A Unified Characterization of Completeness and Triviality for Secure Function Evaluation

Hemanta K. Maji * Manoj Prabhakaran † Mike Rosulek ‡

September 18, 2012

Abstract

We present unified combinatorial characterizations of completeness for 2-party secure function evaluation (SFE) against passive and active corruptions in the information-theoretic setting, so that all known characterizations appear as special cases.

In doing so we develop new technical concepts. We define several notions of isomorphism of SFE functionalities and define the *"kernel"* of an SFE functionality. An SFE functionality is then said to be *"simple"* if and only if it is strongly isomorphic to its kernel. An SFE functionality $\mathcal{F}'$ is a core of an SFE functionality $\mathcal{F}$ if it is "redundancy free" and is weakly isomorphic to $\mathcal{F}$. Then:

- An SFE functionality is complete for security against passive corruptions if and only if it is not simple.
- A deterministic SFE functionality is complete for security against active corruptions if and only if it has a core that is not simple. We conjecture that this characterization extends to randomized SFE as well.

We further give explicit combinatorial characterizations of simple SFE functionalities.

Finally, we apply our new notions of isomorphism to reduce the problem of characterization of trivial functionalities (i.e., those securely realizable without setups) for the case of general SFE to the same problem for the case of simple symmetric SFE.

*University of California, Los Angeles. Supported by NSF CI Postdoctoral Fellowship. hmaji@cs.ucla.edu.
†University of Illinois, Urbana-Champaign. Supported by NSF grant CNS 07-47027. mmp@illinois.edu.
‡University of Montana. Supported by NSF grant CCF-1149647. mikero@cs.umt.edu.

Contents

1 Introduction

Two party secure function evaluation (SFE) is a fundamental concept in modern cryptography. In a seminal work, Kilian [Kil88] introduced the notion of a *complete* function for SFE: given access to an ideally secure implementation of a complete function, every SFE function can be securely implemented using a protocol, without relying on any computational assumptions. He showed that the oblivious transfer functionality (OT) is complete for secuity against active corruption. Earlier results [GMW87, GV87, HM86] already implied that OT is complete for security against passive corruption. Since then several works have *characterized* which functions are complete, under passive and active corruption, in the information-theoretic setting. While complete functionalities are in a sense the "most complex" functionalities, at the other extreme are the *trivial* functionalities which are not useful as setups, because they can be securely realized from scratch.

In this work we develop a unified framework for stating these results in the information-theoretic setting. Unlike previous characterizations, we do not give separate characterizations for SFE with one output or two outputs[1] or for sub-classes of deterministic and randomized SFE. We summarize our definitions and characterizations below. For simplicity, we restrict ourselves to "finite" functionalities through out, though the results do extend to functionalities with *polynomial-sized domains* (but not necessarily exponential-sized domains).

Our Results. We define strong and weak isomorphisms among SFE functionalities (Definition 5). We also use the notion of the kernel of an SFE functionality (Definition 3). We define an SFE functionality to be simple if it is strongly isomorphic to its kernel. For characterizing completeness and triviality against active corruption, we define an SFE functionality $\mathcal{F}'$ to be a core of an SFE functionality $\mathcal{F}$ if it is "redundancy free" and is weakly isomorphic to $\mathcal{F}$.

Completeness. For the case of completeness for security against passive adversaries (passive-completeness, for short), we obtain a complete characterization by piecing together and extending known results for symmetric and asymmetric SFE. In the case of (standalone or UC) security against active corruption, we identify a gap in the known characterizations, but our unified presentation gives a natural conjecture to fill this gap.

Our characterizations for completeness are as follows.

- A (possibly randomized) SFE functionality is passive-complete if and only if it is not simple (e.g. Theorem 1).
- A *deterministic* SFE functionality is UC or standalone-complete if and only if it has a core that is not simple (Theorem 2). The same characterization holds for UC/standalone-completeness of "channel" functionalities as well. We conjecture that this characterization holds for UC/standalone-completeness of all SFE functionalities.

Triviality. It has been known that a *deterministic* SSFE functionality is passive-trivial if and only if it is "decomposable" and is active-trivial if and only if it is "saturated" [Kus89, MPR09, KMR09].

[1] SFE functionalities which produce only one output have been considered in the literature either in the form of"symmetric" SFE (a.k.a. SSFE, which give the output to both parties) or in the form of "asymmetric" SFE (which give the output to only one party).

These characterizations were extended to general (not necessarily symmetric) SFE in [KMR09]. Our contribution is in characterizing passive-triviality for general (not necessarily symmetric) SFE in terms of characterization of passive-triviality for SSFE, in a manner that applies to *both deterministic and randomized SFE*. Briefly, we show that:

- An SFE functionality $\mathcal{F}$ is passive-trivial if and only if it is simple and its kernel (which is a simple SSFE functionality) is passive-trivial.
- An SFE functionality is standalone-trivial if and only if it has a simple core whose kernel (which is a simple SSFE functionality) is standalone-trivial.

Interestingly, the characterization of passive-trivial and standalone-trivial *randomized* SFE still remains open. If this characterization is carried out for simple SSFE functionality, then our results show how to extend it to general SFE.

We heavily rely on prior work which gave characterizations of completeness and triviality for various special cases. Our main contribution is perhaps in identifying how the different results can be unified using arguably elegant definitions. For instance, we unify the characterization of UC- and standalone-completeness for deterministic SFE [KM11] and for randomized channels [CMW04]; further our formulation gives a plausible conjecture for extending these characterizations to all SFE functionalities.

Related Work. The first complete functionality that was discovered was Oblivious Transfer. It was shown to be complete in the information-theoretic setting, against passive adversaries in [GMW87, GV87, HM86], and against active adversaries in [Kil88] (and explicitly extended to UC security in [IPS08]). All subsequent completeness results build on this.

Completeness (w.r.t. security against passive and active corruption) was characterized in [Kil91, Kil00, CK88, CMW04, KM11], for subclasses of SFE functionalities. We observe that a result in [MOPR11] implicitly extended the characterization of completeness w.r.t. security against passive corruption to all SFE functionalities. (Our simpler characterization is proven based on these results.)

On the front of triviality, seminal results in secure multi-party computation established that *all* functionalities are trivial for passive and standalone security, either under computational assumptions [Yao86, GMW87] or, in the setting involving more than two parties, under restrictions on how many parties can be corrupted [CCD88, BGW88]. These results are mostly not applicable for the setting we are considering (information-theoretic security for two-party computation). [Kus89] introduced the notion of "decomposability" and used it to characterize passive-trivial functionalities. Initial proofs considered only perfect security [Kus89, Bea89], but later results extended this to the case of statistical security [MPR09, KMR09]. Triviality for UC-security has a relatively simpler characterization and is well-understood [CKL03, PR08].

2 Preliminaries

A two-party *secure function evaluation (SFE)* functionality $\mathcal{F}(f_A, f_B)$ is a trusted party whose behavior is specified using two functions $f_A : X \times Y \times R \to Z_A$ and $f_B : X \times Y \times R \to Z_B$.

The trusted party takes inputs $x \in X$ from Alice and $y \in Y$ from Bob, samples a private local randomness $r \xleftarrow{\$} R$ and evaluates $a = f_A(x, y, r)$ and $b = f_B(x, y, r)$, the respective outcomes of Alice and Bob. A *deterministic* SFE is one for which $|R| = 1$.

We consider, in the information-theoretic setting, security against adversaries who may be passive (a.k.a. honest-but-curious) or active (a.k.a. byzantine, or malicious). In the latter case, the adversary could be standalone (does not interact with the environment during the course of the protocol, except via protocol input/output for reactive functionalities), or not. Correspondingly, we have three notions of security: *passive security*, *standalone security* and *UC security*. In settings involving active adversaries we consider *security with abort* (if either party is corrupt, the functionality delivers the output of the corrupt party first and then delivers the output to the honest party when instructed by the corrupt party); guaranteed output security notions are beyond the scope of this work.

