characterizing Standalone Security

From Lemma 1, we know that standalone-realizability for SSFE is a special case of passive-realizability, and hence by Theorem 3, any standalone realizable SSFE

must be decomposable. In this section we identify the additional properties that exactly characterize standalone realizability.

Decomposition strategies. Fix some decomposition of an SSFE function $f : X \times Y \to D$. We define an *Alice-strategy* as a function that maps every row decomposition step $X_0 = X_1 \cup \cdots \cup X_t$ to one of the X_i's. Similarly we define a *Bob-strategy* for the set of column decomposition steps. If A, B are Alice and Bob-strategies for f, respectively, then we define $f^*(A, B)$ to be the subset $X' \times Y' \subseteq X \times Y$ obtained by "traversing" the decomposition of f according to the choices of A and B.

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

Definition 5 (Saturation). *Let f be a uniquely decomposable function. We say that f is* saturated *if $f \cong f^*$.*

To understand this condition further, we provide an alternate description for it. For every $x \in X$ we define an Alice-strategy A_x such that at any row decomposition step $X_0 = X_1 \cup \cdots \cup X_t$ where $x \in X_0$, it chooses X_i such that $x \in X_i$. (For X_0 such that $x \notin X_0$, the choice is arbitrary, say X_1.) Similarly for $y \in Y$ we define a Bob-strategy B_y. Note that in the canonical protocol, on inputs x and y, Alice and Bob traverse the decomposition of f according to the strategy (A_x, B_y), to compute the set $f^*(A_x, B_y)$ (where f is constant). If f is saturated, then all Alice strategies should correspond to some x that Alice can use as an input to f. That is, for all Alice-strategies A, there exists a $x \in X$ such that for all $y \in Y$, we have $f^*(A, B_y) = f^*(A_x, B_y)$; similarly each Bob strategy B is equivalent to some B_y.

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

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

Theorem 5. *f is standalone-realizable if and only if f is saturated.*

Proof (Sketch). ($\Leftarrow$) To securely realize a saturated f, we use its canonical protocol. The simulator for an adversary $\mathcal{A}$ is a rewinding simulator, which does the following: Without loss of generality, assume $\mathcal{A}$ corrupts Alice. First fix a random tape ω for $\mathcal{A}$, then for every Bob-strategy B, run the canonical protocol against $\mathcal{A}$ (using random tape ω), effecting the strategy B. The choices of $\mathcal{A}$ at each step uniquely determine an Alice-strategy. By the saturation of f, A is equivalent to some A_x strategy, and we use x as the adversary's input to f. After receiving the output $f(x, y)$, we simulate the unique canonical protocol transcript consistent with $f(x, y)$.

($\Rightarrow$) We shall use the following lemma (proven, in the full version [19], by induction on the number of steps in the protocol):

Lemma 3. *If π is a 2-party protocol whereby both parties agree on the output, then for any way of coloring the possible outputs red and blue, π has one of the 4 properties:*

1. *A malicious Alice can force a red output and can force a blue output.*
2. *A malicious Bob can force a red output and can force a blue output.*
3. *Both a malicious Bob and malicious Alice are able to force a red output.*
4. *Both a malicious Bob and malicious Alice are able to force a blue output.*

By Lemma 1, we have that f is passively realizable and thus decomposable. We can show that if f is not uniquely decomposable, then there is a way to color its outputs red and blue so that each of the 4 conditions of Lemma 3 for any protocol evaluating f is a violation of security. Thus f must be uniquely decomposable.

Now, since f is uniquely decomposable, Theorem 1 implies that f is standalone-realizable if and only if its canonical protocol is standalone secure. Suppose that f is not saturated. Then there is a (without loss of generality) Alice-strategy A that corresponds to no A_x strategy. The strategy A can be trivially effected by a malicious Alice in the canonical protocol. If Bob's input is chosen at random by the environment, then the same outcome can not be achieved by an ideal-world adversary (who must send an input x to f). Thus the canonical protocol is standalone insecure; a contradiction. Therefore f must be saturated. □

6 Characterizing Concurrent Self-composition

Backes et al. [1] showed that even a perfectly secure protocol with rewinding simulator cannot in general be transformed into even a protocol secure under concurrent self-composition. Recall that in concurrent self-composition, the environment initiates several protocol instances, but does not interact with the adversary during their execution. The adversary also corrupts the same party in all protocol instances. We are able to greatly strengthen their negative result to show that concurrent attacks are the rule rather than the exception:

