Must You Know the Code of f to Securely Compute f?

Mike Rosulek*

Department of Computer Science, University of Montana
mikero@cs.umt.edu

Abstract. When Alice and Bob want to securely evaluate a function of their shared inputs, they typically first express the function as a (boolean or arithmetic) circuit and then securely evaluate that circuit, gate-by-gate. In other words, a secure protocol for evaluating f is typically obtained in a *non-black-box-way* from f itself. Consequently, secure computation protocols have high overhead (in communication & computation) that is directly linked to the circuit-description complexity of f.

In other settings throughout cryptography, black-box constructions invariably lead to better practical efficiency than comparable non-black-box constructions. Could secure computation protocols similarly be made more practical by eliminating their dependence on a circuit representation of the target function? Or, in other words, *must one know the code of f to securely evaluate f?*

In this work we initiate the theoretical study of this question. We show the following:

1. A complete characterization of the 2-party tasks which admit such security against semi-honest adversaries. The characterization is inspired by notions of *autoreducibility* from computational complexity theory. From this characterization, we show a class of pseudorandom functions that *cannot* be securely evaluated (when one party holds the seed and the other holds the input) without "knowing" the code of the function in question. On the positive side, we show a class of functions (related to blind signatures) that can indeed be securely computed without "knowing" the code of the function.
2. Sufficient conditions for such security against malicious adversaries, also based on autoreducibility. We show that it is not possible to prove membership in the image of a one-way function in zero-knowledge, without "knowing" the code of the one-way function. We also describe a variant of the GMW compiler for transforming semi-honest to malicious security while preserving the specific black-box property considered here.

1 Introduction

In cryptography, a **black-box** construction is one that uses only the input/output behavior of its components [21,28]. By contrast, a non-black-box construction

* Supported by NSF grant CCF-1149647.

R. Safavi-Naini and R. Canetti (Eds.): CRYPTO 2012, LNCS 7417, pp. 87–104, 2012.

relies on the *code* of its components. Understanding exactly when non-black-box techniques are necessary is important for cryptography, since black-box constructions are typically much more efficient (in their computation and/or communication) than comparable non-black-box constructions.

Secure multi-party computation (MPC) allows mutually distrusting parties to securely evaluate a function f on their shared inputs. This powerful paradigm is well-known in the theoretical community but appears to be seldom used in practice. As a result, much current work focuses on improving the efficiency of MPC constructions to facilitate more widespread use. A recent line of work (see [20] and followup works [22,17,26]) has focused on improving efficiency by removing certain non-black-box techniques used in all earlier work. In particular, these results focus on the black-box use of the underlying *cryptographic primitives* (that is, one-way functions, trapdoor permutations, or standalone-secure oblivious transfer) used in the protocol.

One goal in this paper is to make explicit another non-black-box step inherent to all existing general-purpose MPC protocols. To build a secure protocol for a function f, the function must first be expressed as a low-level circuit ([30,19] use boolean circuits, [16,9] use arithmetic circuits, and [23] uses branching programs). Then, the protocol proceeds to securely evaluate the circuit, gate by gate. In other words, a secure protocol for f is *non-black-box in its usage of f itself.* While this framework provides a straight-forward way to achieve complete generality, it also inherently ties the efficiency (communication, computation, or both) of the protocol to the circuit-representation complexity of f. For this reason, an important line of research has streamlined many aspects of this non-black-box dependence, including techniques for optimizing circuits for MPC [24,27] and exploring alternative circuit representations [3].

It is unreasonable to expect that we can avoid this non-black-box step for *general-purpose* MPC (indeed, our results explicitly confirm this). Still, it is important to understand exactly to what extent the non-black-box dependence is inherent. For which *special-purpose* secure computation tasks can we avoid dependence on the code of the target function altogether (and hopefully construct highly efficient protocols)?

1.1 Overview of the Results

We initiate the theoretical study of when a protocol can securely compute f without "knowing" the code of f. When considering a (standard) secure protocol for a functionality f, that choice of f is fixed and the protocol is allowed to depend arbitrarily on f. In this case, it is not meaningful to place any syntactic restrictions on the protocol (*e.g.*, that it use oracle access to a subroutine implementing f), since the protocol could have a circuit for f hard-coded anyway.

Instead, we model a protocol that "does not know" the code of its target functionality in the following way. The protocol is a pair of oracle machines that, when instantiated with any f from some larger *class* of functionalities, securely emulates a functionality related to f. By considering large classes of

functionalities, we prevent the protocol from hard-coding relevant circuit representations of the functionalities.

Definition (Informal). *Let $\mathcal{F}$ be an ideal (2-party) functionality implemented as an oracle machine, and let $\mathcal{C}$ be a class of functions. Then a* **functionally-black-box (FBB)** *protocol for $\mathcal{F}^{\mathcal{C}}$ is a pair of oracle machines (π_A, π_B) such that, for all $f \in \mathcal{C}$, the instantiated protocol (π_A^f, π_B^f) is a secure protocol for $\mathcal{F}^f$.*

As a natural example, $\mathcal{C}$ can be a class of functions that take two inputs, and $\mathcal{F}$ can be the simple functionality which collects x from Alice, y from Bob, queries its oracle to obtain $z = f(x, y)$, and gives the result to both parties (secure function evaluation). Or, $\mathcal{F}$ can be the functionality which collects (x, w) from Alice, and then gives x to Bob if $f(x, w) = 1$ (zero-knowledge proof).

We point out that it is only the *protocol* which must treat f as a black-box. In particular, the order of quantifiers is such that adversaries and simulators attacking the f-instantiated protocol may depend arbitrarily on the choice of f (and hence could be said to "know" the code of f).

We put forth the FBB property as a *necessary* condition for highly efficient, practical MPC protocols. This work therefore focuses on a theoretical understanding of the FBB property, as a *proxy* for practical efficiency. However, FBB alone is not a sufficient condition for practical protocols. Indeed, the protocols that we construct in this work may not be considered "practical."