We adopt the following useful categories of SFE from the literature:

- *Symmetric SFE (SSFE),* for which $f_A = f_B$. That is, both the parties get the same output.
- *Asymmetric SFE,* for which either f_A or f_B is a constant function. In other words, only one party gets output. A special case of an asymmetric SFE is a *channel* in which the party receiving the output has no input (i.e., if f_B is not a constant function, then $|Y| = 1$).
- There are SFE functionalities which fall into neither of these classes. Sometimes we will use the term *general SFE* to stress that we are considering an SFE which is not necessarily of the above two types.

We restrict ourselves to the universe of *finite* SFE functionalities: the input and output spaces are finite — that is, have size $O(1)$, as a function of the security parameter. In particular, the maximum number of bits needed to represent the input to the parties (and, in the case of randomized functionalities, the number of bits in the random tape of the functionality) does not grow with the security parameter. We remark that for simplicity we considered the distribution over R to be uniform (but $|R|$ need not be a power of 2). However, any fixed arbitrary distribution (which does not change with the security parameter) could be considered, without affecting our results.

Security Notions and Hybrids. The *real-ideal* paradigm for security [GMW87] is used to define security for multi-party computation. Informally, a protocol *securely realizes* a functionality if, for every adversary attacking the actual protocol in the real world, there is a corresponding ideal world adversary (called the simulator) which can achieve the same effect in any environment. Depending on the type of the security required, the capabilities of the adversary vary. We consider three kinds of security: security against passive adversaries (passive-security, for short), security against active adversaries (standalone-security) and Universally Composable security (UC-security). In passive-security the adversary follows the protocol exactly and the ideal world adversary (simulator) does not alter the input it sends to the functionality. In standalone security the adversary can actively corrupt the parties, but it does not interact with the outside environment during the course of the protocol execution.

We consider secure realization of functionalities in presence of "setups" as well. A $\mathcal{G}$-hybrid is a world where trusted implementation of the functionality $\mathcal{G}$ is accessible to both parties. In the

real world, parties can communicate via private communication channel (like the plain model) as well as invoke evaluations of $\mathcal{G}$. In the ideal world, the simulator pretends to provide the access of such a setup to the adversary. A protocol π in the $\mathcal{G}$-hybrid, represented as $\pi^{\mathcal{G}}$, securely realizes a functionality $\mathcal{F}$ if for every real world adversary, there exists an ideal world simulator which can simulate identical behavior.

If a functionality has a secure protocol (in some security model) without any setup (i.e., the functionality is *realizable* in that security model), then it is of no value as a setup, as the access to such an ideal functionality can be replaced by an implementation. We shall refer to such functionalities as *trivial functionalities* (for the corresponding security model or reduction). In terms of reducibility, a trivial functionality reduces to every functionality. At the other extreme, we can consider functionalities to which every functionality reduces. Such functionalities, any of which can replace any other setup, are called *complete functionalities* (for the corresponding security model or reduction).

For brevity, we shall write "*passive-trivial*," "*UC-complete*," etc. to stand for "trivial w.r.t. reductions that are secure against passive adversaries," "complete w.r.t. reductions that are UC-secure" etc.

3 Definitions: Isomorphism, Kernel, Simple SFE and Core

In this section, we define the terms useful in stating unified completeness results for 2-party SFE in various security notions.

3.1 Graph of an SFE Functionality

Given a 2-party SFE $\mathcal{F}(f_A, f_B)$ we define a bipartite graph $\mathrm{G}(\mathcal{F})$ as follows.

Definition 1 (Graph of a 2-party SFE) *Given a SFE functionality $\mathcal{F}(f_A, f_B)$, its corresponding* graph $\mathrm{G}(\mathcal{F})$ *is a weighted bipartite graph constructed as follows. Its partite sets are $X \times Z_A$ and $Y \times Z_B$. For every $(x, a) \in X \times Z_A$ and $(y, b) \in Y \times Z_B$, the edge joining these two vertices is assigned weight*

$$\mathsf{wt}\Big((x,a),(y,b)\Big) := \frac{\Pr\limits_{r \xleftarrow{\$} R}\left[f_A(x,y,r) = a \ \wedge \ f_B(x,y,r) = b\right]}{|X \times Y|}.$$

The choice of the normalizing constant $1/|X \times Y|$ is arbitrary. For this particular choice of constant, we can view the weight of an edge as representing the joint-distribution probability of input-output pairs seen by the two parties when $(x, y, r) \xleftarrow{\$} X \times Y \times R$.

We remark that such representations have appeared in the literature for a long time. In particular, on squaring the (the unweighted version of) the above bipartite graph, the two parts separate into two *characteristic graphs* as defined by Witsenhausen [Wit76] for the correlated source which samples input-output pairs for the two parties. The bipartite graph itself, but again for correlated

sources, has appeared in later works, both in information theory (e.g. [KTRR03]) and in cryptography (e.g. [WW06]). The graph $\mathrm{G}(\mathcal{F})$ as defined above for SFE functionalities was considered in [MOPR11] (for proving the result mentioned in Footnote 3).

For a combinatorial characterization of what we shall define as a simple SFE functionality, the following definition will be useful.

Definition 2 (Product Distribution Graph) *A weighted bipartite graph with partite sets U and V and weight function* wt *is a* product distribution graph *if there exist*

1. *non-empty partitions $\{U_1, \ldots, U_n\}$ and $\{V_1, \ldots, V_n\}$ of U and V respectively, and*
2. *probability distributions p over U, q over V, and c over $[n]$,*

such that for all $u \in U$ and $v \in V$, the weight on edge (u, v) is given by

$$\mathsf{wt}(u,v) = \begin{cases} p_u \cdot q_v / c_k & \text{if } \exists k \in [n], \text{ s.t. } \ c_k > 0, \text{ and } u \in U_k,\ v \in V_k, \\ 0 & \text{otherwise.} \end{cases}$$

Intuitively, a bipartite graph G is a product distribution graph if sampling an edge of G corresponds to first sampling a *connected component* of G, and then within that component further sampling two nodes *independently* from the two partite sets of the component. Thus when $u \in U_k$ and $v \in V_k$, $\mathsf{wt}(u,v) = c_k \cdot (p_u/c_k) \cdot (q_v/c_k)$, where c_k is the probability of selecting the k^{th} component and p_u/c_k (resp. q_v/c_k) is the probability of sampling u (resp. v) conditioned on selecting the k^{th} component.

Given an SFE functionality $\mathcal{F}$, it is convenient to define an associated SSFE functionality as the "common information" that Alice and Bob both get from $\mathcal{F}$ [MOPR11].

Definition 3 (Kernel— Common-information in a SFE) *The* kernel *of a SSFE $\mathcal{F}$ is a symmetric SFE which takes inputs x and y from the parties, samples $r \xleftarrow{\$} R$ and computes $a = f_A(x, y, r)$ and $b = f_B(x, y, r)$. Then it outputs to both parties the connected component of $\mathrm{G}(\mathcal{F})$ which contains the edge $\Big((x, a), (y, b)\Big)$.*

Note that the kernel of $\mathcal{F}$ is a symmetric functionality and is randomized only for randomized SFE $\mathcal{F}$. For example, let $\mathcal{F}$ be (possibly biased) symmetric coin tossing functionality. Its kernel is a randomized functionality and, incidentally, is identical to $\mathcal{F}$ itself. Further, kernel of a kernel is the kernel itself.

3.2 Isomorphisms

We introduce a couple of notions of isomorphism between SFE functionalities, which we use in all our subsequent definitions. The definitions of isomorphism presented here are refinements of a notion of isomorphism in [MOPR11], which in turn was preceded by similar notions (restricted to deterministic SFE) in [KMR09, MPR09].

