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

Hemanta K. Maji* Manoj Prabhakaran* Mike Rosulek*

November 3, 2010

**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 jointly compute a common value $f(x, y)$ in a secure way. Starting from Yao's millionaire's problem [20], 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.

**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

*Department of Computer Science, University of Illinois, Urbana-Champaign. {hmaji2,mmp,rosulek}@uiuc.edu. Partially supported by NSF grants CNS 07-47027 and CNS 07-16626.

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 [19] 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 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).

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.

[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.

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).

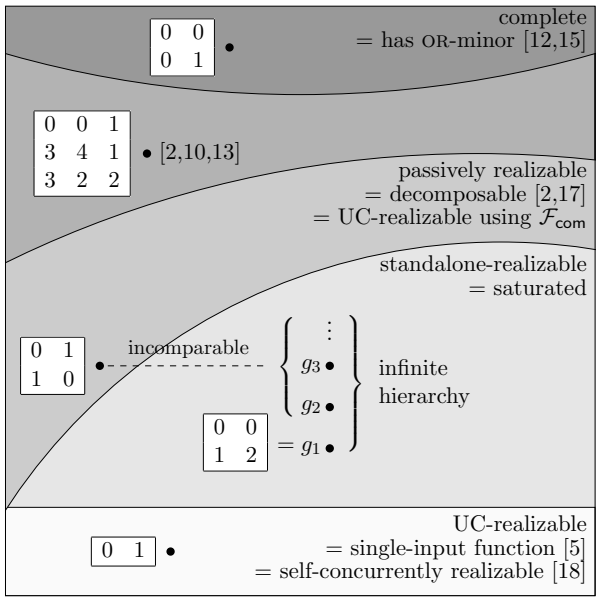

Figure 1: Cryptographic complexity of 2-party SSFE

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 [19].

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 $\pi$ 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" $\pi$, in the UC simulation sense. That is, for any adversary in the canonical-protocol-real-world, there is an adversary in the $\pi$-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 $\pi$ 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 $\epsilon$. Here $k$ is the *security parameter* that is 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; $\text{EXEC}_{\pi}$ 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 $\pi$ 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 $\pi$ 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.

### 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

[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.

- 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 (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.

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

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 $\begin{array}{|cc|}\hline 0 & 1 \\ 1 & 0 \\ \hline \end{array}$ is isomorphic to the function $\begin{array}{|cc|}\hline 1 & 2 \\ 3 & 4 \\ \hline \end{array}$.

**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, $\begin{array}{|cc|}\hline 0 & 0 \\ 1 & 2 \\ \hline \end{array}$ and $\begin{array}{|cc|}\hline 0 & 1 \\ 1 & 0 \\ \hline \end{array}$ are decomposable, but $\begin{array}{|cc|}\hline 0 & 0 \\ 0 & 1 \\ \hline \end{array}$ and $\begin{array}{|ccc|}\hline 0 & 0 & 1 \\ 3 & 4 & 1 \\ 3 & 2 & 2 \\ \hline \end{array}$ 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 $\begin{array}{|cc|}\hline 0 & 0 \\ 1 & 2 \\ \hline \end{array}$ is uniquely decomposable, but $\begin{array}{|cc|}\hline 0 & 1 \\ 1 & 0 \\ \hline \end{array}$ 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 [19] it is shown that for a class of functionalities called "deviation revealing", if a protocol $\pi$ 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 [19]:

**Lemma 1** ([19])**.** *Let $\pi$ be a UC-secure (perhaps in a hybrid world) or a standalone-secure protocol for an SSFE $f$. Then $\pi$ itself is a passive-secure protocol for $f$ as well (in the same hybrid world as $\pi$).*

*Proof.* We show that, without loss of generality, the simulator for $\pi$ maps passive real-world adversaries to passive ideal-world adversaries. A passive adversary $\mathcal{A}$ for $\pi$ is one which receives an input $x$ from the environment, runs $\pi$ 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 $\pi$ 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 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.

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

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 $\pi$ for $f$.

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

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

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 $\pi$ 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 $\nu$ such that, for all $y, y' \in Y$:*

- *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 ⊞-minor, then $SD\left(\mathcal{D}_F^{x,y}, \mathcal{D}_F^{x,y'}\right), SD\left(\mathcal{D}_F^{x',y}, \mathcal{D}_F^{x',y'}\right) \leq \nu(k)$.*

The full proof appears in Appendix A. Below we give a proof sketch.

*Sketch.* Suppose we have a protocol $\pi$ in normal form. Its next message function depends only on the appropriate party's input and the transcript so far. For $b \in \{0,1\}$ and $u \in \{0,1\}^*$, denote by $\pi_A(ub, x)$ is the probability that Alice's next message is $b$ when running on input $x$ and the transcript so far is $u$. Define $\pi_B$ for Bob analogously. Suppose we run $\pi$ on inputs $x, y$, where

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

Alice sends the first message The probability of obtaining a transcript with a particular prefix $u = u_1 \cdots u_n$ is (say $n$ is even):

$$\begin{aligned}\Pr[u|x,y] &= \pi_A(u_1,x)\cdot\pi_B(u_1u_2,y)\cdot\pi_A(u_1u_2u_3,x)\cdots\pi_B(u_1\cdots u_n,y)\\ &= \left(\prod_{i=1}^{n/2}\pi_A(u_1\cdots u_{2i-1},x)\right)\left(\prod_{i=1}^{n/2}\pi_B(u_1\cdots u_{2i},y)\right) = \alpha(u,x)\beta(u,y)\end{aligned}$$

where $\alpha(u,x)$ and $\beta(u,y)$ are defined to be the parenthesized quantities, respectively.

Given $x, x'$ as in the lemma statement, and for a fixed parameter $\mu < 1$, we define $F$ as the *prefix-minimal*[6] elements of:

$$\left\{u \;\middle|\; \left|\alpha(u,x) - \alpha(u,x')\right| \geq \mu[\alpha(u,x) + \alpha(u,x')]\right\}$$

We note that $F$ may not be a proper frontier — i.e., some protocol executions may never reach a point that satisfies the condition on $\alpha(u,\cdot)$. However, these "bad" protocol executions happen only with negligible probability, and thus can be safely ignored for the sake of this proof sketch.

Intuitively, $F$ is constructed to be the first point (i.e., frontier of transcript prefixes) at which Bob can "significantly" distinguish between Alice having $x$ versus $x'$, where significance is parameterized by $\mu$.

Since each individual transcript in $F$ significantly distinguishes between $x$ and $x'$, then clearly the overall transcript distribution over $F$ also significantly distinguishes between $x$ and $x'$. Thus both $\mathrm{SD}\left(\mathcal{D}_F^{x,y}, \mathcal{D}_F^{x',y}\right)$ and $\mathrm{SD}\left(\mathcal{D}_F^{x,y'}, \mathcal{D}_F^{x',y'}\right)$ are "significant" (i.e., proportional to $\mu$).

Next, since $f(x,y) = f(x,y')$, it must be that $\mathrm{SD}\left(\mathcal{D}_F^{x,y}, \mathcal{D}_F^{x,y'}\right)$ is close to 0 by the security of $\pi$. However, at the step in the protocol immediately before $F$ is reached, Bob cannot yet "significantly" distinguish between $x$ and $x'$. Thus (at least intuitively), replacing $x$ by $x'$ at this point elicits a non-significant change in the transcript distribution. Thus, $\mathrm{SD}\left(\mathcal{D}_F^{x',y}, \mathcal{D}_F^{x',y'}\right)$ is also close to 0.