Autoreducibility Characterization for 2-party Passive Security. In computational complexity theory, the notion of *autoreducibility* is a way to measure the "structure" within a set. Very generally, a set A is autoreducible if there exists an oracle machine M such that $M^A(x) = A(x)$, and yet M's oracle queries do not "depend too much" on x. An instance of autoreducibility which shows up frequently in cryptography is the notion of *random self-reducibility.* In that case, the oracle queries of M are distributed independently of the input x.

We define a variant of autoreducibility called **2-hiding autoreducibility**, which subsumes random self-reducibility and has some similarities to "2-oracle instance-hiding" autoreducibility defined by Beaver & Feigenbaum [5]. Intuitively, 2-hiding autoreducibility requires a single oracle machine M such that $M^f(x, y) = f(x, y)$ for every f in a large class $\mathcal{C}$; furthermore, half of M's oracle queries are "independent" of x and the other half are "independent" of y, in some sense. We then show that 2-hiding autoreducibility completely characterizes FBB feasibility (for 2-party deterministic secure function evaluation):

Theorem (Informal). *Let $\mathcal{C}$ be a class of 2-input functions. Then functionally-black-box secure evaluation of $\mathcal{C}$ is possible against semi-honest adversaries, in the presence of an arbitrary trusted setup, if and only if $\mathcal{C}$ is 2-hiding autoreducible.*

We also emphasize that *achieving the FBB property is not simply a matter of providing a very powerful trusted setup* to the parties. Indeed, to be meaningful, the trusted setup must be the same for every $f \in \mathcal{C}$. Since it is only the parties

and not the trusted setup that have access to f, it is not immediate that even a powerful setup can be useful. Subsequently, our characterization can be used to give impossibility results that hold even in the presence of an arbitrary setup. Without loss of generality, one can assume the presence of a *complete* trusted setup such as the oblivious transfer functionality [22], or a common reference string [14].

Non-trivial FBB Feasibility. There is a trivial sense in which some classes of functionalities admit FBB protocols. We say (informally) that a class $\mathcal{C}$ is *learnable* if it is possible to efficiently obtain a circuit for f using only oracle access for f, for every $f \in \mathcal{C}$. Then every learnable class admits an FBB protocol of the following form: the parties independently query f to obtain a (canonical) circuit that evaluates f; they then use a standard general-purpose construction such as [19,14,22]. Thus, the notion of functionally-black-box is most meaningful when considering non-learnable classes of functionalities. (We note that a similar triviality occurs in the context of obfuscation [4], with respect to this same notion of learnability.) Informally, it is possible to "securely compute f without knowing the code of f" if f belongs to some class $\mathcal{C}$ that admits FBB secure protocols but is not learnable from oracle queries.

Using our autoreducibility characterization, we explicitly show that non-trivial FBB protocols are possible. That is, learnability is a strictly stronger condition than FBB feasibility. Intuitively, learnability requires the *entire* function to be deduced from oracle access, while our autoreducibility characterization only requires M to deduce the correct answer on a single, given input. We show a class of functions (related to blind signatures) that admits FBB protocols, but is not explicitly learnable from oracle queries. Thus, in some cases it is in fact possible to securely compute f "without knowing" the code of f.

Infeasibility of PRF Evaluation. As another demonstration of our autoreducibility characterization, we show that it is impossible to securely evaluate arbitrary PRFs (where one party holds the seed and the other holds the PRF input)[1] in an FBB manner. Impossibility holds even if arbitrary trusted setups are available, and even if security is only required against semi-honest adversaries. The result also easily extends to the class of (strong) PRPs. We leave open the question of whether a natural *subclass* of PRFs admits FBB-secure protocols (in a non-trivial way).

Sufficient Conditions for Malicious Security. We define another variant of autoreducibility, called 1-hiding autoreducibility. The definition is similar to that of 2-hiding autoreducibility, except that in 1-hiding autoreducibility all of the oracle machine's queries are independent of both x and y. We then show that 1-hiding autoreducible is a sufficient condition for FBB feasibility against *malicious* adversaries.

[1] Evaluating the AES block cipher in this way is now a relatively standard performance benchmark for MPC protocols [25].

We also show a variant of the GMW compiler to convert semi-honest-secure to malicious-secure protocols, in a way that preserves the FBB property. Just as the GMW compiler uses zero-knowledge proofs as its main component, we require an FBB protocol for the simple functionality that takes input x from Alice and gives $f(x)$ to Bob (FBB with respect to $f \in \mathcal{C}$). This functionality can be considered a zero-knowledge proof of membership in the image of f.

Zero-Knowledge. Beyond the GMW-inspired application described above, zero-knowledge proofs are interesting in their own right. ZK proofs are often the sole source of non-black-box behavior (and thus the efficiency bottleneck) in a protocol.

Let f be a deterministic function and define the relation $R_f(x, w) = 1 \Leftrightarrow f(w) = x$. We show that FBB zero-knowledge proofs are impossible for the class of relations $\{R_f \mid f \text{ is a OWF}\}$. In fact, our impossibility result is much stronger, ruling out even honest-verifier witness-hiding (instead of zero-knowledge), arguments rather than proofs; impossibility further applies to basic standalone security, and holds in the presence of arbitrary trusted setups.

1.2 Other Related Work

The notion of autoreducibility has proven to be a fruitful tool in complexity theory for quantifying the amount of structure in a set. For a gentle introduction to research on this topic, see [29,2].

In cryptography, autoreducibility has already been recognized as a tool for several interesting applications. Abadi, Feigenbaum, & Kilian [1] used "instance hiding" autoreducibility to reason about what would in today's parlance be called a form of outsourced computation. Here, a client wishes to compute $f(x)$ using access to a powerful trusted server who can evaluate the function f; the client wants to learn $f(x)$ but does not want his queries to f to leak any information about x. The notion was later extended by Beaver & Feigenbaum [5] to a setting involving multiple (non-colluding) servers. A summary of this line of work was given by Brassard [12]. Beaver *et al.* [6] also used autoreducibility to construct efficient perfect zero-knowledge proofs.