Crucial to defining isomorphism is the following notion of "locality" for a protocol:

Definition 4 (Local Protocol) *In a* local protocol for $\mathcal{F}$ which uses $\mathcal{G}$ as a setup*, each party maps her $\mathcal{F}$-input to a $\mathcal{G}$-input, calls $\mathcal{G}$ once with that input and, based on her local view (i.e. her given $\mathcal{F}$-input, the output of $\mathcal{G}$, and possibly local randomness), computes her final output, without any further communication between the parties.*

Definition 5 (Isomorphism) *We say that $\mathcal{F}$ and $\mathcal{G}$ are* strongly isomorphic *to each other if there exist two local protocols π_1 and π_2 such that:*

1. *$\pi_1^{\mathcal{G}}$ UC-securely realizes $\mathcal{F}$ and $\pi_2^{\mathcal{F}}$ UC-securely realizes $\mathcal{G}$;*
2. *$\pi_1^{\mathcal{G}}$ passive-securely realizes $\mathcal{F}$ and $\pi_2^{\mathcal{F}}$ passive-securely realizes $\mathcal{G}$; and*
3. *$\mathcal{F}$ and $\mathcal{G}$ have the same input domains, and in $\pi_1^{\mathcal{G}}$ and $\pi_2^{\mathcal{F}}$, the parties invoke the given functionality with the same input as they get from the environment.*

$\mathcal{F}$ and $\mathcal{G}$ are said to be isomorphic *to each other if conditions 1 and 2 are satisfied. $\mathcal{F}$ and $\mathcal{G}$ are said to be* weakly isomorphic *to each other if condition 1 is satisfied.*

A few remarks on the definition mentioned above are in order.

1. It is not hard to see that Condition 1, which is required by all three definitions of isomorphisms, is equivalent to the (seemingly weaker) condition obtained by replacing UC-security with standalone security. This is because of the nature of a local reduction.

2. Condition 2 might seem weaker than Condition 1 since the former requires security against a weaker adversary. But security against a weaker adversary is not always a weaker requirement, since it requires that the ideal-world adversaries (simulators) are also weaker (passive, in this case).

3. All these notions of isomorphism are equivalence relations. In particular, they are transitive due to secure composition of local reductions (under all three notions of security).

4. Another consequence of secure composition is that isomorphism (and hence strong isomorphism) preserves UC and standalone reducibility, as well as reducibility against passive adversaries, between functionalities, and weak isomorphism preserves UC and standalone reducibility (but not necessarily passive reducibility) between functionalities. For example, if $\mathcal{F}$ UC-securely (resp., standalone-securely or passive-securely) reduces to $\mathcal{G}$, and $\mathcal{F}$ and $\mathcal{F}'$ are isomorphic to each other, and $\mathcal{G}$ and $\mathcal{G}'$ are isomorphic to each other, then $\mathcal{F}'$ UC-securely (resp., standalone-securely or passive-securely) reduces to $\mathcal{G}'$.

As we shall see shortly, an important property of an SFE functionality is whether or not it is (strongly) isomorphic to its kernel.

Definition 6 (Simple SFE) *A (possibly randomized) SFE functionality $\mathcal{F}$ is said to be* simple *if it is strongly isomorphic to its kernel.*

We shall see from our characterizations that the above definition remains unaltered if strong isomorphism is replaced by isomorphism (see Lemma 3). However, for *deriving* the characterizations, it is convenient to use the stricter notion of isomorphism in this definition.

Though Definition 6 is in terms of isomorphism, below we give an explicit combinatorial characterization of simple SFE functionalities. This combinatorial characterization will be useful in seeing how our definition unifies several definitions in the literature for special cases (in Section 3.3).

Lemma 1 *The following statements are equivalent.*

1. $\mathcal{F}$ *is simple.*
2. $\mathrm{G}(\mathcal{F})$ *is a product distribution graph.*
3. *For any nodes* $u_0, u_1 \in X \times Z_A$ *and* $v_0, v_1 \in Y \times Z_B$*, the weights in* $\mathrm{G}(\mathcal{F})$ *satisfy*

$$\mathsf{wt}(u_0, v_0)\mathsf{wt}(u_1, v_1) = \mathsf{wt}(u_0, v_1)\mathsf{wt}(u_1, v_0)$$

We prove Lemma 1 in Appendix A.

3.3 Special cases of simple SFE

The definition of simple functionalities that we presented above unifies several definitions that appeared in the literature for special classes of functionalities.

- **Deterministic symmetric SFE.** The first instance where simple functionalities were identified was for the special case of deterministic symmetric SFE: in this case a functionality is *not* simple if and only if the matrix representing the function f has an "OR minor" (i.e., $\exists x_0, x_1, y_0, y_1, z_0, z_1$ with $z_0 \neq z_1$ and $f(x_a, y_b) = z_{a \vee b}$, for $a, b \in \{0, 1\}$) [Kil91].
- **Randomized symmetric SFE.** In [Kil00] this was generalized to randomized symmetric SFE functionality: in this case (as described in Appendix B) a functionality is not simple iff $\exists x_0, x_1, y_0, y_1, z$ such that

$$\Pr[f(x_0, y_0) = z] > 0; \text{ and } \Pr[f(x_0, y_1) = z] > 0; \text{ and}$$
$$\Pr[f(x_0, y_0) = z] \cdot \Pr[f(x_1, y_1) = z] \neq \Pr[f(x_1, y_0) = z] \cdot \Pr[f(x_0, y_1) = z].$$

  It is easy to see that this is a generalization of the previous definition by setting $z = z_1$.
- **Randomized asymmetric SFE.** In [Kil00], the characterization of simple functionalities, specialized to the case of randomized *asymmetric* SFE too appears. Kilian gives a combinatorial condition for being non-simple, but also notes (the more intuitive characterization) that the condition does not hold (i.e., the functionality *is* simple) if and only if the functionality has a passive-secure protocol which involves a single deterministic message from Alice to Bob. Equivalently, a (possibly randomized) asymmetric SFE is simple if and only if it is strongly isomorphic to a deterministic functionality in which Bob has no input.
- **Deterministic SFE.** Another generalization, this time to deterministic, but general (not necessarily symmetric or asymmetric) SFE, appears in [KM11]: as described in Appendix B, a deterministic SFE functionality is not simple iff it has an "OT-core": i.e., there are inputs x, x' for Alice and y, y' for Bob such that $f_A(x, y) = f_A(x, y')$, $f_B(x, y) = f_B(x', y)$ and $\Big(f_A(x', y), f_B(x, y')\Big) \neq \Big(f_A(x', y'), f_B(x', y')\Big)$.

All these special cases of the definition of simple functionalities were identified to characterize complete functionalities (see Theorem 1 and Theorem 2).

3.4 Redundant Inputs and Core of an SFE functionality

To study security against active adversaries alone (i.e., not also against passive adversaries) it is useful to have a notion of "redundant" inputs of an SFE functionality that will never be needed by an active adversary. Combinatorial definitions of redundancy have appeared in the literature before for special classes of SFE functionalities, but our definition is in terms of weak isomorphism and applies to all SFE functionalities.

To state our definition we use the following notation. Given a function $f : X \times Y \times R \to Z$, and $x \in X$ (resp. $y \in Y$), let $f|_{X\setminus\{x\}}$ (resp. $f|_{Y\setminus\{y\}}$) denote the function obtained by restricting f to the domain $(X\setminus\{x\}) \times Y \times R$ (resp. $X \times (Y\setminus\{y\}) \times R$). For a functionality $\mathcal{F}(f_A, f_B)$, and $x \in X$ (resp. $y \in Y$), let $\mathcal{F}|_{X\setminus\{x\}}$ (resp. $\mathcal{F}|_{Y\setminus\{y\}}$) denote the functionality $\mathcal{F}'(f_A|_{X\setminus\{x\}}, f_B|_{X\setminus\{x\}})$ (resp. $\mathcal{F}'(f_A|_{Y\setminus\{y\}}, f_B|_{Y\setminus\{y\}})$).

Definition 7 (Redundant Inputs) *A functionality $\mathcal{F}$ with input domain $X \times Y$ is said to have a* redundant input *$x \in X$ (resp. $y \in Y$) if $\mathcal{F}$ is weakly isomorphic to $\mathcal{F}|_{X\setminus\{x\}}$ (resp. $\mathcal{F}|_{Y\setminus\{y\}}$). $\mathcal{F}$ is said to be* redundancy-free *if it has no redundant inputs.*

We highlight two special cases:

- For deterministic SFE functionalities, Alice's input x is redundant iff there is an input $x' \neq x$ that *dominates* x: i.e., Alice can substitute x' for x without Bob noticing (i.e., for all inputs y of Bob, $f_B(x, y) = f_B(x', y)$) while still allowing her to calculate her correct output (i.e., there is a deterministic mapping $T_{x,x'}$ such that for all inputs y of Bob, $f_A(x, y) = T_{x,x'}(f_A(x', y))$).
- For (possibly randomized) asymmetric functionalities (in which only Bob receives a non-constant output), Alice's input x is redundant iff Alice could instead send a "convex combination" of other inputs to achieve the same effect for Bob. That is, there exists $x_1, \ldots, x_k \in X$, $r_1, \ldots, r_k \in \mathbb{R}$ with $x \notin \{x_1, \ldots, x_k\}$, $\sum_i r_i = 1$ and for all $y \in Y$, we have $f_B(x, y) \equiv \sum_i r_i f_B(x_i, y)$. In the previous expression, the output of $f_B(\cdot, \cdot)$ is interpreted as a probability distribution over Z_B (equivalently, a stochastic vector in $\mathbb{R}^{|Z_B|}$).