The bounds obtained for both statistical distances depend on $\mu$, and by setting $\mu$ as an appropriate function of the protocol's statistical error, both quantities can be made simultaneously negligibly close to their desired values. □

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 $\pi$ for $f$, the canonical protocol for $f$ is "as secure as" $\pi$. That is, for every adversary attacking the canonical protocol, there is an adversary attacking $\pi$ which achieves the same effect in every environment.*

*Proof Sketch.* The complete proof (given in Appendix B) 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

[6] That is, the set obtained by repeatedly removing elements from the set which have a proper prefix in the set.

$x, x', y, y'$ 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 $\pi$. The simulator $\mathcal{S}$ simply keeps track of the current step in the decomposition of $f$, and runs $\pi$ 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 $\pi$ to any input consistent with the canonical protocol transcript so far, if necessary. Since the $\pi$-transcript so far is nearly independent of such distinctions among the adversary's input, it is indeed possible to swap inputs to $\pi$ 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 $\pi$ 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 $\pi$ for $f$, we have $\mathbb{E}[R(\pi, x, y)] \geq R(f, x, y) - \nu$, where $\nu$ is a negligible function in the security parameter of $\pi$.*

*Proof.* The proof of Theorem 1 constructs for $\pi$ 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 $\pi$ must visit all the relevant frontiers in order, one *strictly* after the next, with overwhelming probability. □

# 4 Characterizing Passive Security

In this section, we extend the characterization of Beaver [2] and Kushilevitz [**?**] 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.

($\Leftarrow$) The complete proof appears in Appendix C. Suppose $f$ is not a decomposable function. Without loss of generality, we may assume that $f$ is not even row- or column- decomposable at the top level. We will show that any *correct* protocol for $f$ can not also be a *secure* protocol for $f$. In particular, we will show that there exists two inputs $\sigma = (x, y)$ and $\sigma' = (x', y')$ such that $f(\sigma) = f(\sigma')$ and an adversary can distinguish the two inputs based on the generated transcript with significant advantage. This will break the security of the protocol.

We will follow a frontier based approach and look for the first time where Alice or Bob releases significant information about their input. Suppose at a node $v$ on the frontier Alice has released significant information about her input but Bob has not. We can partition Alice input set into two sets $A_v$ and $\bar{A}_v = X \setminus A_v$ such that all inputs in $A_v$ induce significantly different distribution as compared to her inputs in $\bar{A}_v$. More concretely, for any $x_1 \in A_v$ and $x_2 \in \bar{A}_v$, we have $\Pr[v|x_1, y]$ significantly lower than $\Pr[v|x_2, y]$, for any Bob input $y \in Y$. But we know that $f$ is not decomposable. So, there is $x_1^* \in A_v, x_2^* \in \bar{A}_v, y^* \in Y$ such that $f(x_1^*, y^*) = f(x_2^*, y^*)$. Thus at $v$, the security of the protocol for $f$ is compromised for input pairs: $(x_1^*, y^*)$ and $(x_2^*, y^*)$.

We show that in any correct protocol for $f$, Alice or Bob has to release information about their input with high probability. And for every transcript where Alice or Bob releases some information about their input, there exists a pair of inputs whose security is compromised by the protocol for $f$. When the partition $A_v$ and $\bar{A}_v$ is suitably defined, we can apply a pigeon-hole argument and show that the security for one such pair is compromised with significant probability in the protocol. □

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

Figure 3: Commitment functionality $\mathcal{F}_{\mathsf{com}}$

**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.*

*Proof Sketch.* ($\Leftarrow$) By Lemma 1, any UC-secure protocol $\pi$ 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 $\pi$ 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 $\pi$ 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 $\pi$. More details and an extension to the randomized case are given in Appendix D.

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 $\sigma$. 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 $\chi$ has exactly one 1) or open all $\sigma(i)$ (and verify that $\sigma$ is a permutation). Bob also commits to his own input in a similar way, with Alice giving challenges. Then both parties simulate the $\pi$ protocol one step at a time. When it is Alice's turn in $\pi$, she computes the appropriate next message $m$ 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 $Z = \{x' \mid \pi(t, x') \neq m\}$; i.e., the set of inputs for which the protocol instructs Alice to send a message *other than* $m$ at this point. Then Alice can open enough of her commitments to prove that $\chi_i = 0$ for all $i \in Z$ 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.*

*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 $\omega$ for $\mathcal{A}$, then for every Bob-strategy $B$, run the canonical protocol against $\mathcal{A}$ (using random tape $\omega$), 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 $f(x, y)$.

($\Rightarrow$) In Appendix E, we prove the following lemma by induction on the number of steps in the protocol:

**Lemma 3.** *If $\pi$ is a 2-party protocol whereby both parties agree on the output, then for any way of coloring the possible outputs red and blue, $\pi$ 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 $n > m + 1$.*[7] *Then there is no UC-secure protocol for $f$ in the $\mathcal{F}$-hybrid world; i.e., $f \not\sqsubseteq \mathcal{F}$.*

*Sketch.* We use a modified version of Theorem 1. Suppose for contradiction that $\pi$ is a protocol for $f$ in the $\mathcal{F}$-hybrid world. By Lemma 1, $\pi$ is also passive-secure in the same setting. Define $\widehat{\pi}$ to be the result of replacing each call to $\mathcal{F}$ in $\pi$ 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 $\pi$ adversary that achieves the same effect by interacting with $\mathcal{F}$ appropriately during these points.

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 $\pi$ in the $\mathcal{F}$-hybrid world, a contradiction.

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

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 $g = g_2$ defined in Corollary 8.

**Corollary 11.** *If $g$ is uniquely decomposable, then $g \not\sqsubseteq \text{XOR}$.*

*Proof.* The proof of Theorem 7 goes through with the following modifications: Suppose the adversary attacking $g$ plans to corrupt Alice. Then we will choose to replace the ideal XOR functionality with the canonical 2-round protocol in which Alice goes first. Thus *every* Alice turn in $\widehat{\pi}$ is a safe time for the adversary to swap its input, so the argument goes through without consideration of "behaving consistently" in certain places. □

Combined with Corollary 10, we have that 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 [19] — 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] M. Backes, J. Müller-Quade, and D. Unruh. On the necessity of rewinding in secure multiparty computation. In *TCC*, pages 157–173, 2007.

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

[3] 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.

[4] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. Electronic Colloquium on Computational Complexity (ECCC) TR01-016, 2001. Extended abstract in FOCS 2001.

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

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

[7] O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *J. ACM*, 38(3):691–729, July 1991. Preliminary version in FOCS' 86.

[8] R. Impagliazzo and S. Rudich. Limits on the provable consequences of one-way permutations. In *STOC*, pages 44–61. ACM, 1989.

[9] Y. Ishai, M. Prabhakaran, and A. Sahai. Founding cryptography on oblivious transfer - efficiently. In D. Wagner, editor, *CRYPTO*, volume 5157 of *Lecture Notes in Computer Science*, pages 572–591. Springer, 2008.

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

[11] J. Kilian. *Uses of Randomness in Algorithms and Protocols.* PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 1989.

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

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

[14] J. Kilian, E. Kushilevitz, S. Micali, and R. Ostrovsky. Reducibility and completeness in private computations. *SIAM J. Comput.*, 29(4):1189–1208, 2000.