The question of FBB protocols requires a fundamentally different style of autoreducibility than studied in previous works. First, existing definitions consider an oracle machine which is required to work only for a single fixed language (or function); whereas here we require a single oracle machine which works for any function from a large class. Second, we must explicitly consider functions on two inputs, and make a distinction between oracle queries that depend on each of the inputs. In addition, our notion of "query independence" is specific to our application of secure protocols. Finally, many previous definitions of autoreducibility allow the oracle machine to be instantiated with an oracle that is distinct from (but depends on) the required output function — *e.g.*, an oracle machine computes the function f when given oracle access to g, a low-degree encoding of f.

2 Preliminaries

A probability $p(n)$ is negligible if for all, $c > 0$, $p(n) < n^{-c}$ for all but finitely many n. We write $p(n) \approx q(n)$ if $|p(n) - q(n)|$ is negligible. A probability $p(n)$ is overwhelming if $p(n) \approx 1$.

When discussing security of MPC protocols, we use Canetti's framework of Universally Composable (UC) security [13]. The low-level details of the security model are not crucial to our results. We do, however, make one distinction about notation:

In the MPC literature, the notation π^f often refers to a protocol π where the parties can use a *shared* ideal functionality f (the f-hybrid model, in UC parlance). In this work, write π^f to instead denote a protocol machine equipped with (local) oracle access to *independent* instances of f. In that sense, our notation reflects the complexity-theoretic idea of an oracle computation.

Definition 1 (Related-Key security for PRFs [8]). *Let Φ be a class of functions over $\{0,1\}^k$. Then $F : \{0,1\}^k \times \{0,1\}^k \to \{0,1\}^k$ is a* **pseudorandom function secure under Φ-related-key attacks (Φ-RKA-PRF)** *if for all efficient M, $|\Pr[M^{\mathcal{O}}(1^k) = b] - \frac{1}{2}|$ is negligible in k, in the following game:*

Bit b is chosen at random and s is chosen uniformly from $\{0,1\}^k$. If $b = 0$ then queries of the form $\mathcal{O}(\phi, x)$, where $\phi \in \Phi$ and $x \in \{0,1\}^k$, are answered as $F(\phi(s), x)$. If $b = 1$ then the query $\mathcal{O}(\phi, x)$ is answered as $R_{\phi(s)}(x)$, where for each v, R_v is chosen as a random function from $\{0,1\}^k \to \{0,1\}^k$.

If $\otimes$ is a group operation on $\{0,1\}^k$, and $\Phi = \{\phi_\Delta \mid \Delta \in \{0,1\}^k\}$ where $\phi_\Delta(x) = x \otimes \Delta$, then we say that Φ is **group-induced.** Bellare & Cash [7] give constructions of Φ-RKA-PRFs (and PRPs) for group-induced Φ.

Definition 2 (Blind signatures [15]). *Let $\Sigma = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Ver})$ be a signature scheme. A* **blind signature protocol** *for Σ is one in which the client has input $(1^k, m)$, the signer has input $(1^k, sk)$, and the client receives output $\mathsf{Sign}(sk, m)$. The blindness condition is that the view of the signer is statistically independent of m.*

3 Functionally Black-Box Protocols

We now give the formal definition of MPC protocols that "do not know" the code of their target function f, as described in the introduction:

Definition 3. *Let $\mathcal{C}$ be a class of functions, and let $\mathcal{F}$ be an ideal functionality that is implemented as an (uninstantiated) oracle machine. Let π_A and π_B be interactive oracle machines. Then we say that $\pi = (\pi_A, \pi_B)$ is a* **functionally-black-box (FBB)** *protocol for $\mathcal{F}^{\mathcal{C}}$ in a certain security model if, for all $f \in \mathcal{C}$, the protocol (π_A^f, π_B^f) is a secure protocol (in the model in question) for the ideal functionality $\mathcal{F}^f$.*

Importantly, the definition defers to the security condition *separately* for each instantiation (π_A^f, π_B^f). Thus, adversaries and simulators attacking the f-instantiated protocol in the security definition can depend arbitrarily on the choice of $f \in \mathcal{C}$. This models the fact that adversaries are not restricted to use f as a black-box. Indeed, the intent is to characterize convenience/efficiency for the honest parties, without compromising any level of security.

Instantiations Used in This Work. Let $\mathcal{F}_{\text{SFE}}$ be the non-reactive functionality that takes input x from Alice and y from Bob, queries its oracle f to obtain $z = f(x, y)$, and gives the output to both parties. Then an FBB protocol for $\mathcal{F}_{\text{SFE}}^{\mathcal{C}}$ would allow the protocols to have only black-box access to the function f being evaluated.

As another example, let $\mathcal{F}_{\text{FZK}}$ be the functionality that takes input w from Alice, queries its oracle f to obtain $x = f(w)$, and gives output x to Bob. When instantiated with function f, $\mathcal{F}_{\text{FZK}}^f$ is essentially a zero-knowledge argument (of knowledge) functionality for statements of the form "$\exists w : x = f(w)$". Note that in this setting we allow parties to have access to the function f rather than the (more restricted) NP relation $R_f(x, w) = 1 \Leftrightarrow f(w) = x$.

Simple Observations. Suppose $\mathcal{C}$ is **learnable** in the following sense. There exists a PPT oracle TM M such that with overwhelming probability (in k), $M^f(1^k, 1^t)$ outputs a circuit that agrees with f on $\{0,1\}^t$, for all $f \in \mathcal{C}$. In the simple case where M will always output a *canonical* circuit, then an FBB protocol can be obtained by having both parties (independently) running M, and then using an appropriate non-FBB protocol construction on the resulting circuit. Even if M does not reliably output the same circuit each time, an FBB protocol can be obtained using the approach outlined later in Theorem 2 (the class is 1-hiding autoreducible via the oracle machine which runs $C \leftarrow M^f(1^k, 1^{|x|+|y|}); C(x, y)$).