Definition 8 (Core) *An SFE functionality $\mathcal{F}'$ is said to be a* core *of an SFE functionality $\mathcal{F}$ if $\mathcal{F}$ and $\mathcal{F}'$ are weakly isomorphic to each other and $\mathcal{F}'$ is redundancy-free.*

For any SFE functionality, one can find a core by successively removing from its domain, one at a time, inputs that are redundant (based on the set of inputs that have not been removed yet). To see this, note that if $\mathcal{F}'$ is obtained by removing a single redundant input from the domain of $\mathcal{F}$, then by definition of being redundant, $\mathcal{F}$ and $\mathcal{F}'$ are weakly isomorphic to each other. This process must terminate after a constant number of steps, since the domains are finite. By transitivity of weak isomorphism the final redundancy free functionality obtained is indeed weakly isomorphic to $\mathcal{F}$ and hence a core of $\mathcal{F}$.

From this and the transitivity of weak isomorphism, it follows that the core of $\mathcal{F}$ is *unique* up to weak isomorphism. In fact, for the two special cases of deterministic SFE and randomized channel SFE that are required in Theorem 2 and Theorem 4, the core of $\mathcal{F}$ is unique with respect to (plain) isomorphisms whose input mapping is a bijection. For the former case, this was explicitly observed

in [KM11] (where a core of $\mathcal{F}$ was called the redundancy-free version of $\mathcal{F}$). For the latter case, consider the set of points in $\mathbb{R}^{|Z_B|}$ denoting the probability distributions of $f_B(x)$; then the inputs in a core correspond to the vertices of the convex-hull of this set of points. (If the points for multiple values of x coincide on a vertex of the convex-hull, a core will retain exactly one of these inputs.)

To characterize active security, one will typically consider only a core of the functionality. Redundancy-free functionalities also have the convenient property that a protocol for a redundancy-free functionality that is secure against active adversaries is also secure against passive adversaries:[2]

Lemma 2 *If $\mathcal{F}$ is redundancy-free, then any protocol for $\mathcal{F}$ that is standalone-secure is also passive-secure.*

Note that this is not true in general (i.e., when $\mathcal{F}$ has redundant inputs). In a general active-secure protocol, the simulator for a passively corrupt adversary may not be a passive ideal adversary.

Proof: Let π be such a protocol for $\mathcal{F}$. It suffices to show that in π, the simulator for a passive adversary is without loss of generality passive itself. By symmetry, consider a passive dummy adversary $\mathcal{A}$ for Alice, which runs the protocol honestly and outputs its entire view. Note that $\mathcal{A}$ receives an actual input x from the environment.

Let $\mathcal{S}$ be the simulator for $\mathcal{A}$. Let x denote the input provided by the environment, and let $\mathcal{E}_x$ denote the event that $\mathcal{S}$ sends something other than x to $\mathcal{F}$. If for all x, $\mathcal{E}_x$ is negligible, then we are essentially done. We can modify $\mathcal{S}$ to always send x to $\mathcal{F}$; the interaction's outcome changes only by a negligible amount and hence the modified $\mathcal{S}$ is a passive ideal adversary and a valid simulator for $\mathcal{A}$.

Otherwise, fix an x such that $\mathcal{E}_x$ is non-negligible. Then there is a way to condition the randomness of $\mathcal{S}$ so that $\mathcal{E}_x$ always occurs, and the outputs reported by both parties is indistinguishable from the correct output, for all possible inputs of Bob. Call the resulting simulator $\mathcal{S}_x$. Then the following is a local protocol for $\mathcal{F}$ using $\mathcal{F}|_{X\setminus\{x\}}$: Bob runs the dummy protocol; if Alice's input is not x, then she runs the dummy protocol. If Alice's input is x, she runs $\mathcal{S}_x$ and reports the prescribed output of the simulated adversary. The properties established for $\mathcal{S}_x$ show that this is a secure protocol for $\mathcal{F}$ in which Alice never uses input x. Since there is always a local protocol for $\mathcal{F}|_{X\setminus\{x\}}$ using $\mathcal{F}$, we have that $\mathcal{F}$ and $\mathcal{F}|_{X\setminus\{x\}}$ are weakly isomorphic, so x is redundant in $\mathcal{F}$. This contradicts the redundancy-freeness of $\mathcal{F}$, so this case cannot happen. □

4 Completeness of Two Party Functionalities

The first complete functionality that was discovered was Oblivious Transfer. It was shown to be complete against passive adversaries in [GMW87, GV87, HM86], and against active adversaries in [Kil88] (and explicitly extended to UC security in [IPS08]). All subsequent completeness results build on this.

We have a full understanding of SFE functionalities that are complete under security against *passive* adversaries.

[2] Thus redundancy-free functionalities are a special case of what are called "deviation-revealing functionalities" [PR08], a notion that is defined more generally for reactive functionalities.

Theorem 1 *A finite (possibly randomized) 2-party SFE functionality is passive-complete in the information theoretic setting if and only if it is not simple.*

The first step towards such a characterization was taken by Kilian, for the special case of deterministic symmetric SFE [Kil91]. As mentioned before, for this case the complete functionalities are those with an OR minor. Later, Kilian extended it to the setting of randomized, *symmetric* SFE functionalities, and also for randomized *asymmetric* SFE functionalities [Kil00]. [KM11] includes the case of deterministic general SFE. We observe that a result in [MOPR11] can be used to obtain the complete characterization.[3] Our proof below directly uses Kilian's characterization (rather than extending Kilian's protocol as in [MOPR11]) along with Lemma 1.

Proof: [Proof of Theorem 1]

For the first direction, suppose for contradiction that $\mathcal{F}$ is passive-complete and it is simple. This implies that $\mathcal{K}$, the kernel of $\mathcal{F}$, is also passive-complete. Now, we shall invoke the completeness characterization of randomized symmetric functionalities SFE by Kilian [Kil00] to show that $\mathcal{K}$ is not complete. Let $U_{x,k} := (\{x\} \times Z_A) \cap U_k$ be the set of nodes in the k-th connected component of $\mathrm{G}(\mathcal{F})$ which are of the form $u = (x, a)$ for some $a \in Z_A$. The probability that a randomly sampled edge lies in the k-th component and its corresponding Alice and Bob inputs are x and y, respectively, is:

$$\sum_{(u',v') \in U_{x,k} \times V_{x,k}} \mathsf{wt}(u', v').$$

But $\mathsf{wt}(u', v') = p_{u'} \cdot q_{v'}/c_k$, because $\mathrm{G}(\mathcal{F})$ is a product distribution graph. So, the previous probability expression can be re-written as:

$$\left(\sum_{u' \in U_{x,k}} p_{u'} \right) \times \left(\sum_{v' \in V_{y,k}} q_{v'} \right) \frac{1}{c_k} = P_{x,k} \times Q_{y,k}.$$

Now it is easy to verify that:

$$\Pr[k|x_0, y_0] \cdot \Pr[k|x_1, y_1] = |X \times Y|^2 P_{x_0,k} P_{x_1,k} Q_{y_0,k} Q_{y_1,k} = \Pr[k|x_0, y_1] \cdot \Pr[k|x_1, y_0],$$

for every $k \in [n]$ and $(x, y) \in X \times Y$. By [Kil00], this implies that $\mathcal{K}$ is not a passive-complete SSFE.