[15] D. Kraschewski and J. Müller-Quade. Completeness theorems with constructive proofs for symmetric, asymmetric and general 2-party-functions. Unpublished Manuscript, 2008. `http://iks.ira.uka.de/eiss/completeness`.

[16] 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 these proceedings.

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

[18] Y. Lindell. Lower bounds for concurrent self composition. In M. Naor, editor, *TCC*, volume 2951 of *Lecture Notes in Computer Science*. Springer, 2004.

[19] M. Prabhakaran and M. Rosulek. Cryptographic complexity of multi-party computation problems: Classifications and separations. In *CRYPTO*, pages 262–279, 2008.

[20] A. C. Yao. Protocols for secure computation. In *Proc.* 23*rd FOCS*, pages 160–164. IEEE, 1982.

# A Proof of Lemma 2

To prove Lemma 2, we first prove the following technical lemma, which relates the statistical difference on a frontier to a function of the frontier's prefixes.

**Lemma 4.** *Let $F$ be a frontier of partial transcripts in which Alice has just spoken, and let $S$ be the set of proper prefixes of elements in $F$, also where Alice has just spoken.*

*Then there exist constants $c_\pi \geq 0$ and $\lambda(v, y, y') \geq 0$ for all $v \in S$ such that:*

$$\sum_{u \in F} \alpha(u, x)\big|\beta(u, y) - \beta(u, y')\big| = c_\pi + \sum_{v \in S} \alpha(v, x)\lambda(v, y, y')$$

*Proof.* For clarity, assume that each message in the protocol is a single bit, and that the parties strictly alternate rounds. The proof also holds for longer messages, but is more cumbersome.

Let $S_0$ be the prefix-minimal elements of $S$ (if Alice speaks first in the protocol, then $S_0 = \{0, 1\}$, else take $S_0 = \{\epsilon\}$). We will define a sequence $S_0 \subset S_1 \subset \cdots \subset S_t$ inductively as follows. Let $L(S_i)$ be the set of elements of $S_i$ that have no extensions in $S_i$; that is, $L(X) = \{x \in X \mid (\forall y)\ xy \notin X\}$. We define a *pop* operation on elements of $L(S_i)$ as follows. When we pop an element $u \in L(S_i)$, we expand the partial transcript $u$ by two moves in the protocol, setting $S_{i+1} = S_i \cup \{u00, u01, u10, u11\}$. After some finite number of operations we reach $S_t$ such that $L(S_t) = F$.

We define the *weight* of a tree $S_i$ as:

$$w(S_i) = \sum_{u \in L(S_i)} \alpha(u, x)\big|\beta(u, y) - \beta(u, y')\big|$$

Observe that $w(S_t) = \mathrm{SD}\Big(\mathcal{D}_F^{x,y}, \mathcal{D}_F^{x,y'}\Big)$. We define the quantity $c_\pi = w(S_0) \geq 0$. We define the intermediate quantity $\delta(u, y, y') = |\beta(u, y) - \beta(u, y')|$. Suppose the node $v$ was popped in $S_i$ to obtain $S_{i+1}$. Then,

$$\begin{aligned} w(S_{i+1}) - w(S_i) = {} & \alpha(v00, x)\delta(v00, y, y') + \alpha(v01, x)\delta(v01, y, y') \\ & + \alpha(v10, x)\delta(v10, y, y') + \alpha(v11, x)\delta(v11, y, y') - \alpha(v, x)\delta(v, y, y') \end{aligned}$$

Since $v0$ and $v1$ are nodes where Bob has just spoken, we get that $\delta(v00, y, y') = \delta(v01, y, y') = \delta(v0, y, y')$ and $\delta(v10, y, y') = \delta(v11, y, y') = \delta(v1, y, y')$. Similarly, $\alpha(v0, x) = \alpha(v1, x) = \alpha(v, x)$. By definition, we have $\alpha(v00, x) + \alpha(v01, x) = \alpha(v0, x)$ and $\alpha(v10, x) + \alpha(v11, x) = \alpha(v1, x)$. Using these observation we get:

$$\begin{aligned} w(S_{i+1}) - w(S_i) &= \left(\alpha(v00, x) + \alpha(v01, x)\right)\delta(v0, y, y') \\ &\quad + \left(\alpha(v10, x) + \alpha(v11, x)\right)\delta(v1, y, y') - \alpha(v, x)\delta(v, y, y') \\ &= \alpha(v0, x)\delta(v0, y, y') + \alpha(v1, x)\delta(v1, y, y') - \alpha(v, x)\delta(v, y, y') \\ &= \alpha(v, x)\left[\delta(v0, y, y') + \delta(v1, y, y') - \delta(v, y, y')\right] \end{aligned}$$

Define $\lambda(v, y, y') = \delta(v0, y, y') + \delta(v1, y, y') - \delta(v, y, y')$ and it is easy to see that $\lambda(v, y, y') \geq 0$. The expression for $w(S_t)$ telescopes, and we obtain the desired result:

$$w(S_t) = c_\pi + \sum_{v \in S} \alpha(x, v)\lambda(v, y, y') \qquad \square$$

*Proof of Lemma 2.* Suppose we have a protocol $\pi$ in normal form, and let $\epsilon$ denote the simulation error of $\pi$ (i.e., the statistical difference between real- and ideal-world outputs). The next message function depends only on the appropriate party's input and the transcript so far. For $b \in \{0,1\}$ and $u \in \{0,1\}^*$, denote by $\pi_A(ub, x)$ is the probability that Alice's next message is $b$ when running on input $x$ and the transcript so far is $u$. Define $\pi_B$ for Bob analogously. Suppose we run $\pi$ on inputs $x, y$, where Alice sends the first message The probability of obtaining a particular transcript prefix $u = u_1 \cdots u_n$ is (say $n$ is even):

$$\begin{aligned}\Pr[u|x,y] &= \pi_A(u_1,x)\cdot\pi_B(u_1u_2,y)\cdot\pi_A(u_1u_2u_3,x)\cdots\pi_B(u_1\cdots u_n,y)\\ &= \left(\prod_{i=1}^{n/2}\pi_A(u_1\cdots u_{2i-1},x)\right)\left(\prod_{i=1}^{n/2}\pi_B(u_1\cdots u_{2i},y)\right) = \alpha(u,x)\beta(u,y)\end{aligned}$$

where $\alpha(u,x)$ and $\beta(u,y)$ are defined to be the parenthesized quantities, respectively.

Given $x, x'$ as above, and for a fixed $\mu < 1$, to be defined later, we define the frontier $F$ as the *prefix-minimal* elements of:

$$\{u \mid u \text{ is a complete transcript, or } \left|\alpha(u,x) - \alpha(u,x')\right| \geq \mu(\alpha(u,x) + \alpha(u,x'))\}$$

By "complete transcript", we mean one on which the parties terminate and give output.

We partition $F$ into $F_{\text{good}} \cup F_{\text{bad}}$, where $F_{\text{good}}$ are the elements $u \in F$ which satisfy $|\alpha(u,x) - \alpha(u,x')| \geq \mu(\alpha(u,x)+\alpha(u,x'))$. Each element $u \in F_{\text{bad}}$ therefore satisfies $\alpha(u,x)/\alpha(u,x') \leq (1+\mu)/(1-\mu)$.