Theorem 6. *If f is standalone-realizable but not UC-realizable, then f has no protocol secure against concurrent self-composition.*

Proof. A function f is UC-realizable if and only if it is decomposable with decomposition depth 1 [5]. Thus an f as in the theorem statement must be uniquely decomposable (by Theorem 5), with decomposition depth at least 2. We show a concurrent attack against two instances of the *canonical protocol* for f, which by Theorem 1 will imply that f is not realizable under concurrent self-composition.

By symmetry, suppose Alice moves first in the canonical protocol, and let x, x' be two inputs which induce different messages in the first round of the protocol. Let y, y' be two inputs which induce different messages in the second round of the protocol when Alice has input x (thus $f(x, y) \neq f(x, y')$). We consider an environment which runs two instances of f, supplies inputs for Alice for the instances, and outputs 1 if Alice reports particular expected outputs for the two instances. The environment chooses randomly from one of the following three cases:

1. Supply inputs x and x'. Expect outputs $f(x, y')$ and $f(x', y)$.
2. Supply inputs x' and x, Expect outputs $f(x', y)$ and $f(x, y')$.
3. Supply inputs x and x. Expect outputs $f(x, y)$ and $f(x, y)$.

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

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

Put differently, UC-realizability is equivalent to concurrent-self-composition-realizability for SSFE. This is in fact a special case of a theorem by Lindell [18], although our proof is much more direct and requires only 2 instances of the protocol/functionality, as in [1].

7 Finer Complexity Separations

Finally, we develop a new technique for deriving separations among SSFE functions with respect to the $\sqsubseteq$ relation, and apply the technique to concrete functions to inform the landscape of cryptographic complexity.

Theorem 7. *Let $\mathcal{F}$ be any (not necessarily SSFE) UC functionality with a passive-secure, m-round protocol. Let f be an SSFE function with unique decomposition of depth $n > m + 1$.*[6] *Then there is no UC-secure protocol for f in the $\mathcal{F}$-hybrid world; i.e., $f \not\sqsubseteq \mathcal{F}$.*

Proof (Sketch). We use a modified version of Theorem 1. Suppose for contradiction that π is a protocol for f in the $\mathcal{F}$-hybrid world. By Lemma 1, π is also passive-secure in the same setting. Define $\widehat{\pi}$ to be the result of replacing each call to $\mathcal{F}$ in π with the m-round passive-secure protocol for $\mathcal{F}$. $\widehat{\pi}$ is a passive-secure protocol for f in the plain setting. Say that an adversary *behaves consistently* for a span of time if it runs the protocol honestly with an input that is fixed throughout the span.

We mark every $\widehat{\pi}$ transcript prefix that corresponds to a step in the $\mathcal{F}$ subprotocol, except for the first step of that subprotocol. Intuitively, if a $\widehat{\pi}$-adversary behaves consistently during every span of marked transcripts, then that adversary can be mapped to a π adversary that achieves the same effect by interacting with $\mathcal{F}$ appropriately during these points.

[6] We remark that this condition can be tightened to $n > m$ via a more delicate analysis.

Since f is uniquely decomposable, we describe a feasible adversary $\mathcal{A}$ attacking the canonical protocol for f, and apply Theorem 1 to obtain an equivalent adversary $\mathcal{S}$ attacking $\widehat{\pi}$. Theorem 1 always constructs an adversary $\mathcal{S}$ that behaves consistently except for a small number of times where it might swap its input. We will construct $\mathcal{A}$ so that $\mathcal{S}$ only needs to swap its input at most once. If we can ensure that $\mathcal{S}$ will always be able to swap its input at an unmarked point in $\widehat{\pi}$, then $\mathcal{S}$ behaves consistently during the marked spans, so we can obtain an adversary successfully attacking π in the $\mathcal{F}$-hybrid world, a contradiction.

Suppose by symmetry that Alice goes first in the canonical protocol for f. Let x_0, y_0 be inputs that cause the canonical protocol to take a maximum number of rounds, and let y_1 be such that the (unique) transcripts for x_0, y_0 and x_0, y_1 agree until Bob's last message; thus $f(x_0, y_0) \neq f(x_0, y_1)$. Let x_1 be an input for which Alice's first message is different than for x_0.