Next, we prove the more interesting direction: if $\mathcal{F}$ is not simple then $\mathcal{F}$ is passive-complete. For this it is enough to show how to use $\mathcal{F}$ to passive-securely realize a channel C in which Alice has two inputs 0 and 1, and the distributions D_0 and D_1 of the output that Bob receives on input 0 and 1 respectively are such that they are not identical, but nor do they have disjoint supports. This is because by a characterization in Kilian [Kil00], such asymmetric non-trivial channels are passive-complete.

[3] [MOPR11] extends the protocol in [Kil00] for asymmetric SFE to show that if an SFE functionality $\mathcal{F}$ is not (strongly) isomorphic to its kernel, then it is complete for security against passive adversaries. (Though the statement in [MOPR11] is not in terms of strong isomorphism, the protocols that establish completeness of $\mathcal{F}$ only uses the condition of $\mathcal{F}$ not being *strongly* isomorphic to its kernel.) On the other hand, a functionality which is (strongly) isomorphic to its kernel is not complete, since the kernel (which is an SSFE) is itself simple and hence not complete by one of the characterizations in [Kil00].

First we describe the channel C we shall securely realize using $\mathcal{F}$. By Lemma 1 we know that $\mathrm{G}(\mathcal{F})$ is not a product distribution graph. Given $u \in X \times Z_A$, we can consider the following distribution D_u over $Y \times Z_B$. The probability of a node $v \in Y \times Z_B$ induced by D_u is: $\mathsf{wt}(u,v)/\sum_{v' \in Y \times Z_B} \mathsf{wt}(u,v')$. Since, $\mathrm{G}(\mathcal{F})$ is not a product distribution graph, there is some connected component with two nodes $u, u' \in X \times Z_A$ such that D_u and $D_{u'}$ are not identical distributions over $Y \times Z_B$. Since u and u' are connected, there is a path $(u = \hat{u}_0, \hat{v}_0, \hat{u}_1, \hat{v}_1, \ldots, \hat{u}_t = u')$ in $\mathrm{G}(\mathcal{F})$. Then there must exist $\hat{u}_i, \hat{u}_{i+1}$ such that $D_{\hat{u}_i}$ and $D_{\hat{u}_{i+1}}$ are not identical. Let $D_0 = D_{\hat{u}_i}$ and $D_1 = D_{\hat{u}_{i+1}}$. Then D_0 and D_1 are not identical, but their supports intersect (at $\hat{v}_i$).

Now we describe how to securely realize the channel C by invoking $\mathcal{F}$ several times. For convenience, let $u_0 = \hat{u}_i$ and $u_1 = \hat{u}_{i+1}$ so that $D_b = D_{u_b}$ for $b \in \{0,1\}$. For $b \in \{0,1\}$, let p_b be the probability that when $\mathcal{F}$ is invoked with random inputs $(x,y) \xleftarrow{\$} X \times Y$, Alice sees outcome a and $(x,a) = u_b$. We know that $\min\{p_0, p_1\} \geq 1/|X \times Y \times R| = \Theta(1)$. To implement the channel, Alice and Bob invoke the functionality $\mathcal{F}$ with uniformly drawn inputs κ times, where κ is the security parameter. Let I_0 and I_1 be the set of indices of the executions where Alice's input-output pair is u_0 and u_1 respectively. With probability at least $1 - 2^{-\Omega(\kappa)}$ both these sets are non-empty. To send a bit b via channel C, Alice sends a random index $i \xleftarrow{\$} I_b$ to Bob and Bob interprets his corresponding input-output pair in the i-th invocation of $\mathcal{F}$ as the output of the channel. It is not hard to show that this is a passive-secure realization of C. □

A consequence of the above characterizations is the following lemma which gives an alternate definition for a simple functionality (where strong isomorphism in Definition 6 is replaced by isomorphism).

Lemma 3 *An SFE functionality $\mathcal{F}$ is simple if and only if it is isomorphic to its kernel.*

Proof: Clearly, if $\mathcal{F}$ is simple, i.e. strongly isomorphic to its kernel $\mathcal{K}$, then it is also isomorphic to its kernel. For the converse, assume for contradiction that $\mathcal{F}$ is isomorphic to its kernel $\mathcal{K}$ but $\mathcal{F}$ is not simple. Since $\mathcal{F}$ is not simple, by Theorem 1, $\mathcal{F}$ is passive-complete. $\mathcal{F}$ is isomorphic to $\mathcal{K}$ implies that $\mathcal{K}$ itself is passive-complete. But kernel of $\mathcal{K}$ is identical to $\mathcal{K}$ and, hence, they are strongly isomorphic to each other. This implies that $\mathcal{K}$ is simple and passive complete — a contradiction by Theorem 1. □

For the case of active corruption, in standalone as well as the UC setting, a characterization of complete functionalities is known for special cases. This was first shown for the special case of deterministic, *asymmetric* SFE (in which f_A is the constant function) by Kilian [Kil00]. The complete characterization for deterministic SFE — including the extension to UC security — is due to Kraschewski and Müller-Quade [KM11], who phrased it in terms of the presence of an OT-core (see Section 3.3). For the case of *channels* (i.e., asymmetric SFE in which only one party has an input and only the other party gets an output), UC and standalone-completeness was characterized in [CK88, CMW04].[4] Our characterization unifies these two results into a common characterization.

Theorem 2 *A finite 2-party SFE functionality that is*

- *deterministic, or*

[4] [CMW04] does not explicitly deal with UC-security. However the simulator implicit in the analysis of the protocol in [CMW04] is a straightline simulator, and can be used to argue UC-completeness as well.

- *a channel*

is UC or standalone-complete in the information theoretic setting if and only if it has a core that is not simple.

Proof: We rely on the characterizations of [KM11] and [CMW04] to prove this result.

First consider the case of deterministic 2-party SFE. Kraschewski and Müller-Quade [KM11] showed that $\mathcal{F}$ is UC or standalone-complete if and only if the "redundancy-free version" of $\mathcal{F}$ has an OT-core (see Section 3.3). For deterministic SFE $\mathcal{F}$, a redundancy-free version of $\mathcal{F}$ in the sense of [KM11] is the same as a core of $\mathcal{F}$, (and in fact is isomorphic to every core of $\mathcal{F}$). Also, as discussed in Section 3.3, a deterministic SFE has an OT-core if and only if it is not simple. Thus the characterization of [KM11] can be recast as saying that a deterministic SFE $\mathcal{F}$ is UC or standalone-complete if and only if it has a core that is not simple (and equivalently, every core of $\mathcal{F}$ is not simple).

Next we consider the case of channels. Crépeau, Morozov and Wolf [CMW04] showed that complete channels are exactly those channels for which, after removing "redundant inputs," the resulting channel is "non-trivial." As we described after Definition 8, for an asymmteric SFE, and in particular for a channel SFE $\mathcal{F}$, redundancy-free version of $\mathcal{F}$ in the sense of [CMW04] is isomorphic to every core of $\mathcal{F}$. Here a trivial channel is what [Kil00] characterized as non-simple (randomized) asymmetric SFE (see Section 3.3). Thus the characterization of [CMW04] too can be recast as saying that a deterministic SFE $\mathcal{F}$ is standalone-complete if and only if it has a core that is not simple (and equivalently, every core of $\mathcal{F}$ is not simple). The proof in [CMW04] can be extended to cover UC-completeness as well. □

Extending this characterization to cover randomized SFE remains an open problem. We conjecture that the same characterization as in Theorem 2 holds for all SFE (and not just deterministic SFE or channel SFE).

5 Characterizing Trivial SFE

There are three main classes of trivial SFE functions, depending on the type of security. The simplest 2-party functionalities are the ones which are trivial under UC security. The functionalities are equivalent to noiseless channels [CKL03]. A much richer class of functionalities is obtained by considering triviality under information theoretic *passive security* (this section), and triviality under information theoretic *standalone active security* (Section 5.2. We focus on these two low-complexity classes below. These two classes have been characterized only restricted to deterministic functionalities. Our characterization reduces the problem of characterizing triviality of general SFE functionalities to the problem of characterizing triviality of simple SSFE functionalities. We remark that it still remains open to give a combinatorial characterization of trivial SSFE outside of determinsitic SFE.