Suppose $\mathcal{C}$ contains only functions whose input domains are **constant-sized** (*i.e.*, not growing in size with k). Then $\mathcal{C}$ is (canonically) learnable in the above sense, by exhaustively querying the function. For this reason, we only consider classes which contain functions over infinite domains.

4 Classification for Semi-honest Security

In this section we define a notion of autoreducibility, and then show that it completely characterizes feasibility of FBB MPC protocols against semi-honest adversaries.

Definition 4. *Let $\mathcal{C}$ denote a class of functions on two inputs. Let M be a PPT oracle machine, and suppose each oracle query is tagged with a label $i \in \{1, 2\}$ (say, by setting some internal variable at the time of the query). Then for $i \in \{1, 2\}$ define $\mathcal{Q}_i[M, f; z]$ to be the random variable containing the sequence of oracle queries made with label i during the computation $M^f(z)$. We say that $\mathcal{C}$ is* **2-hiding autoreducible** *if there exists a PPT oracle machine M such that for all $f \in \mathcal{C}$:*

1. *For all x, y, we have that $\Pr[M^f(1^k, x, y) = f(x, y)]$ is overwhelming in k.*
2. *There exists a PPT machine $\mathcal{S}_{f,1}$ such that for all x, y, the following ensembles are indistinguishable (in k):*

$$\{\mathcal{S}_{f,1}(1^k, f(x,y), x)\}_k \quad \textit{and} \quad \{\mathcal{Q}_1[M, f; 1^k, x, y]\}_k.$$

3. *There exists a PPT machine $\mathcal{S}_{f,2}$ such that for all x, y, the following ensembles are indistinguishable (in k):*

$$\{\mathcal{S}_{f,2}(1^k, f(x,y), y)\}_k \quad \textit{and} \quad \{\mathcal{Q}_2[M, f; 1^k, x, y]\}_k.$$

In other words, M is able to use oracle access to f to determine $f(x, y)$, yet its type-1 oracle queries to f are "independent" of y and its type-2 queries are "independent" of x, in some sense. A special case is when M's type-1 queries are distributed independently of y (and type-2 queries independently of x).

We now give our main classification for semi-honest security, which holds in the presence of arbitrary trusted setups. Without loss of generality, we can take the trusted setup to be the oblivious transfer functionality, which we denote by $\mathcal{F}_{\text{OT}}$.

Theorem 1. *There is a FBB protocol for $\mathcal{F}_{\text{SFE}}^{\mathcal{C}}$ secure against PPT semi-honest adversaries in the $\mathcal{F}_{\text{OT}}$-hybrid model if and only if $\mathcal{C}$ is 2-hiding autoreducible.*

Proof. ($\Leftarrow$) Suppose that the oracle machine M satisfies the definition of 2-hiding autoreducibility for $\mathcal{C}$. Without loss of generality, assume that the number of queries made by M depends only on the input 1^k (*i.e.*, it does not depend on x or y), and that M strictly alternates between type-1 and type-2 queries. We then define the ideal functionality $\mathcal{F}_M$ as in Figure 1.

There is a UC-secure protocol for $\mathcal{F}_M$ in the $\mathcal{F}_{\text{OT}}$-hybrid model, so it suffices to design a FBB MPC protocol for $\mathcal{F}_{\text{SFE}}^{\mathcal{C}}$ in the $\mathcal{F}_M$-hybrid model. (Note that the same $\mathcal{F}_M$ will be used in the protocol for each $f \in \mathcal{C}$.) The FBB protocol for $\mathcal{F}_{\text{SFE}}^{f}$, $f \in \mathcal{C}$, is relatively straight-forward. On inputs x for Alice and y for Bob, the parties send (INIT, x) and (INIT, y) to $\mathcal{F}_M$, respectively. Whenever a party receives output (QUERY, q) from $\mathcal{F}_M$, for $q \neq \perp$, that party uses its local oracle to compute $r = f(q)$, and gives (RESP, r) to $\mathcal{F}_M$. When the parties receive (OUT, z) from $\mathcal{F}_M$, they output z.

Clearly the protocol is FBB. To show that this protocol is secure against semi-honest adversaries, it suffices to show that the view of an honest party can be simulated in the ideal world. The simulator for a semi-honest Alice is as follows. Fix $f \in \mathcal{C}$ and recall that the simulator for the f-instantiated protocol can depend arbitrarily on f. The simulator receives Alice's input x from the environment. When Alice sends (INIT, x) to $\mathcal{F}_M$, the simulator sends x to the ideal functionality and receives $z = f(x, y)$. The simulator then computes $Q \leftarrow \mathcal{S}_{f,1}(1^k, z, x)$, where $\mathcal{S}_{f,1}$ is the machine guaranteed by the 2-hiding autoreducibility definition. For each query q in the sequence Q, the simulator gives (QUERY, q) and then $(\text{QUERY}, \perp)$ to Alice on behalf of $\mathcal{F}_M$. Finally, the simulator gives output (OUT, z) to Alice on behalf of $\mathcal{F}_M$. It is clear that Alice's view

The functionality maintains a configuration of the machine M in an internal variable S. This variable is manipulated by commands from Alice and Bob, as described below. After every such input, the functionality simulates M with configuration S. If M terminates with output z, give output (OUT, z) to both parties and halt. If M queries its oracle with a type-1 query q, then give output (QUERY, q) to Alice and $(\text{QUERY}, \perp)$ to Bob, and wait. If M queries its oracle with a type-2 query q, then give output $(\text{QUERY}, \perp)$ to Alice and (QUERY, q) to Bob, and wait.