Consider any $y$ such that $f(x,y) \neq f(x',y)$. Let $A \subseteq F_{\text{bad}}$ be the transcripts in $F_{\text{bad}}$ which do *not* induce output $f(x,y)$ (all elements of $F_{\text{bad}}$ are complete transcripts, and the output is a function of the transcript alone). Similarly let $B \subseteq F_{\text{bad}}$ be the transcripts in $F_{\text{bad}}$ which do *not* induce output $f(x',y)$. Thus $\Pr[A|x,y]$ and $\Pr[B|x',y]$ must be at most $\epsilon$. Observe that $F_{\text{bad}} \subseteq A \cup B$. Thus we have:

$$\begin{aligned}\Pr[F_{\text{good}}|x,y] &= 1 - \Pr[F_{\text{bad}}|x,y] \geq 1 - \sum_{u\in A}\alpha(u,x)\beta(u,y) - \sum_{u\in B}\alpha(u,x)\beta(u,y)\\ &\geq 1 - \left(\frac{1+\mu}{1-\mu}\right)\sum_{u\in A}\alpha(u,x)\beta(u,y) - \left(\frac{1+\mu}{1-\mu}\right)\sum_{u\in B}\alpha(u,x')\beta(u,y)\\ &= 1 - \left(\frac{1+\mu}{1-\mu}\right)(\Pr[A|x,y] + \Pr[B|x',y]) \geq 1 - 2\left(\frac{1+\mu}{1-\mu}\right)\epsilon\end{aligned}$$

The same bound holds for $\Pr[F_{\text{good}}|x',y]$. Now,

$$\begin{aligned}\text{SD}\left(\mathcal{D}_F^{x,y}, \mathcal{D}_F^{x',y}\right) &= \frac{1}{2}\sum_{u\in F}\left|\alpha(u,x) - \alpha(u,x')\right|\beta(u,y) \geq \frac{1}{2}\sum_{u\in F_{\text{good}}}\mu\left(\alpha(u,x) + \alpha(u,x')\right)\beta(u,y)\\ &= \frac{\mu}{2}\Big(\Pr[F_{\text{good}}|x,y] + \Pr[F_{\text{good}}|x',y]\Big) \geq \mu\left(1 - 2\left(\frac{1+\mu}{1-\mu}\right)\epsilon\right)\end{aligned}$$

Now consider any $y$ and $y'$ such that $x, x', y, y'$ form a $\boxplus$-minor. Since $f(x,y) = f(x,y')$, we must have $\text{SD}\left(\mathcal{D}_F^{x,y}, \mathcal{D}_F^{x,y'}\right) \leq \epsilon$ by the security of the protocol. Let $S$ be the set of all proper prefixes of elements $F$ that correspond to points where Alice has just spoken. By Lemma 4, we can express

the statistical difference at $F$ in terms of a linear combination of $\alpha(v,x)$ values for $v \in S$. Since by definition every $v \in S$ satisfies $\alpha(v,x')/\alpha(v,x) \leq (1+\mu)/(1-\mu)$, we have:

$$\begin{aligned}
\mathrm{SD}\Big(\mathcal{D}_F^{x',y}, \mathcal{D}_F^{x',y'}\Big) &= \frac{1}{2}\sum_{u\in F} \alpha(u,x')\big|\beta(u,y) - \beta(u,y')\big| = \frac{1}{2}\left(c_\pi + \sum_{v\in S}\alpha(v,x')\lambda(v,y,y')\right) \\
&\leq \frac{1}{2}\left(c_\pi + \frac{1+\mu}{1-\mu}\sum_{v\in S}\alpha(v,x)\lambda(v,y,y')\right) \leq \frac{1}{2}\left(\frac{1+\mu}{1-\mu}\right)\left(c_\pi + \sum_{v\in S}\alpha(v,x)\lambda(v,y,y')\right) \\
&= \left(\frac{1+\mu}{1-\mu}\right)\mathrm{SD}\Big(\mathcal{D}_F^{x,y}, \mathcal{D}_F^{x,y'}\Big) \leq \left(\frac{1+\mu}{1-\mu}\right)\epsilon
\end{aligned}$$

Choosing $\mu = 1-\sqrt{\epsilon}$ and defining $\nu = 5\sqrt{\epsilon}$, we get the desired bounds. □

# B Proof of Theorem 1

We now prove our main technical result, Theorem 1. First, we establish a convenient lemma:

**Lemma 5.** *For frontiers $F$ and $G$, let "$[G \prec F|x,y]$" denote the event that when running the protocol on inputs $x, y$, the transcript reaches $G$ strictly before reaching $F$. Then*

$$SD\Big(\mathcal{D}_F^{x,y}, \mathcal{D}_F^{x',y'}\Big) \leq SD\Big(\mathcal{D}_G^{x,y}, \mathcal{D}_G^{x',y'}\Big) + \frac{1}{2}(\Pr[G \prec F|x,y] + \Pr[G \prec F|x',y'])$$

*Proof.* Partition $F$ into $F_1$ and $F_2$, where $F_1$ are the strings in $F$ which are prefixes of strings in $G$. Thus the distribution over the frontier $F_1$ is a function of the distribution over the frontier $G$. Then

$$\begin{aligned}
\mathrm{SD}\Big(\mathcal{D}_F^{x,y}, \mathcal{D}_F^{x',y'}\Big) &= \frac{1}{2}\sum_{u\in F_1} |\alpha(u,x)\beta(u,y) - \alpha(u,x')\beta(u,y')| + \frac{1}{2}\sum_{u\in F_2} |\alpha(u,x)\beta(u,y) - \alpha(u,x')\beta(u,y')| \\
&\leq \mathrm{SD}\Big(\mathcal{D}_G^{x,y}, \mathcal{D}_G^{x',y'}\Big) + \frac{1}{2}\sum_{u\in F_2}\big(\alpha(u,x)\beta(u,y) + \alpha(u,x')\beta(u,y')\big) \\
&= \mathrm{SD}\Big(\mathcal{D}_G^{x,y}, \mathcal{D}_G^{x',y'}\Big) + \frac{1}{2}\big(\Pr[F_2|x,y] + \Pr[F_2|x',y']\big) \\
&= \mathrm{SD}\Big(\mathcal{D}_G^{x,y}, \mathcal{D}_G^{x',y'}\Big) + \frac{1}{2}(\Pr[G \prec F|x,y] + \Pr[G \prec F|x',y'])
\end{aligned}$$

□

We can now prove Theorem 1. For simplicity, we restrict attention to the case where the decomposition of $f$ is binary (that is, each row decomposition is $X = X_0 \cup X_1$ and each column decomposition is $Y = Y_0 \cup Y_1$), as that is all that is needed for our impossibility results, and the proof is much simpler.

As before, let $\epsilon$ be the statistical error of the protocol $\pi$, let $\nu$ be the negligible error guaranteed by Lemma 2, and define the quantity $\mu_i$ according to the recurrence $\mu_0 = 2\epsilon$ and $\mu_{i+1} = 11\nu + 16\mu_i$. Note that for fixed $i$, $\mu_i$ is negligible whenever $\epsilon$ and $\nu$ are negligible.