5.1 Passive Trivial SFE

Theorem 3 *A finite 2-party SFE functionality $\mathcal{F}$ is passive-trivial in the information theoretic setting if and only if it is simple and its kernel (which is a simple SSFE functionality) is passive-trivial.*

Proof: If $\mathcal{F}$ is simple, then it is strongly isomorphic to its kernel. Hence, if the latter is passive-trivial, then so is $\mathcal{F}$.

The other direction is a simple consequence of Theorem 1. If $\mathcal{F}$ is not simple, then by Theorem 1, it is passive-complete. A passive-complete functionality is not passive-trivial (as otherwise, all functionalities will be passive-trivial, which is not the case). □

An interesting special case of this appeared in [Kil00]: for an asymmetric deterministic SFE, its kernel is simply a constant functionality and is passive-trivial. Hence an asymmetric SFE is passive-trivial if and only if it is simple. That is, any asymmetric SFE is either passive-trivial or is complete.

5.2 Standalone Trivial SFE Functionalities

Theorem 4 *A finite 2-party SFE functionality is UC- or standalone-trivial in the information theoretic setting if and only if it has a simple core whose kernel (which is a simple SSFE functionality) is respectively UC or standalone-trivial.*

Proof: We give the proof for standalone-triviality; the argument for UC-triviality is similar.

Suppose a finite 2-party SFE functionality $\mathcal{F}$ has a simple core $\mathcal{F}'$ whose kernel $\mathcal{K}$ is standalone-trivial. Since, $\mathcal{F}'$ is simple, i.e. strongly isomorphic to $\mathcal{K}$, and $\mathcal{K}$ is standalone trivial, we conclude that $\mathcal{F}'$ is also standalone trivial. Since $\mathcal{F}$ is weakly-isomorphic to $\mathcal{F}'$ and weak isomorphism preserves standalone triviality, $\mathcal{F}$ itself is standalone trivial.

To see the converse, suppose $\mathcal{F}$ is a standalone-trivial SFE. Let $\mathcal{F}'$ be a core of $\mathcal{F}$. Standalone triviality of $\mathcal{F}$ implies that $\mathcal{F}'$ is also standalone trivial. Now, Lemma 2 implies that $\mathcal{F}'$ is also passive trivial and, in particular, it is not passive complete. By Theorem 1, $\mathcal{F}'$ is simple. Now, the core $\mathcal{K}$ of $\mathcal{F}'$ is standalone trivial because $\mathcal{F}$ is weakly isomorphic to $\mathcal{F}'$ and $\mathcal{F}'$ is strongly isomorphic to $\mathcal{K}$.

Note that if *any* core of $\mathcal{F}$ is standalone trivial, then so are all cores. Because $\mathcal{F}$ is weakly isomorphic to both cores and standalone triviality of one of them shall entail standalone triviality of the other core. □

References

[Bea89] Donald Beaver. Perfect privacy for two-party protocols. In Joan Feigenbaum and Michael Merritt, editors, *Proceedings of DIMACS Workshop on Distributed Computing and Cryptography*, volume 2, pages 65–77. American Mathematical Society, 1989. 2

[BGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In Janos Simon, editor, *STOC*, pages 1–10. ACM, 1988. 2

[CCD88] David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols. In Janos Simon, editor, *STOC*, pages 11–19. ACM, 1988. 2

[CK88] Claude Crépeau and Joe Kilian. Achieving oblivious transfer using weakened security assumptions (extended abstract). In *FOCS*, pages 42–52. IEEE, 1988. 2, 11

[CKL03] Ran Canetti, Eyal Kushilevitz, and Yehuda Lindell. On the limitations of universally composable two-party computation without set-up assumptions. In Eli Biham, editor, *EUROCRYPT*, volume 2656 of *Lecture Notes in Computer Science*. Springer, 2003. 2, 12

[CMW04] Claude Crépeau, Kirill Morozov, and Stefan Wolf. Efficient unconditional oblivious transfer from almost any noisy channel. In Carlo Blundo and Stelvio Cimato, editors, *SCN*, volume 3352 of *Lecture Notes in Computer Science*, pages 47–59. Springer, 2004. 2, 11, 12

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play ANY mental game. In Alfred V. Aho, editor, *STOC*, pages 218–229. ACM, 1987. See [**?**, Chap. 7] for more details. 1, 2, 3, 9

[GV87] Oded Goldreich and Ronen Vainish. How to solve any protocol problem - an efficiency improvement. In Carl Pomerance, editor, *CRYPTO*, volume 293 of *Lecture Notes in Computer Science*, pages 73–86. Springer, 1987. 1, 2, 9

[HM86] Stuart Haber and Silvio Micali. Unpublished manuscript, 1986. 1, 2, 9

[IPS08] Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Founding cryptography on oblivious transfer - efficiently. In David Wagner, editor, *CRYPTO*, volume 5157 of *Lecture Notes in Computer Science*, pages 572–591. Springer, 2008. 2, 9

[Kil88] Joe Kilian. Founding cryptography on oblivious transfer. In Janos Simon, editor, *STOC*, pages 20–31. ACM, 1988. 1, 2, 9

[Kil91] Joe Kilian. A general completeness theorem for two-party games. In Cris Koutsougeras and Jeffrey Scott Vitter, editors, *STOC*, pages 553–560. ACM, 1991. 2, 7, 9

[Kil00] Joe Kilian. More general completeness theorems for secure two-party computation. In F. Frances Yao and Eugene M. Luks, editors, *STOC*, pages 316–324. ACM, 2000. 2, 7, 10, 11, 12, 13

[KM11] Daniel Kraschewski and Jörn Müller-Quade. Completeness theorems with constructive proofs for finite deterministic 2-party functions. In Yuval Ishai, editor, *TCC*, volume 6597 of *Lecture Notes in Computer Science*, pages 364–381. Springer, 2011. 2, 7, 8, 10, 11, 12

[KMR09] Robin Künzler, Jörn Müller-Quade, and Dominik Raub. Secure computability of functions in the IT setting with dishonest majority and applications to long-term security. In Omer Reingold, editor, *TCC*, volume 5444 of *Lecture Notes in Computer Science*, pages 238–255. Springer, 2009. 1, 2, 5

[KTRR03] Prashant Koulgi, Ertem Tuncel, Shankar L. Regunathan, and Kenneth Rose. On zero-error coding of correlated sources. *IEEE Transactions on Information Theory*, 49(11):2856–2873, 2003. 4

[Kus89] Eyal Kushilevitz. Privacy and communication complexity. In *FOCS*, pages 416–421. IEEE, 1989. 1, 2

[MOPR11] Hemanta K. Maji, Pichayoot Ouppaphan, Manoj Prabhakaran, and Mike Rosulek. Exploring the limits of common coins using frontier analysis of protocols. In Yuval Ishai, editor, *TCC*, volume 6597 of *Lecture Notes in Computer Science*, pages 486–503. Springer, 2011. 2, 4, 5, 10

[MPR09] Hemanta K. Maji, Manoj Prabhakaran, and Mike Rosulek. Complexity of multi-party computation problems: The case of 2-party symmetric secure function evaluation. In Omer Reingold, editor, *TCC*, volume 5444 of *Lecture Notes in Computer Science*, pages 256–273. Springer, 2009. 1, 2, 5

[PR08] Manoj Prabhakaran and Mike Rosulek. Cryptographic complexity of multi-party computation problems: Classifications and separations. In David Wagner, editor, *CRYPTO*, volume 5157 of *Lecture Notes in Computer Science*, pages 262–279. Springer, 2008. 2, 9

[Wit76] Hans S. Witsenhausen. The zero-error side information problem and chromatic numbers (corresp.). *IEEE Transactions on Information Theory*, 22(5):592–593, 1976. 4

[WW06] Stefan Wolf and Jürg Wullschleger. Oblivious transfer is symmetric. In Serge Vaudenay, editor, *EUROCRYPT*, volume 4004 of *Lecture Notes in Computer Science*, pages 222–232. Springer, 2006. 4

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets. In *FOCS*, pages 162–167. IEEE Computer Society, 1986. 2

A Proof of Lemma 1

Proof: We shall show the following implications: (1) $\Rightarrow$ (3) $\Rightarrow$ (2) $\Rightarrow$ (1). In fact, first we shall show (2) $\Leftrightarrow$ (3) because this result gives a local test to check whether a graph is product distribution graph or not, which could be of independent interest.