1. On inputs (INIT, x) from Alice and (INIT, y) from Bob, initialize S as the initial configuration of oracle machine M on input $(1^k, x, y)$, where k is the global security parameter. Then simulate M as described above.
2. On input (RESP, r) from Alice, ensure that in configuration S, M is awaiting a reply to a type-1 oracle query (otherwise abort). Update S to reflect a response r on the oracle response tape, then simulate M as described above.
3. On input (RESP, r) from Bob, do the same as the previous step except ensure that M is awaiting a reply to a type-2 oracle query.

Fig. 1. Ideal functionality $\mathcal{F}_M$, parameterized by oracle PPT machine M

is computationally indistinguishable from that of the real interaction, by the guarantees of $\mathcal{S}_{f,1}$. The protocol is essentially symmetric with respect to Alice and Bob, and so security holds for a semi-honest Bob as well. Note that since we consider only semi-honest security, the adversary will indeed provide correct oracle responses to $\mathcal{F}_M$. Indeed, a malicious party could easily invalidate this security argument by providing incorrect oracle responses to $\mathcal{F}_M$.

($\Rightarrow$) Suppose $\pi = (\pi_A, \pi_B)$ is an FBB protocol for $\mathcal{F}_{\text{SFE}}^{\mathcal{C}}$ in the $\mathcal{F}_{\text{OT}}$-hybrid model. Define an oracle machine M as follows: It internally simulates instances of π_A, π_B, and $\mathcal{F}_{\text{OT}}$ as in their protocol interaction. On input $(1^k, x, y)$ to M, it gives input $(1^k, x)$ to π_A and input $(1^k, y)$ to π_B. It then simulates the three subcomponents in the natural way. Whenever π_A makes an oracle query, M makes the same query as a type-1 query and returns the result to the π_A component. Similarly, queries made by π_B are made as type-2 queries. The final output of π_A is taken to be the output of M.

By the correctness of the protocol π, we have that the output of M is equal to $f(x, y)$ with overwhelming probability for all x, y, when instantiated with f as its oracle. Now fix a particular $f \in \mathcal{C}$. The security of π instantiated with f implies the existence of a simulator $\mathcal{S}$ in the $\mathcal{F}_{\text{SFE}}^{f}$-ideal model. We define $\mathcal{S}_{f,1}$ to do the following, on input $(1^k, f(x, y), x)$: First, run $\mathcal{S}$ on input $(1^k, f(x, y))$ against a semi-honest corrupt Alice with input $(1^k, x)$. Then $\mathcal{S}$ generates a simulated view for Alice; output the sequence of simulated oracle queries contained in Alice's view. By the soundness of $\mathcal{S}$, the output of $\mathcal{S}_{f,1}$ is indistinguishable from the sequence of type-1 queries made by M, as required for 2-hiding autoreducibility. The required $\mathcal{S}_{f,2}$ algorithm is defined symmetrically.

Discussion. The protocol for realizing $\mathcal{F}_M$ in the $\mathcal{F}_{\text{OT}}$-hybrid model uses the oracle machine M in a highly non-black-box manner. So while the protocol is black-box in the code of $f \in \mathcal{C}$, it is not black-box in the code of M. However, we note that the code of M may be significantly simpler than that of $f \in \mathcal{C}$ (for example, when M's oracle queries are made uniformly), and also that M is fixed for the entire class $\mathcal{C}$.

4.1 A Positive Example, and Comparison to Learnability

Using our characterization, we can show that FBB feasibility is a *strictly weaker* condition than learnability (as defined in Section 3). In other words, it is indeed possible to securely evaluate certain classes of functions without "knowing" the code of the function.

Let $\Sigma = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Ver})$ be an existentially unforgeable signature scheme. Without loss of generality, we assume that the Sign algorithm is deterministic (it can be derandomized using a PRF). Let (π_C, π_S) denote a blind signature protocol for this scheme (Definition 2), where C is the client and S is the signer. We call a blind signature protocol **modular** if the π_S protocol does not use the signing key except via oracle access to $\mathsf{Sign}(sk, \cdot)$. That is, the signer executes the protocol as $\pi_S^{\mathsf{Sign}(sk,\cdot)}(1^k, vk)$. As a concrete example, the Boneh-Lynn-Shacham signature scheme [11] supports such a blind signature protocol [10].

For a signing key sk, define the function $S_{sk}(x, y) = \mathsf{Sign}(sk, x)$.

Lemma 1. *If Σ has a modular and semi-honest secure blind signature protocol, then the class $\mathcal{C}_\Sigma = \{S_{sk} \mid sk \in \{0,1\}^*\}$ is 2-hiding autoreducible but not learnable. (In fact, the class $\mathcal{C}_\Sigma$ is 1-hiding autoreducible; defined later in Section 5.)*

Proof. We construct a machine M for the definition of 2-hiding autoreducibility. On input $(1^k, x, y)$, the machine M simulates instances of $\pi_C(1^k, vk, x)$ and $\pi_S^{\mathsf{Sign}(sk,\cdot)}(1^k, vk)$ in the natural way. Whenever π_S queries its oracle on message m, M makes a type-1 query $(m, \perp)$ and uses the result as the response to π_S. Finally, M uses the final output of π_C as its own output.

By the correctness of the π protocol, M satisfies the desired correctness condition for 2-hiding autoreducibility. From the blindness property of π, it follows that the oracle queries made by π_S in the protocol are distributed independently of x. Hence, the entire set of queries made by M (all type-1) are distributed independently of (x, y).

The fact that $\mathcal{C}_\Sigma$ is not learnable follows from the existential unforgeability of Σ. Suppose a machine M, using oracle access to S_{sk} for a randomly chosen sk, is able to output a circuit correctly computing S_{sk}. Then the following is an attack in the unforgeability game for Σ: On input $(1^k, vk)$ run $M(1^k)$. Whenever M makes an oracle query (x, y), request a signature on x in the game and return the result to M. When M outputs a circuit C, choose any x^* such that no query

of the form $(x^*, \cdot)$ was ever made. Then run $C(x^*, \perp)$, which by assumption is a valid signature on x^*; hence, a forgery.[2]

4.2 A Negative Example: Infeasibility of PRFs

In this section, we treat pseudorandom functions as functions of two arguments: the first argument being the seed and the second argument being the PRF input. Thus, a functionality evaluating a PRF in our terminology corresponds to a functionality which takes a seed from Alice and an input from Bob, and evaluates the PRF accordingly.