The bulk of the proof is in the following inductive lemma. For each step $X \times Y$ in the decomposition of $f$, we will construct a frontier $F(X,Y)$ with the following property:

**Lemma 6.** *Let $d$ be the depth (number of decomposition steps) of the decomposition of $f\big|_{X\times Y}$. For all $x, x' \in X$ and $y, y' \in Y$, if $(x,y)$ and $(x',y')$ are in different parts of the* next *decomposition step of $f\big|_{X\times Y}$, then $SD\Big(\mathcal{D}^{x,y}_{F(X,Y)}, \mathcal{D}^{x',y'}_{F(X,Y)}\Big) \geq 1-\mu_d$; otherwise, $SD\Big(\mathcal{D}^{x,y}_{F(X,Y)}, \mathcal{D}^{x',y'}_{F(X,Y)}\Big) \leq \mu_d$.*

*Proof.* We prove the claim inductively according to the decomposition of $f$. For $X \times Y$ such that $f$ is constant on $X \times Y$, we define $F(X,Y)$ as the set of *complete* transcripts (i.e., those on which the parties terminate and give output). By the security of the protocol, we have that $\mathrm{SD}\Big(\mathcal{D}^{x,y}_{F(X,Y)}, \mathcal{D}^{x',y'}_{F(X,Y)}\Big) \leq 2\epsilon = \mu_0$ for all $x, x' \in X$ and $y, y' \in Y$.

Otherwise suppose that $X \times Y$ is further decomposed as $X = X_0 \cup X_1$ (the case for a column decomposition is symmetric). Our first step is to pick representatives $x_0 \in X_0$ and $x_1 \in X_1$ so that there are certain $\boxplus$-minors involving $x_0$ and $x_1$, and we will be able to apply Lemma 2 directly. We pick the representatives as follows:

If $f$ is constant on $X_1 \times Y$, then we pick the representative $x_0 \in X_0$ arbitrarily. Otherwise, suppose $X_1 \times Y$ is decomposed as $Y = Y_0 \cup Y_1$. This column decomposition $Y_0 \cup Y_1$ is invalid for $X_0 \times Y$, since if it were valid, the decomposition of $f$ would not be unique. Therefore there must be an $x_0 \in X_0$, $y \in Y_0$, $y' \in Y_1$ such that $f(x_0, y) = f(x_0, y')$. We use this $x_0$ as the representative for $X_0$. In this way, we have ensured that for any $x_1 \in X_1$, these values $x_0, x_1, y, y'$ form a $\boxplus$-minor, which (intuitively) witnesses the fact that the $X = X_0 \cup X_1$ decomposition must happen before the $Y = Y_0 \cup Y_1$ decomposition. The representative for $X_1$ is chosen analogously.

Given these representatives $x_0, x_1$, we define $F(X,Y)$ to be the frontier given by Lemma 2 for $x_0, x_1$. We refer to $F(X,Y)$ as $F$ for short, and $F(X_0, Y)$ as $G$.

We wish to show that for all $x, x' \in X_0$ and $y, y' \in Y$, $\mathrm{SD}\Big(\mathcal{D}^{x,y}_F, \mathcal{D}^{x',y'}_F\Big) \leq \mu_d$ (the analogous claim for $X_1$ is symmetric). If $f$ is constant on $X_0 \times Y$, then the statistical difference is in fact at most $2\epsilon$ by the security of the protocol, and we are done. Otherwise, if $X_0 \times Y$ is decomposed as $Y = Y_0 \cup Y_1$, there must be $y_0 \in Y_0$, $y_1 \in Y_1$ such that $x_0, x_1, y_0, y_1$ are a $\boxplus$ minor, by our choice of $x_0, x_1$. Thus by Lemma 2, we have that $\mathrm{SD}\big(\mathcal{D}^{x_0,y_0}_F, \mathcal{D}^{x_0,y_1}_F\big) \leq \nu$. However, inductively we also have that $\mathrm{SD}\big(\mathcal{D}^{x_0,y_0}_G, \mathcal{D}^{x_0,y_1}_G\big) \geq 1 - \mu_{d-1}$. So by Lemma 5, $\Pr[F \prec G | x_0, y_0] \geq 1 - 2(\nu + \mu_{d-1})$.

Now, take any $x \in X_0$, $y \in Y_0$. We have inductively that $\mathrm{SD}\big(\mathcal{D}^{x,y}_G, \mathcal{D}^{x_0,y_0}_G\big) \leq \mu_{d-1}$. Note that the condition "$F \prec G$" for a transcript can be determined by seeing only the transcript up to the point $G$. This suggests a statistical test for distributions on $G$. Thus we must have

$$\Big|\Pr[F \prec G | x, y] - \Pr[F \prec G | x_0, y_0]\Big| \leq \mathrm{SD}\big(\mathcal{D}^{x,y}_G, \mathcal{D}^{x_0,y_0}_G\big) \leq \mu_{d-1}.$$

Combining what we know, this implies that $\Pr[F \prec G | x, y] \geq 1 - (2\nu + 3\mu_{d-1})$. An analogous statement also holds for with $Y_1$ in place of $Y_0$.

Now take any $x \in X_0$, $y \in Y_0$. By Lemma 5, and the inductive hypothesis,

$$\mathrm{SD}\big(\mathcal{D}^{x,y}_F, \mathcal{D}^{x_0,y_0}_F\big) \leq \mathrm{SD}\big(\mathcal{D}^{x,y}_G, \mathcal{D}^{x_0,y_0}_G\big) + \frac{1}{2}\Big(\Pr[G \prec F | x_0, y_0] + \Pr[G \prec F | x, y]\Big) \leq 2\nu + 4\mu_{d-1}$$

The same bound holds with $y_1$ in place of $y_0$ and $Y_1$ in place of $Y_0$.

Finally, combining everything, for any $x, x' \in X_0$, $y \in Y_0$, and $y' \in Y_1$, we have by the triangle inequality:

$$\begin{aligned}\mathrm{SD}\Big(\mathcal{D}^{x,y}_F, \mathcal{D}^{x',y'}_F\Big) &\leq \mathrm{SD}\big(\mathcal{D}^{x,y}_F, \mathcal{D}^{x_0,y_0}_F\big) + \mathrm{SD}\big(\mathcal{D}^{x_0,y_0}_F, \mathcal{D}^{x_0,y_1}_F\big) + \mathrm{SD}\Big(\mathcal{D}^{x_0,y_1}_F, \mathcal{D}^{x',y'}_F\Big) \\ &\leq (2\nu + 4\mu_{d-1}) + \nu + (2\nu + 4\mu_{d-1}) = 5\nu + 8\mu_{d-1} \leq \mu_d\end{aligned}$$

The case where $y, y' \in Y_0$ or $y, y' \in Y_1$ is shown in a similar way. This proves the first part of the inductive claim.

For the other part of the claim, since $X = X_0 \cup X_1$ is a valid row-decomposition of $X \times Y$, we know that $f(x_0, y) \neq f(x_1, y)$ for every $y \in Y$. Thus $\mathrm{SD}\big(\mathcal{D}_F^{x_0,y}, \mathcal{D}_F^{x_1,y}\big) \geq 1 - \nu$ by Lemma 2 and the definition of $F$. By a similar triangle inequality argument, we have that for all $x \in X_0$, $x' \in X_1$, $y, y' \in Y$:

$$\begin{aligned}\mathrm{SD}\Big(\mathcal{D}_F^{x,y}, \mathcal{D}_F^{x',y'}\Big) &\geq -\mathrm{SD}\big(\mathcal{D}_F^{x,y}, \mathcal{D}_F^{x_0,y}\big) + \mathrm{SD}\big(\mathcal{D}_F^{x_0,y}, \mathcal{D}_F^{x_1,y}\big) - \mathrm{SD}\Big(\mathcal{D}_F^{x_1,y}, \mathcal{D}_F^{x',y'}\Big) \\ &\geq 1 - \nu - 2(5\nu + 8\mu_{d-1}) \geq 1 - \mu_{d-1}.\end{aligned}$$

□

Given the previous lemma, we prove Theorem 1:

*Proof of Theorem 1.* Given such frontiers corresponding to each decomposition step, the required simulation is natural. The simulator's job is to simulate the canonical protocol to $\mathcal{A}$, while interacting with the honest party in the protocol $\pi$. The simulator $\mathcal{S}$ simply keeps track of a subdomain in the decomposition of $f$, and runs $\pi$ honestly with any representative input. Whenever it reaches the next frontier for a decomposition step by the honest party, it does the following: By the definition of the frontier it can determine (with only negligible error) which part of the decomposition the honest party's input belongs to, and simulate the next step in the canonical protocol to the adversary. Then the simulator receives the adversary's next move in the canonical protocol and changes its own input for $\pi$ accordingly, if necessary. After the last message of the canonical protocol, $\mathcal{S}$ continues honestly running $\pi$ until it terminates.