Proof of (2) $\Rightarrow$ (3): Let $\mathrm{G}(\mathcal{F})$ be the graph of functionality $\mathcal{F}$ and $\{U_1, \ldots, U_n\}$ and $\{V_1, \ldots, V_n\}$ be the partition of the left and right partite sets of the connected components. The only interesting case is when there exists a $k \in [n]$ such that $u_0, u_1 \in U_k$ and $v_0, v_1 \in V_k$; otherwise $\mathsf{wt}(u_0, v_0)\mathsf{wt}(u_1, v_1) = 0 = \mathsf{wt}(u_0, v_1)\mathsf{wt}(u_1, v_0)$ and condition 3 holds trivially. Let p_u and q_v be the distributions over U

and V respectively and c_k be the distribution over the connected components of $\mathrm{G}(\mathcal{F})$. Now, we have:

$$\begin{aligned}\mathsf{wt}(u_0, v_0)\mathsf{wt}(u_1, v_1) &= p_{u_0} q_{v_0} \times p_{u_1} q_{v_1} \ / \ c_k^2 \\ &= p_{u_0} q_{v_1} \times p_{u_1} q_{v_0} \ / \ c_k^2 \\ &= \mathsf{wt}(u_0, v_1)\mathsf{wt}(u_1, v_0)\end{aligned}$$

Proof of (3) $\Rightarrow$ (2): Let U_k and V_k be the partite sets of the k-th component of the graph. Let W be the sum of weights on all edges of the graph and W_k be the sum of weights of edges in the k-th component. We define c_k as the distribution over the partitions such that probability of $k \in [n]$ is W_k/W. We represent the weight of the edge between i and j node as $\mathsf{wt}(i, j)$. The probability distribution over the edges is $\mathsf{wt}^*(i, j) = \mathsf{wt}(i, j)/W$. We define the weight of the node $i \in U$ as $p_i = \sum_{j' \in V} \mathsf{wt}(i, j')/W$. It is easy to observe that p_i is a probability distribution over U. Similarly, define the probability of $j \in V$ as $q_j = \sum_{i' \in U} \mathsf{wt}(i', j)/W$. Now, consider $i \in U_k$ and $j \in V_k$ and evaluate the expression $p_i \times q_j / c_k$:

$$\begin{aligned}p_i \times q_j / c_k &= \left(\sum_{j' \in V} \mathsf{wt}(i, j')\right) \times \left(\sum_{i' \in U} \mathsf{wt}(i', j)\right) \times \frac{1}{W_k W} \\ &= \left(\sum_{j' \in V_k} \mathsf{wt}(i, j')\right) \times \left(\sum_{i' \in U_k} \mathsf{wt}(i', j)\right) \cdot \frac{1}{W_k W} \\ &= \sum_{(i', j') \in U_k \times V_k} \mathsf{wt}(i, j')\mathsf{wt}(i', j) \frac{1}{W_k W} \\ &= \sum_{(i', j') \in U_k \times V_k} \mathsf{wt}(i, j)\mathsf{wt}(i', j') \frac{1}{W_k W} \\ &= (\mathsf{wt}(i, j)/W) \times (W_k/W_k) = \mathsf{wt}^*(i, j)\end{aligned}$$

Finally, we show the equivalence of $\mathcal{F}$ being simple with the other two statements. In the following, we shall represent the kernel of $\mathcal{F}$ as $\mathcal{K}$.

Proof of (1) $\Rightarrow$ (3): Suppose we are given a local protocol $\pi_{\mathcal{F}}$ which securely realizes $\mathcal{F}$ in the $\mathcal{K}$ hybrid. By definition, parties invoke $\mathcal{K}$ with the same input as their input for $\mathcal{F}$; and this protocol is passive, standalone and UC secure. Consider the experiment where $x \xleftarrow{\$} X$ and $y \xleftarrow{\$} Y$. We shall condition our analysis on the output of $\mathcal{K}$ being k in the real protocol, when inputs to Alice and Bob are x and y. The probability that Alice outputs a is denoted by $\tilde{p}_k(x, a)$, since $\pi_{\mathcal{F}}$ is a local protocol. Similarly, the probability that Bob outputs b is represented by $\tilde{q}_k(y, b)$.

Let c_k be the probability that $\mathcal{K}$ gives k as output to both parties when $x \xleftarrow{\$} X$ and $y \xleftarrow{\$} Y$. Note that c_k is the sum of weights on edges in the k-th component in $\mathrm{G}(\mathcal{F})$. The probability of edge joining $u = (x, a)$ and $v = (y, b)$ in the real world execution is $\tilde{\mathsf{wt}}(u, v) = c_k \times \tilde{p}_k(u)\tilde{q}_k(v)$, for every $u \in U_k$ and $v \in V_k$.

By security of the protocol, we can claim that:[5]

$$\begin{aligned} & \left|\tilde{\mathsf{wt}}(u,v) \;-\; \mathsf{wt}(u,v)\right| \leq \mathrm{negl}(\kappa) \\ \Leftrightarrow \quad & |c_k \tilde{p}_k(u)\tilde{q}_k(v) \;-\; \mathsf{wt}(u,v)| \leq \mathrm{negl}(\kappa) \end{aligned}$$

Consider drawing an edge from $\mathrm{G}(\mathcal{F})$ with probability equal to the weight on the edge. Let $p_k(u)$ be the probability that Alice's node is $u = (x, a)$, conditioned on the event that an edge in the k^{th} component is selected. Formally, $p_k(u) = \sum_{v' \in V_k} \mathsf{wt}(u, v')/c_k$. Similarly, we define $q_k(v) = \sum_{u' \in U_k} \mathsf{wt}(u', v)/c_k$. By security of the protocol, we can claim that $|\tilde{p}_k(u), p_k(u)| \leq \mathrm{negl}(\kappa)$ and $|\tilde{q}_k(v), q_k(v)| \leq \mathrm{negl}(\kappa)$. Here we use union bounds over V_k and U_k respectively.

Thus, we can conclude that

$$|c_k p_k(u) q_k(v) \;-\; \mathsf{wt}(u,v)| \leq \mathrm{negl}(\kappa)$$

Note that the function $\mathsf{wt}(\cdot,\cdot)$ assigns values which are integral multiples of $1/|X \times Y \times R|$. Therefore, c_k, $p_k(u)$ and $q_k(v)$ are also integral multiples of $1/|X \times Y \times R|$. So, if $c_k p_k(u) q_k(v)$ is not equal to $\mathsf{wt}(u, v)$, then

$$|c_k p_k(u) q_k(v) \;-\; \mathsf{wt}(u,v)| \geq \frac{1}{|X \times Y \times R|} \text{ which is non-negligible}$$

Thus, we can conclude that $\mathsf{wt}(u, v) = c_k p_k(u) q_k(v)$ and this trivially satisfies condition 3 (alternately, interpret $p_k(u)c_k$ as the distribution over U and $q_k(v)c_k$ as the distribution over V).

Proof of (2) $\Rightarrow$ (1): Let the distribution over U and V be p_u and q_v respectively; and the distribution over the connected components be c_k. From the product distribution guarantee we have $\mathsf{wt}(u, v) = p_u \cdot q_v / c_k$.

1. Computing $\mathcal{F}$ in $\mathcal{K}$ hybrid (Protocol $\pi_{\mathcal{F}}$): We provide the algorithm for Alice; and Bob's algorithm is symmetrically defined. On input x, Alice sends x to $\mathcal{K}$ setup and receives the connected component k as output. Given x and k there is distribution over her output $d_{x,k}(a)$ as induced by the edges in the k-th component of $\mathrm{G}(\mathcal{F})$. She locally samples her outcome according to this distribution. Formally, the probability of her output being a is:

$$\frac{p_{(x,a)}}{\sum_{(x,a') \in U_k} p_{(x,a')}}$$

2. Computing $\mathcal{K}$ in $\mathcal{F}$ hybrid (Protocol $\pi_{\mathcal{K}}$): Again, we provide Alice's algorithm. On input x, Alice sends x to $\mathcal{F}$ setup and receives her output a. Given (x, a) there is a unique k such that the node (x, a) lies in the k-th component of $\mathrm{G}(\mathcal{F})$. Alice outputs k.