We now show that FBB protocols are impossible for the class of all pseudorandom functions. While this claim is sensible at an intuitive level (pseudorandomness precludes significant structure like that required for 2-hiding autoreducibility), the proof has some subtlety. To apply the security of the PRF we must have its seed (secretly) chosen at random, whereas in the 2-hiding autoreducibility definition, the oracle machine is given Alice's input (taken to be the PRF seed) and can arbitrarily query the *unseeded* PRF $f(\cdot, \cdot)$ as an oracle. We show that, given a PRF secure against *related-key attacks*, we can "embed" an additional seed into the PRF oracle in a way that allows the PRF security to apply to the 2-hiding autoreducibility interaction.

Lemma 2. *Define $\mathcal{C}_{\text{PRF}} = \{f \mid f \text{ is a PRF}\}$. If Φ-RKA-secure PRFs exist for group-induced Φ (Definition 1), and injective PRGs exist, then $\mathcal{C}_{\text{PRF}}$ is not 2-hiding autoreducible, and thus there is no FBB protocol for $\mathcal{F}_{\text{SFE}}^{\mathcal{C}_{\text{PRF}}}$, even against semi-honest adversaries and in the $\mathcal{F}_{\text{OT}}$-hybrid model.*

Additionally, we point out that the proof goes through with minimal modification with respect to the class of pseudorandom *permutations* (as before, assuming the existence of *RKA-secure* PRPs). Regarding the condition in the lemma statement, Bellare & Cash [7] give constructions of suitable PRFs (and PRPs) under either the DDH or DLIN assumptions, and also assuming the existence of collision-resistant hash functions.

Proof. Let f be a PRF secure against group-induced RKA attacks. For concreteness and clarity, we write the allowed group operation as $\oplus$. Let $g : \{0,1\}^k \to \{0,1\}^{2k}$ be an injective PRG and define the following function for an arbitrary string s:

$$f_s(x, y) = f(s \oplus g(x), g(y)).$$

So that f_s is defined for inputs of arbitrary length, we assume that the fixed string s is padded with zeroes or truncated to the appropriate length ($|g(x)|$) in the expression $s \oplus g(x)$. Now we claim that for each (fixed) string $s \in \{0,1\}^*$,

[2] The same argument holds with a significantly weaker requirement on learnability — the circuit C need only agree with S_{sk} on some noticeable fraction of inputs, and this only with noticeable probability.

the function f_s is a PRF (interpreting x as its seed). Consider an efficient oracle machine A. We have:

$$\Pr_{x\leftarrow\{0,1\}^k}[A^{f(s\oplus g(x),\cdot)}(1^k)=1] \approx \Pr_{x'\leftarrow\{0,1\}^{2k}}[A^{f(s\oplus x',\cdot)}(1^k)=1]$$
$$\approx \Pr_{x'\leftarrow\{0,1\}^{2k}}[A^{f(x',\cdot)}(1^k)=1] \approx \Pr_{R}[A^{R}(1^k)=1].$$

In the above, R is a random oracle; thus f_s is a PRF. Indistinguishability holds due to the pseudorandomness of g, the fact that $\oplus$ is a group operation, and the pseudorandomness of f, respectively.

Suppose for the sake of contradiction that oracle machine M satisfies the condition required for 2-hiding autoreducibility of the class of pseudorandom functions. Then for *every* PRF h and all strings x and y, we have that $\Pr[M^h(1^k,x,y)=h(x,y)]$ is overwhelming in k. In particular, the same probability is overwhelming for random choice of $x,y\in\{0,1\}^k, s\in\{0,1\}^{2k}$, and setting $h=f_s$ (since *each* f_s is a PRF). In the RKA-PRF security game for f, the oracle machine is allowed to make queries of the form (ϕ,z) and obtain either $f(s\oplus\phi,z)$ for randomly chosen s, or $R_\phi(z)$, where each R_ϕ is an independent random function. Hence,

$$\Pr_{\substack{x,y\leftarrow\{0,1\}^k\\ s\leftarrow\{0,1\}^{2k}}}[M^{f_s}(1^k,x,y)=f_s(x,y)] \approx \Pr_{x,y\leftarrow\{0,1\}^k}[M^{R_{g(\cdot)}(\cdot)}(1^k,x,y)=R_{g(x)}(y)]$$
$$= \Pr_{x,y\leftarrow\{0,1\}^k}[M^{R}(1^k x,y)=R(x,y)].$$

Here, each R_ϕ, and finally R, is chosen as a random function. The last equality holds from the fact that g is injective.

From this we see that when the machine M is given inputs $x,y\in\{0,1\}^k$ chosen randomly, and an oracle f_s with $s\in\{0,1\}^{2k}$ chosen randomly, it must query the oracle on (x,y) with overwhelming probability. By an averaging argument, there must exist a negligible function δ and strings $\{s_k\}_{k\in\mathbb{N}}$, each $s_k\in\{0,1\}^{2k}$, such that:

$$\forall k: \Pr_{x,y\leftarrow\{0,1\}^k}[M^{f_{s_k}}(1^k,x,y) \text{ queries its oracle on } (x,y)] \geq 1-\delta(k).$$

When M queries its oracle on its given input, this query is either a type-1 or a type-2 query. Thus, there must exist additional $\{b_k\}_{k\in\mathbb{N}}$, each $b_k\in\{1,2\}$, satisfying:

$$\forall k: \Pr_{x,y\leftarrow\{0,1\}^k}[(x,y)\in\mathcal{Q}_{b_k}[M,f_{s_k};1^k,x,y]] \geq \frac{1}{2}-\delta(k).$$

Importantly, even for a fixed s_k, the function f_{s_k} is a PRF and thus in the class $\mathcal{C}$ for which the properties of M apply. So for each s_k, there is a corresponding simulator $\mathcal{S}_k$ that takes input 1^k, $f_{s_k}(x,y)$, and either x or y (depending on b_k), and whose output is indistinguishable from $\mathcal{Q}_{b_k}[M,f_{s_k};1^k,x,y]$.