The simulation is perfect except for the following possible sources of error: (1) $\mathcal{S}$ may give an incorrect guess for the honest party's next canonical-protocol message; (2) when honestly executing $\pi$ with the honest party, $\mathcal{S}$ often changes its input; and (3) $\mathcal{S}$ may reach the frontiers in an unexpected order. Events 1 and 3 both happen with negligible probability by the constructions of the frontiers. Furthermore, conditioned on the frontiers being encountered in the expected order, $\mathcal{S}$ only changes its input only at points where the transcript distribution is nearly independent of its input. In other words, a transcript in which $\mathcal{S}$ changes its input, say from $x$ to $x'$, is indistinguishable from a transcript in which $\mathcal{S}$ had used $x'$ as its input the whole time. Thus with overwhelming probability, the transcript seen by the honest party is indistinguishable from one in which $\mathcal{S}$ executes $\pi$ entirely honestly with an input that is consistent with the entire canonical-protocol transcript, so the effect given by $\mathcal{S}$ (against $\pi$) is the same as that of $\mathcal{A}$ (against the canonical protocol). □

# C Proof of Theorem 3

*Proof of Theorem 3.* ($\Leftarrow$) Suppose $f$ is an not decomposable. Without loss of generality, let us assume that the function $f$ is not decomposable at the top level itself, i.e.

1. For any partition $A$ and $\bar{A}$ of Alice input set $X$, there exists Alice inputs $x \in A$ and $x' \in \bar{A}$; and Bob input $y^*$ such that $f(x, y^*) = f(x', y^*)$, and
2. For any partition $A$ and $\bar{A}$ of Bob input set $Y$, there exists Bob inputs $y \in A$ and $y' \in \bar{A}$; and Alice input $x^*$ such that $f(x^*, y) = f(x^*, y')$.

Let the set of Alice and Bob input sets $X$ and $Y$ be $m$ and $n$ respectively. For any $(1 - \nu)$ correct protocol for $f$, we will show that there exists a pair of inputs $\sigma_1 = (x, y)$ and $\sigma_2 = (x', y')$ such that:

1. $x = x'$ or $y = y'$,
2. $f(\sigma_1) = f(\sigma_2)$, and
3. $\mathrm{SD}\big(\mathcal{D}_L^{\sigma_1}, \mathcal{D}_L^{\sigma_2}\big) \geq \left(1 - \frac{1}{2^{1/(m+n)}}\right) \frac{1-4mn\nu}{4mn(m+n)}$, where $L$ is the set of leaves in the transcript tree.

We know that the probability $\Pr[v|x,y]$ can be equivalently written as $\alpha(v,x)\beta(v,y)$. Starting from the root of the transcript tree, expand a node $v$ if:

1. For all Alice inputs $x, x'$, $\alpha(v,x) \leq 2\alpha(v,x')$, and
2. For all Bob inputs $y, y'$, $\beta(v,y) \leq 2\beta(v,y')$.

The set of nodes which are not expanded form a frontier $F$ in the transcript tree. Let $B \subseteq F$ be the set of leaves where we kept on expanding and finally reached the leaves. So, for any pairs of inputs $\sigma, \sigma'$ and $\tau \in B$, we have $\Pr[\tau|\sigma] \leq 4\Pr[\tau|\sigma']$. If $\Pr[\tau|\sigma] > 0$ for some input $\sigma \in X \times Y$ then $\Pr[\tau|\tilde{\sigma}] > 0$ for all inputs $\tilde{\sigma} \in X \times Y$. Since, given a transcript $\tau$ the output corresponding to it is fixed and $f$ is not a constant function, there exists $\sigma_\tau$ such that $\tau$ gives wrong answer for the input $\sigma_\tau$. Now, we can show that the probability of generating any transcript in $B$ given *any* input $\sigma$ is significantly small due to the correctness of the protocol:

$$\begin{aligned}\sum_{\tau \in B} \Pr[\tau|\sigma] &\leq 4 \sum_{\tau \in B} \Pr[\tau|\sigma_\tau] \\ &\leq 4 \sum_{\sigma' \in X \times Y} \sum_{\tau \in W_{\sigma'}} \Pr[\tau|\sigma'] \leq 4mn\nu\end{aligned}$$

, where $W_{\sigma'}$ is the set of transcripts which provide wrong answer for the input $\sigma'$.

Any node $v \in F \setminus B$ either satisfies:

1. There exists Alice inputs $x, x'$ such that $\alpha(v,x) > 2\alpha(v,x')$ and for all Bob inputs $y, y'$ we have $\beta(v,y) \leq 2\beta(v,y')$, or
2. There exists Bob inputs $y, y'$ such that $\beta(v,y) > 2\beta(v,y')$ and for all Alice inputs $x, x'$ we have $\alpha(v,x) \leq 2\alpha(v.x')$.

Consider the case when $v$ satisfies the first condition. Define $S_i$ as the set of Alice inputs $\tilde{x}$ such that $\alpha(v,\tilde{x}) \leq 2^{i/m}\alpha(v,x')$. Let $m'$ be the minimum $i$ such that $S_i$ is the set of all Alice inputs $X$. Since $\alpha(v,x) > 2\alpha(v,x')$, we have $m' > m$. There are at most $m$ distinct sets possible. So, we can define $i^* < m'$ as the largest $i$ such that $S_i = S_{i+1}$. We partition Alice inputs into two sets $S = S_{i^*}$ and $\bar{S} = X \setminus S_{i^*}$. This partition satisfies the following two properties:

1. For any $x_1 \in \bar{S}$ and $x_2 \in S$ we have $\alpha(v,x_1) > 2^{1/m}\alpha(v,x_2)$. This follows from the fact that $\alpha(v,x_2) \leq 2^{i^*/m}\alpha(v,x')$ and $\alpha(v,x_1) > 2^{(i^*+1)/m}\alpha(v,x')$. Intuitively, it implies that all Alice inputs in one set look *significantly* different from all the inputs in the other set.
2. For any $x_1 \in \bar{S}$ and any Bob input $y$, we have $\Pr[v|x_1,y] \geq \frac{1}{4}\max_\sigma \Pr[v|\sigma]$. This follows from the fact that $i^*$ is the largest $i$ such that $S_i = S_{i+1}$. So, $2\alpha(v,x_1) \geq \alpha(v,\tilde{x})$ for any Alice input $\tilde{x}$. And we know that $2\beta(v,y) \geq \beta(v,\tilde{y})$ for any Bob input $\tilde{y}$. Intuitively, it implies that for all $\tilde{\sigma} \in S \times Y$, $\Pr[v|\tilde{\sigma}]$ is quite *high*.