We consider an environment which chooses a random $b \leftarrow \{0, 1\}$ and supplies x_b as Alice's input. It expects the adversary to detect and correctly report its choice of b. If $b = 0$, then the environment chooses a random $c \leftarrow \{0, 1\}$, and gives c to the adversary. The environment expects the adversary to induce Alice to report output $f(x_b, y_c)$. The environment outputs 1 if the adversary succeeds at both phases (guessing b and inducing $f(x_b, y_c)$ when $b = 0$). These conditions are similar to those of a "split adversary" considered by [5]. In the ideal world, an adversary must choose an input for f with no knowledge of Alice's input, and it is easy to see that the adversary fails with probability at least $1/4$. On the other hand, a trivial adversary $\mathcal{A}$ attacking the canonical protocol for f can always succeed.

Applying Theorem 1 to $\mathcal{A}$ will result in a $\mathcal{S}$ that considers n levels of frontiers in $\widehat{\pi}$ – one for each step in the canonical protocol. $\mathcal{S}$ only needs to swap its input at most once (possibly from y_0 to y_1). By the choice of x_0 and x_1, $\mathcal{S}$ can make its decision to swap after visiting the first frontier. Let k be the last round in which Bob moves, then $k \in \{n-1, n\}$. By the choice of y_0 and y_1, $\mathcal{S}$ can safely swap anywhere except after visiting the $(k-1)th$ frontier. It suffices for the transcript to encounter an unmarked step in $\widehat{\pi}$ in this range. This is true with overwhelming probability, since there is an unmarked step in every $m \leq k-1$ consecutive steps in the protocol, and the frontiers are encountered in strict order with overwhelming probability, □

Let $g_n : \{0, 2, \ldots, 2n\} \times \{1, 3, \ldots, 2n+1\} \to \{0, 1, \ldots, 2n\}$ be defined as $g_n(x, y) = \min\{x, y\}$. It can be seen by inspection that g_n has a unique decomposition of depth $2n$. The corresponding canonical protocol is the one in which Alice first announces whether $x = 0$, then (if necessary) Bob announces whether $y = 1$, and so on — the "Dutch flower auction" protocols from [1].

Corollary 8. *The functions $g_1, g_2, \ldots$ form a strict, infinite hierarchy of increasing $\sqsubseteq$-complexity. That is, $g_i \sqsubseteq g_j$ if and only if $i \leq j$.*

Proof. By Theorem 7, $g_i \not\sqsubseteq g_j$ when $i > j$. It suffices to show that $g_n \sqsubseteq g_{n+1}$, since the $\sqsubseteq$ relation is transitive. It is straight-forward to see that the following is a UC-secure protocol for g_n in the g_{n+1}-hybrid world: both parties to send

their g_n-inputs directly to g_{n+1}, and abort if the response is out of bounds for the output of g_n (i.e., $2n$ or $2n+1$). The simulator for this protocol crucially uses the fact that the adversary can receive the output and then abort. □

Corollary 9. *There is no function f which is complete (with respect to the $\sqsubseteq$ relation) for the class of passively realizable SSFE functions, or for the class standalone-realizable SSFE functions.*

Proof. This follows from Theorem 7 and by observing that there exist functions of arbitrarily high decomposition depth in both classes in question. □

Corollary 10. *There exist SSFE functions f, g whose complexities are incomparable; that is, $f \not\sqsubseteq g$ and $g \not\sqsubseteq f$.*

Proof. If f is passively realizable but not standalone realizable, and g is standalone realizable, then $f \not\sqsubseteq g$, since the class of standalone security is closed under composition (at least when restricted to SSFE functions with abort, where the only kind of composition possible is sequential composition). On the other hand, if g has a unique decomposition depth at least 2 larger than the decomposition depth of f, then $g \not\sqsubseteq f$, by Theorem 7. □

One example of such a pair of functions is $f = \text{XOR}$ and $g = g_2$ from Corollary 8. In fact, using a more careful analysis, one can choose $g = g_1$ (see [19]). Thus XOR is incomparable to the entire $\{g_i\}$ hierarchy.