Then we can use such a simulator to invert the PRG g with probability essentially half. The attack uses the non-uniform advice $\{(s_k,b_k,\mathcal{S}_k)\}_k$.

On input $(1^k, \beta)$: // *where* $\alpha \leftarrow \{0,1\}^k$ *and* $\beta = f(\alpha)$

1. If $b_k = 1$: choose $x \leftarrow \{0,1\}^k$ and run $Q \leftarrow \mathcal{S}_k(1^k, f(s_k \oplus g(x), \beta), x)$.
2. If $b_k = 2$: choose $y \leftarrow \{0,1\}^k$ and run $Q \leftarrow \mathcal{S}_k(1^k, f(s_k \oplus \beta, g(y)), y)$.
3. For each $(a,b) \in Q$: output a if $g(a) = \beta$; output b if $g(b) = \beta$.

By our assumption, Q is a polynomial-length list that, with probability essentially $1/2$, contains a value (a,b) such that $\beta \in \{g(a), g(b)\}$. Thus, we can output the preimage of β with probability essentially $1/2$. Since g is a PRG, and hence a one-way function, we have achieved a contradiction. Thus, the class of PRFs is not 2-hiding autoreducible.

Discussion and Interpretation. The proof considers only PRFs of a certain form. Let f be a fixed, Φ-RKA-secure PRF as above and g a fixed, length-doubling injective PRG. Then define $\widehat{\mathcal{C}}^{f,g}_{\text{PRF}} = \{f_s \mid s \in \{0,1\}^*\}$, where $f_s(x,y) = f(s \oplus g(x), g(y))$. More precisely, we have shown that $\widehat{\mathcal{C}}^{f,g}_{\text{PRF}}$ ($\subseteq \mathcal{C}_{\text{PRF}}$) is not 2-hiding autoreducible.

Admittedly, the class $\widehat{\mathcal{C}}^{f,g}_{\text{PRF}}$ is not the most natural subclass of PRFs. The result shown here leaves open the possibility that some other class $\mathcal{C} \subseteq \mathcal{C}_{\text{PRF}}$ of PRFs is 2-hiding autoreducible (and not in a trivial sense, as when $|\mathcal{C}| < \infty$). For instance, let F^g denote the well-known GGM construction [18] applied to a PRG g. Is the class $\mathcal{C}_{\text{GGM}} = \{F^g \mid g \text{ is a PRG}\} \subseteq \mathcal{C}_{\text{PRF}}$ 2-hiding autoreducible?[3]

5 Results for Malicious Security

In this section we describe two constructions of FBB MPC protocols that achieve security against malicious adversaries.

Autoreducibility Criterion. Our first construction is similar in spirit to the one given in Section 4. Like that construction, it is based on a variant of autoreducibility.

Definition 5. *Let $\mathcal{C}$ denote a class of functions of two inputs. Let M be a PPT oracle machine and define $\mathcal{Q}[M, f; z]$ to be the random variable containing the sequence of oracle queries made during the computation $M^f(z)$. We say that $\mathcal{C}$ is* **1-hiding autoreducible** *if there exists a PPT oracle machine M such that for all $f \in \mathcal{C}$:*

1. *For all x, y, we have that $\Pr[M^f(1^k, x, y) = f(x,y)]$ is overwhelming in k.*
2. *There exist PPT machine $\mathcal{S}_{f,1}$ and $\mathcal{S}_{f,2}$ such that for all x, y, the ensembles $\{\mathcal{S}_{f,1}(1^k, f(x,y), y)\}_k$, $\{\mathcal{S}_{f,2}(1^k, f(x,y), y)\}_k$, and $\{\mathcal{Q}[M, f; 1^k, x, y]\}_k$ are indistinguishable in k.*

[3] An alternative way to frame the question is as follows: define $\mathcal{F}_{\text{GGM}}$ to be the oracle functionality which takes input x from Alice, y from Bob, and evaluates the GGM construction on seed x, input y, and taking the oracle g as the underlying PRG. Let $\mathcal{C}_{\text{PRG}}$ denote the class of all PRGs. Is an FBB secure protocol possible for $\mathcal{F}_{\text{GGM}}^{\mathcal{C}_{\text{PRG}}}$?

In other words, M is able to use oracle access to f to determine $f(x, y)$, yet its oracle queries to f are "independent" of x and of y in some sense. A special case of Definition 5 is when the M's oracle queries are distributed uniformly (analogous to the definition of random self-reducibility, except defined with respect to a class of functions).

Theorem 2. *If $\mathcal{C}$ is 1-hiding autoreducible, then $\mathcal{F}_{\text{SFE}}^{\mathcal{C}}$ has a FBB, UC-secure (i.e., against malicious adversaries) protocol in the $\mathcal{F}_{\text{OT}}$-hybrid model.*

Proof (Proof sketch). The construction is quite similar to the one in the proof of Theorem 1. Both parties access an ideal functionality $\mathcal{F}_M$ which carries out an execution of the machine M from the definition of 1-hiding autoreducibility. In this setting, however, $\mathcal{F}_M$ gives output to *both* parties whenever M makes an oracle query. It then waits for responses from both Alice and Bob, and aborts if the two parties give different responses. Otherwise, it continues its simulation of M, using the parties' response as the oracle response. As before, when the simulation of M finishes, $\mathcal{F}_M$ gives the output to both parties.