Since $f$ is not decomposable, there exists $x_1^* \in \bar{S}$ and $x_2^* \in S$ and a Bob input $y^*$ such that:

1. $f(x_1^*, y^*) = f(x_2^*, y^*)$,
2. $\Pr[v|x_1^*, y^*] > 2^{1/m} \Pr[v|x_2^*, y^*] \geq 2^{1/(m+n)} \Pr[v|x_2^*, y^*]$, and
3. $\Pr[v|x_1^*, y^*] > \frac{1}{4} \Pr[v|\sigma]$ for any input $\sigma \in X \times Y$.

The last two properties are consequences of the definition of $\bar{S}$ and $S$.

We say that the pair of inputs $\sigma_1^* = (x_1^*, y^*)$ and $\sigma_2^* = (x_2^*, y^*)$ *witness* the distinction at partial transcript $v$. Similarly, we can also define "witness" when $v$ satisfies the case: "There exists Bob inputs $y, y'$ such that $\beta(v, y) > 2\beta(v, y')$ and for all Alice inputs $x, x'$ we have $\alpha(v, x) \leq 2\alpha(v, x')$."

Observe that there are at most $mn(m+n)$ distinct witnesses. Pick any input $\sigma \in X \times Y$ and consider the distribution of witnesses when we generate $v \in F \setminus B$ with probability $\Pr[v|\sigma]$. There exists $L \subseteq F \setminus B$ and witness $\sigma_1$ and $\sigma_2$ such that:

1. $\sum_{v \in L} \Pr[v|\sigma_1] \geq \frac{1}{4} \sum_{v \in L} \Pr[v|\sigma] \geq \frac{1-4mn\nu}{4mn(m+n)}$, and
2. $\Pr[v|\sigma_1] > 2^{1/(m+n)} \Pr[v|\sigma_2]$

Now, it is immediate that for this choice of $\sigma_1$ and $\sigma_2$ we have $\mathrm{SD}\big(\mathcal{D}_L^{\sigma_1}, \mathcal{D}_L^{\sigma_2}\big) \geq \left(1 - \frac{1}{2^{1/(m+n)}}\right) \frac{1-4mn\nu}{4mn(m+n)}$. This implies that any $(1-\nu)$ correct protocol for $f$ (which is not decomposable) can not be a secure protocol, when $m$ and $n$ are constants. □

# D Proof of Theorem 4

The complete protocol is given in Figure 4, and we prove its security below in Lemma 7.

**Lemma 7.** *If $\pi$ is a passive-secure protocol for $f$, then the compiled version of $\pi$ (Figure 4) is a UC-secure protocol in the $\mathcal{F}_{\mathsf{com}}$-hybrid world.*

*Proof.* We describe the simulator for a corrupt Alice (the other case is symmetric). In the setup phase, the simulator simulates $\mathcal{F}_{\mathsf{com}}$ to immediately extract the $\sigma[i, j]$ and $\chi[i, j]$ commitments. It then picks random $b_i$ challenges. If any of the $b_i$ will result in the adversary being "caught" in the setup phase, then the simulator does not contact the ideal functionality for $f$. The simulator gives random consistent values for party 2's commitments in the setup phase, gives the $b_i$ challenges, and then aborts after the adversary opens its commitments.

Otherwise, if the $b_i$ challenges would not cause an abort, the simulator will contact the ideal functionality. We say that $\sigma[i, \cdot]$ is valid if it is a permutation of $\{1, \ldots, n_x\}$. We say that $\chi[i, \cdot]$ is valid if all but one of the values is zero. If for all $i$, either $\sigma[i, \cdot]$ or $\chi[i, \cdot]$ are invalid (and yet the $b_i$ challenges did not catch the adversary — an event which can happen only with probability $2^{-k}$), the simulator aborts. Otherwise consider an arbitrary $i$ such that both $\sigma[i, \cdot]$ and $\chi[i, \cdot]$ are valid. Let $x$ be the unique value such that $\chi[i, \sigma[i, x]] = 1$. The simulator will send $x$ to the ideal functionality on behalf of the adversary, and receive $f(x, y)$. It is finally the simulator's task to decide whether to deliver the output to honest Bob.

Given $x$ and $f(x, y)$, choose any $y^*$ such that $f(x, y) = f(x, y^*)$. The simulator then proceeds to simulate Bob honestly as if it had input $y^*$. The simulator can always allow Bob to open his commitments consistently. If at any point the simulated Bob aborts, the simulation ends without

Our protocol is essentially symmetric with respect to the two parties. We describe only the behavior of Alice. We denote the parties' input ranges as $X = \{1, \ldots, n_x\}$ and $Y = \{1, \ldots, n_y\}$, respectively. The protocol's security parameter is $k$.

Setup phase:

1. Choose random permutations $\sigma_1, \ldots, \sigma_k$ of $X$. Let $\chi$ be the characteristic vector of $x$ in $X$. Commit to the values $\chi[i, j] = \chi_{\sigma_i^{-1}(j)}$ and $\sigma[i, j] = \sigma_i(j)$ for $i \leq k$ and $j \leq n_x$.

2. Wait for Bob to similarly commit to his own $\chi'[i, j]$ and $\sigma'[i, j]$ values.

3. Receive $b_i \in \{0, 1\}$ for $i \leq k$. For each $i$, if $b_i = 0$, then open each $\sigma[i, j]$ commitment; otherwise, open each $\chi[i, j]$ commitment.

4. Similarly, choose random $b_i \leftarrow \{0, 1\}$ for all $i \leq k$ and send them to Bob. For each $i$ such that $b_i = 0$, expect $\sigma'[i, j]$ to be opened and verify that they constitute a permutation of $\{1, \ldots, n_y\}$, else abort. For each $i$ such that $b_i = 1$, expect each $\chi'[i.j]$ to be opened and verify that all but one of them are equal to 0, else abort.

Then, for each step of $\pi$:

1. If this is a step where Alice sends a message, then let $m$ be the message prescribed by $\pi$ protocol at this point. Let $Z$ be the set of inputs which would have prescribed sending a message other than $m$ at this point in the protocol. Send $m$ to Bob, and and for each $i \leq k$ and $j \in Z$, open the commitments of $\sigma[i, j]$ and $\chi[i, \sigma_i(j)]$ $(= \chi_j)$. (Half of these commitments will already be opened).

2. If this is a step where Bob sends a message, then expect a next message $m$ for the $\pi$ protocol simulation. Let $Z$ be the set of inputs for Bob which would have prescribed that Bob send message other than $m$ at this point in the protocol. from party 2. For each $i \leq k$ and $j \in Z$, expect $\sigma'[i, j]$ to be opened (if not already). If any $\sigma'[i, j] \notin \{1, \ldots, n_y\}$, or if $\sigma'[i, j] = \sigma'[i, j']$ for any $j \neq j'$, then abort. Expect $\chi'[i, \sigma'[i, j]]$ to be opened to 0 (if not already), else abort.

When $\pi$ terminates, terminate with the same output.