8 Conclusion and Open Problems

We have gained significant insight into the terrain of SSFE functions. However, there are regions of complexity of SSFE functions that we do not fully understand. In particular, we have not studied the class of incomplete functions which are not passive realizable. Nor have we attempted fully characterizing which functions are reducible to which ones. Going beyond SSFE functions, it remains open to explore similar questions for multi-party functionalities, for reactive functionalities, and (in the case of passive-security) for randomized functionalities. In the computationally bounded setting, the "zero-one conjecture" from [20] — that all functionalities are either realizable or complete — remains unresolved. It is also an intriguing problem to consider cryptographic complexity of multi-party functionalities vis a vis the "complexity" of cryptographic primitives (like one-way functions) that are required to realize them (in different hybrid worlds). In short, our understanding of cryptographic complexity of multi-party functionalities is still quite limited. There are exciting questions that probably call for a fresh set of tools and approaches.

References

1. Backes, M., Müller-Quade, J., Unruh, D.: On the necessity of rewinding in secure multiparty computation. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 157–173. Springer, Heidelberg (2007)

2. Beaver, D.: Perfect privacy for two-party protocols. In: Feigenbaum, J., Merritt, M. (eds.) Proceedings of DIMACS Workshop on Distributed Computing and Cryptography, vol. 2, pp. 65–77. American Mathematical Society (1989)
3. Beimel, A., Malkin, T., Micali, S.: The all-or-nothing nature of two-party secure computation. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 80–97. Springer, Heidelberg (1999)
4. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. Electronic Colloquium on Computational Complexity (ECCC) TR01-016 (2001); Extended abstract in FOCS 2001
5. Canetti, R., Kushilevitz, E., Lindell, Y.: On the limitations of universally composable two-party computation without set-up assumptions. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656. Springer, Heidelberg (2003)
6. Chor, B., Kushilevitz, E.: A zero-one law for boolean privacy (extended abstract). In: STOC, pp. 62–72. ACM, New York (1989)
7. Goldreich, O., Micali, S., Wigderson, A.: Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. J. ACM 38(3), 691–729 (1991); Preliminary version in FOCS 1986
8. Impagliazzo, R., Rudich, S.: Limits on the provable consequences of one-way permutations. In: STOC, pp. 44–61. ACM Press, New York (1989)
9. Ishai, Y., Prabhakaran, M., Sahai, A.: Founding cryptography on oblivious transfer - efficiently. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 572–591. Springer, Heidelberg (2008)
10. Kilian, J.: Founding cryptography on oblivious transfer. In: STOC, pp. 20–31. ACM, New York (1988)
11. Kilian, J.: Uses of Randomness in Algorithms and Protocols. PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology (1989)
12. Kilian, J.: A general completeness theorem for two-party games. In: STOC, pp. 553–560. ACM, New York (1991)
13. Kilian, J.: More general completeness theorems for secure two-party computation. In: Proc. 32nd STOC, pp. 316–324. ACM, New York (2000)
14. Kilian, J., Kushilevitz, E., Micali, S., Ostrovsky, R.: Reducibility and completeness in private computations. SIAM J. Comput. 29(4), 1189–1208 (2000)
15. Kraschewski, D., Müller-Quade, J.: Completeness theorems with constructive proofs for symmetric, asymmetric and general 2-party-functions (unpublished manuscript, 2008)
16. Künzler, R., Müller-Quade, J., Raub, D.: Secure computability of functions in the it setting with dishonest majority and applications to long-term security (in these proceedings)
17. Kushilevitz, E.: Privacy and communication complexity. In: FOCS, pp. 416–421. IEEE, Los Alamitos (1989)
18. Lindell, Y.: Lower bounds for concurrent self composition. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 203–222. Springer, Heidelberg (2004)
19. Maji, H., Prabhakaran, M., Rosulek, M.: Complexity of multiparty computation problems: The case of 2-party symmetric secure function evaluation. Cryptology ePrint Archive, Report 2008/454 (2008), http://eprint.iacr.org/
20. Prabhakaran, M., Rosulek, M.: Cryptographic complexity of multi-party computation problems: Classifications and separations. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 262–279. Springer, Heidelberg (2008)
21. Yao, A.C.: Protocols for secure computation. In: Proc. 23rd FOCS, pp. 160–164. IEEE, Los Alamitos (1982)