Below we prove the security of these protocols. For simplicity, we shall assume that in the first protocol (and in the simulation for the second) it is possible to sample an outcome *exactly* according to a requisite distribution. In general this is not true (for instance, when the probabilities involved

[5] Since $\mathcal{F}$ is a finite functionality, the probability of an edge in $\mathrm{G}(\mathcal{F})$, when evaluated with random input, is at least $1/|X \times Y \times R|$. Thus, significant fraction of the soundness error in a particular edge propagates as soundness error in the overall experiment where $x \xleftarrow{\$} X$ and $y \xleftarrow{\$} Y$.

have binary infinite binary expansions). But this assumption can be removed by carrying out the sampling to within an exponentially small error (using polynomially many coins); this affects the security error only by an exponentially small amount.

Proofs for Protocol $\pi_{\mathcal{F}}$: Let us argue the correctness of the protocol. Define $W_{x,y}(k)$ as the weight of edges in k-th component of $\mathrm{G}(\mathcal{F})$ when Alice and Bob inputs are x and y respectively; and $W_{x,y} = \sum_{k\in[n]} W_{x,y}(k)$. Consider the event that the edge connecting (x,a) and (y,b) in $\mathrm{G}(\mathcal{F})$ lies in the k-th connected component. In both real and ideal worlds the probability of this event is $W_{x,y}(k)/W_{x,y}$. Now, we shall analyze the probability of joint distribution of (x,a) and (y,b) conditioned on this event. Let $U_{x,k}$ be the subset of the k-th connected component's left partite set which have Alice input x. Similarly, define $V_{y,k}$. The probability of the edge connecting $u = (x,a)$ and $v = (y,b)$ in the ideal world is:

$$\frac{\mathsf{wt}(u,v)}{W_{x,y}(k)}$$

The probability of the edge connecting u and v in the real world is:

$$\frac{p_u}{\sum_{u'\in U_{x,k}} p_{u'}} \times \frac{q_v}{\sum_{v'\in V_{y,k}} q_{v'}} = \frac{\mathsf{wt}(u,v)}{W_{x,y}(k)}$$

This shows that the protocol $\pi_{\mathcal{F}}$ is perfectly correct.

For security, we shall construct a simulator for Alice. Malicious Bob's case is analogous. When malicious Alice is invoked with inputs x, she sends $\tilde{x}$ to $\mathcal{K}$. The simulator, who is implementing the setup $\mathcal{K}$, forwards $\tilde{x}$ to the external $\mathcal{F}$ functionality. It receives an outcome $\tilde{a}$ from the external functionality. Next, the simulator sends the connected component in $\mathrm{G}(\mathcal{F})$ which contains the vertex $(\tilde{x},\tilde{a})$. Simulation is perfect because the probability of malicious Alice seeing $\tilde{k}$ in real and ideal work is exactly $W_{\tilde{x},y}(\tilde{k})/W_{\tilde{x},y}$.

Proofs for Protocol $\pi_{\mathcal{K}}$: The correctness of the protocol is trivial. Both in the real and ideal world, the probability of k being the output when Alice and Bob have inputs x and y respectively is $W_{x,y}(k)/W_{x,y}$.

For security, we shall construct a simulator for malicious Alice. When malicious Alice is invoked with inputs x, she sends $\tilde{x}$ to $\mathcal{F}$. The simulator, who is implementing the setup $\mathcal{F}$, forwards $\tilde{x}$ to the external $\mathcal{K}$ functionality and receives the connected component $\tilde{k}$ from the external functionality. It samples a node $\tilde{u} = (\tilde{x},\tilde{a})$ from $U_{\tilde{x},\tilde{k}}$ according to the distribution $p_{\tilde{u}}$ and sends $\tilde{a}$ as the output of $\mathcal{F}$. The simulation is perfect (up to sampling *exactly* with the requisite probabilities) because the probability of malicious Alice seeing $\tilde{a}$ in the real world is:

$$\frac{\sum_{v'\in V_{y,\tilde{k}}} \mathsf{wt}(\tilde{u},v')}{W_{\tilde{x},y}}$$

While the probability of the same event in the simulation is:

$$\begin{aligned}\frac{W_{\tilde{x},y}(\tilde{k})}{W_{\tilde{x},y}} \times \frac{p_{\tilde{u}}}{\sum_{u'\in U_{\tilde{x},\tilde{k}}} p_{u'}} &= \frac{W_{\tilde{x},y}(\tilde{k})}{W_{\tilde{x},y}} \times \frac{c_k \sum_{v'\in V_{y,\tilde{k}}} \mathsf{wt}(\tilde{u},v')}{c_k \sum_{(u',v')\in U_{\tilde{x},\tilde{k}}\times V_{y,\tilde{k}}} \mathsf{wt}(u',v')} \\ &= \frac{W_{\tilde{x},y}(\tilde{k})}{W_{\tilde{x},y}} \times \frac{\sum_{v'\in V_{y,\tilde{k}}} \mathsf{wt}(u,v')}{W_{\tilde{x},y}(\tilde{k})}\end{aligned}$$

This is identical to the previous expression. □

B Special Cases of Simple SFE

For the special case (possibly randomzied) SSFE functionalities, note that each connected component in the graph of an SSFE functionality has the same output value z. So we observe that Kilian's condition that $\exists x_0, x_1, y_0, y_1, z$ such that

$$\Pr[f(x_0, y_0) = z] > 0; \text{ and } \Pr[f(x_0, y_1) = z] > 0; \text{ and}$$
$$\Pr[f(x_0, y_0) = z] \cdot \Pr[f(x_1, y_1) = z] \neq \Pr[f(x_1, y_0) = z] \cdot \Pr[f(x_0, y_1) = z].$$

can be rephrased as follows: *there exists some connected component in the graph of the functionality that is not a product distribution*, or equivalently, *the functionality is not simple*. In terms of the above values x_0, x_1, y_0, y_1, z, this component (with output z) has nodes (x_0, z), (y_0, z), (x_0, z), (x_1, z). They are connected because the edges $((x_0, z), (y_0, z))$ and $((x_0, z), (y_1, z))$ are present, and (x_1, z) is connected with them either by the edge $((x_1, z), (y_0, z))$ or by the edge $((x_1, z), (y_1, z))$ (i.e., it is not the case that $\Pr[f(x_1, y_1) = z] = \Pr[f(x_1, y_0) = z] = 0$). This connected component is not a product distribution, because if it were, then $\Pr[f(x_0, y_0) = z] \Pr[f(x_1, y_1) = z] = p_A(x_0, z)p_A(x_1, z)p_B(y_0, z)p_B(y_1, z) = \Pr[f(x_1, y_0) = z] \Pr[f(x_0, y_1) = z]$, for some functions p_A and p_B.

To see the simplification in the case of deterministic SFE, note that in the graph of a deterministic SFE, a connected component must be a complete bipartite graph to have a product distribution. So, to *not* be a product graph, there must be two distinct nodes (x, a), (x', a') on the left and two nodes (y, b), (y', b') on the right such that there are edges $((x, a), (y, b))$, $((x, a), (y', b'))$, $((x', a'), (y, b))$, but the edge $((x', a'), (y', b'))$ is not present. That is, there are inputs $x \neq x'$ for Alice and $y \neq y'$ for Bob[6] such that $f_A(x, y) = f_A(x, y')$, $f_B(x, y) = f_B(x', y)$ and $(f_A(x', y'), f_B(x', y')) \neq (f_A(x', y), f_B(x, y'))$ (i.e., either $f_A(x', y) \neq f_A(x', y')$ or $f_B(x, y') \neq f_B(x', y')$ or both). That is the tuple (x, x', y, y') is an OT-core.

[6] If $x = x'$, then $a = f_A(x, y) = f_A(x', y) = a'$ and (x, a) and (x', a') are not distinct; similarly if $y = y'$ then $b = b'$.