Figure 4: UC-secure compilation of passive-secure protocol $\pi$ in the $\mathcal{F}_{\mathsf{com}}$-hybrid world

delivering the output in the ideal world. If simulated Bob terminates successfully, the simulator delivers the output. Note that the adversary cannot successfully deviate from the steps of $\pi$ that are inconsistent with having input $x$, by virtue of there being a consistent $\chi[i, \cdot]$ and $\sigma[i, \cdot]$ uniquely committed to $x$. If the adversary tries sending a bit in $\pi$ inconsistent with its input, there will be no way for it to consistently open this particular committing $\sigma[i, \cdot]$ and $\chi[i, \cdot]$. Since $f(x, y) = f(x, y^*)$, the steps of $\pi$ simulated by the simulator must be statistically indistinguishable. Furthermore, the commitments opened by the simulator are distributed identically as when Bob runs the protocol with input $y$. By the security of $\pi$, the simulation is indistinguishable from the real-world interaction (though with an added error probability of $2^{-k}$). □

To compile a randomized protocol, we make each party commit to its random tape as well. However, security only holds if the random tape is chosen uniformly. To ensure this, we use a

coin-tossing-into-the-well approach. Let $S$ be the possible space of (input, random tape) pairs. On input $x$, Alice chooses a random tape $r$ and "commits" to $(x, r)$ by committing to many random permutations and permuted characteristic vectors. For each pair of commitments, Bob randomly asks Alice to completely open one or the other. Then Bob chooses another random tape $s$ (for Alice) and sends it to Alice. Now Alice runs each step of the protocol on $x$ with random tape $r \oplus s$. Both parties know $s$, so on Alice's move, she sends the message $m$, then opens the commitments corresponding to the $(x', r')$ pairs such that the protocol prescribes sending a message other than $m$ on input $x'$ and random tape $r' \oplus s$.

# E Proof of Theorem 5

**Lemma 8.** *Let $\pi$ be an arbitrary protocol for two parties to agree on a value in a domain $Z$ with probability 1. (That is, the* outcome *of the protocol is a function of the transcript.) Then, for every partition $Z = C \cup D$, the protocol $\pi$ falls into one of the following types:*

- *"A" if Alice has a strategy to force the outcome to be a value in $C$ with probability 1, and a strategy to force the outcome to be a value in $D$ with probability 1.*
- *"B" if Bob has a strategy to force the outcome to be a value in $C$ with probability 1, and a strategy to the outcome to be a value in $D$ with probability 1.*
- *"C" if both Alice and Bob have a strategy to force the outcome to be in $C$ with probability 1.*
- *"D" if both Alice and Bob have a strategy to force the outcome to be in $D$ with probability 1.*

*Proof.* The proof is by induction on the maximum number of rounds in $\pi$, when $\pi$ is described in the normal form for protocols. If $\pi$ has 0 rounds, $\pi$ has a constant outcome. So for any partition $Z = C \cup D$, $\pi$ is a C-type or D-type protocol.

Assuming the inductive hypothesis to hold for protocols up to $n - 1$ rounds, we consider a protocol $\pi$ of $n$ rounds. Suppose the first message is sent by Alice. Without loss of generality, suppose the each message in the protocol is a single bit, and that the parties alternate rounds. Then we define two protocols $\pi_0$ and $\pi_1$ which are defined as the protocols obtained by assuming the first message in an execution of $\pi$ to be 0 or 1, respectively. Now, by the inductive hypothesis, $\pi_0$ and $\pi_1$ are one of the four types. We need to consider all combinations of possibilities, but up to symmetries, there are only the following five cases:

1. If $\pi_0$ is of A-type so is $\pi$, because Alice can first choose to send 0, and then follow the strategies in $\pi_0$.
2. If $\pi_0$ is of B-type and $\pi_1$ is of C-type, then $\pi$ is of C-type: Alice can force a $C$ outcome by sending 1 and following her strategy in $\pi_1$ after that; for Bob, in both $\pi_0$ and $\pi_1$ he has a strategy to force the outcome to be in $C$, one of which he will be able to follow after the first message from Alice.
3. If both $\pi_0$ and $\pi_1$ are of B-type, then $\pi$ is of B-type.
4. If $\pi_0$ is of C-type and $\pi_1$ is of D-type, then $\pi$ is of A-type.
5. If both $\pi_0$ and $\pi_1$ are of C-type then $\pi$ is of C-type. □

**Lemma 9.** *If $f$ is an SSFE function that is decomposable but not uniquely decomposable, then there is a partition of the range of $f$ as $Z = C \cup D$ such that Alice has no $x$ such that $\forall y$, $f(x, y) \in C$, and Bob has no $y$ such that $\forall x$, $f(x, y) \in D$.*

*Proof.* Below, in Claim 1, we show that if $f$ has two distinct decompositions, then there is a decomposition of $f$ that contains a step $X' \times Y' \subseteq X \times Y$ where $f\big|_{X' \times Y'}$ is both row- and column-decomposable. Without loss of generality, we may assume that in every decomposition step, say $X = X_0 \cup X_1$, the set of outputs $f(X_0, Y)$ is disjoint from that of $f(X_1, Y)$. Renaming the outputs of a function in this way results in a function that is isomorphic to the original function.

We show by induction on the depth of this decomposition that the outputs of $f$ can be colored as $C$ and $D$ in the desired way. For a function $f$ which is both row- and column-decomposable, say as $X \times Y = (X_1 \cup \cdots \cup X_m) \times (Y_1 \cup \cdots \cup Y_n)$, we color the outputs of $f$ as a "checkerboard" — that is, if $(x, y) \in X_i \times Y_j$ where $i + j$ is odd, then we color $f(x, y)$ with $C$ (and color $D$ if $i + j$ is even). In this way, since $m, n \geq 2$, the condition of the lemma holds for $f$.

Otherwise, suppose $f$ is row-decomposable as $X \times Y = (X_1 \cup \cdots \cup X_n) \times Y$, and some $f\big|_{X_i \times Y}$ can be colored to satisfy the condition of the lemma. Then we color the outputs of $X_i \times Y$ inductively as above, and color all other outputs in $X \times Y$ as $C$. Similarly, if $f$ is column-decomposable, then we inductively color part of its domain and color the other outputs $D$. Inductively, the condition of the lemma holds with this coloring. □

**Claim 1.** *If $f$ has two distinct decompositions, then there is a decomposition of $f$ that contains a step $X' \times Y' \subseteq X \times Y$ where $f\big|_{X' \times Y'}$ is both row- and column-decomposable.*

*Proof.* Take any two distinct decompositions, and consider the first level at which they differ. Without loss of generality, assume that they differ at the very first step. If one decomposition of $f$ starts with a row-decomposition step and the other starts with a column-decomposition step, then we are done. Otherwise, $f$ is decomposable in two different ways, startying with (by symmetry) two *different* row-decomposition steps; say, $X = (X_1 \cup \cdots \cup X_m) = (X'_1 \cup \cdots \cup X'_n)$, with $\{X_1, \ldots, X_m\} \neq \{X'_1, \ldots, X'_n\}$. Then by symmetry, there must be some $X_i$ and $X'_j$ such $X_i \cap X'_j$ and $X_i \cap \overline{X'_j}$ are both non-empty. One can easily verify that $X_i \times Y$ can be row-decomposed starting with the step $X_i = (X_i \cap X'_j) \cup (X_i \setminus X'_j)$. In particular, this implies that $f$ is not constant over $X_i \times Y$. Since $X_i \times Y$ appears in one of the two valid row-decompositions of $f$, and $f$ is not constant over this domain, and row- and column- decomposition steps must alternate, we have that $X_i \times Y$ is also column-decomposable. □

The above lemmas establish one direction of Theorem 5, as outlined in the proof sketch in Section 5. By Lemma 8, for all ways to color the outputs of $f$, there are corresponding adversarial strategies to force a certain color of output, in the real-world protocol execution. However, by Lemma 9, there is a way to color the outputs of a non-uniquely-decomposable function $f$ in such a way that no ideal-world strategy can accomplish the same effect.