The simulator for a corrupt Alice in the f-instantiated protocol is as follows. When Alice gives input (INIT, x) to $\mathcal{F}_M$, the simulator sends x to the ideal functionality and receives output $z = f(x, y)$. Then the simulator runs $(q_1, \ldots, q_t) \leftarrow \mathcal{S}_{f,1}(1^k, z, x)$. For $i \in [t]$, the simulator gives output (QUERY, q_i) to Alice on behalf of $\mathcal{F}_M$, and aborts if Alice responds with anything but $(\text{RESP}, f(q_i))$. Recall that the simulator can depend arbitrarily on f and can therefore compute and verify $f(q_i)$. Finally, the simulator gives (OUT, z) to Alice and delivers the output in the ideal model. Soundness of this simulation follows from the definition of 1-hiding autoreducibility. The simulation for corrupt Bob is analogous.

A GMW-style Protocol Compiler. We describe another way to construct malicious-secure FBB protocols, similar in spirit to the well-known GMW compiler [19]. We use zero-knowledge proofs to "compile" a semi-honest-secure protocol into a malicious-secure one, preserving the relevant FBB property.

Theorem 3. *Let $\mathcal{F}$ be an ideal functionality implemented as an oracle machine. If there exists a FBB protocol for $\mathcal{F}^{\mathcal{C}}$ secure against semi-honest adversaries in the $\mathcal{F}_{\text{OT}}$-hybrid model and a FBB protocol for $\mathcal{F}_{\text{FZK}}^{\mathcal{C}}$ (Section 3) that is UC-secure in the $\mathcal{F}_{\text{OT}}$-hybrid model, then there is an FBB protocol for $\mathcal{F}^{\mathcal{C}}$ that is UC-secure in the $\mathcal{F}_{\text{OT}}$-hybrid model.*

In other words, malicious security for the (possibly simpler) function $\mathcal{F}_{\text{FZK}}^{\mathcal{C}}$ can be leveraged to yield malicious security for $\mathcal{F}^{\mathcal{C}}$, while preserving the $\mathcal{C}$-FBB property.

Proof. The basic approach is to leverage the $\mathcal{F}_{\text{FZK}}$ functionality to ensure consistency of each party's oracle responses in the $\mathcal{F}^{\mathcal{C}}$ protocol. We first augment the $\mathcal{F}^{\mathcal{C}}$ protocol so that whenever one party is asked to make local oracle query q, the other party obtains a *commitment* of q. Then we augment $\mathcal{F}_{\text{FZK}}^{\mathcal{C}}$ to check

that the prover indeed provided the value q underlying the commitment, and also give the verifier a commitment to $f(q)$. Then both parties can return the commitment to $f(q)$ to the $\mathcal{F}^{\mathcal{C}}$ protocol to ensure that the query was answered consistently.

Let Com denote a statistically binding commitment scheme with non-interactive reveal phase. Let (π_A, π_B) be the FBB protocol for $\mathcal{F}^{\mathcal{C}}$. Define the ideal functionality $\widetilde{\mathcal{F}}_{\pi}$ to do the following:

1. Simulate π_A, π_B, and $\mathcal{F}_{\text{OT}}$ in the natural way, with inputs of Alice being fed into π_A, outputs of π_A being given to Alice, and likewise for Bob with π_B.
2. Whenever π_A makes oracle query q, honestly generate a commitment $c \leftarrow \mathsf{Com}(q)$ with decommitment value σ. Give $(\text{QUERY}, c, q, \sigma)$ to Alice and (QUERY, c) to Bob, and wait.
3. Upon receiving inputs $(\text{RESP}, c^*, \sigma^*)$ from Alice and (RESP, c^*) from Bob, ensure that π_A is currently waiting for an oracle response and that σ^* is a valid decommitment of c^* — say, to the value r^*. If so, then continue the simulation taking r^* as the oracle response for π_A.

The functionality has analogous behavior for Bob with respect to π_B. Let (ρ_P, ρ_V) be the FBB protocol for $\mathcal{F}_{\text{FZK}}^{\mathcal{C}}$. Define the ideal functionality $\widetilde{\mathcal{F}}_{\rho}$ to do the following:

1. Expect common input c. On input (INIT, σ) from the prover, ensure that σ is a valid decommitment of c, say, to the value q^*.
2. Simulate instances of ρ_P, ρ_V, and $\mathcal{F}_{\text{OT}}$ in the natural way with input q^* to the sender.
3. Whenever ρ_P makes an oracle query q, give (QUERY, q) to prover, expect (RESP, r) from the prover, and continue simulating the components with r as the oracle answer for ρ_P.
4. When ρ_V generates output r^* for the verifier, honestly generate a commitment $c^* = \mathsf{Com}(r^*)$ with decommitment value σ^*. Give $(\text{OUT}, c^*, \sigma^*)$ to the prover, and (OUT, c^*) to verifier.

Let $\widetilde{F}_{\text{FZK}}^{\mathcal{C}}$ denote the functionality which does the following:

1. Expect common input c. On input σ from the prover, ensure that σ is a valid decommitment of c, say, to the value q^*.
2. Query the oracle $f \in \mathcal{C}$ on input q^* to obtain $r^* = f(q^*)$.
3. Honestly generate a commitment $c^* = \mathsf{Com}(r^*)$ with decommitment value σ^*. Give (c^*, σ^*) to the prover, and c^* to verifier.

Then the protocol in the $\widetilde{\mathcal{F}}_{\rho}$-hybrid model (thus the $\mathcal{F}_{\text{OT}}$-hybrid model) which answers (QUERY, q) messages using the local oracle is a UC-secure, FBB protocol for $\widetilde{F}_{\text{FZK}}^{\mathcal{C}}$.

Now the UC-secure FBB protocol for $\mathcal{F}^{\mathcal{C}}$ is as follows. Parties give their initial inputs to $\widetilde{\mathcal{F}}_{\pi}$ and report outputs from $\widetilde{\mathcal{F}}_{\pi}$. Say that $\widetilde{\mathcal{F}}_{\pi}$ gives $(\text{QUERY}, c, q, \sigma)$ to Alice and (QUERY, c) to Bob. Then the parties invoke $\widetilde{F}_{\text{FZK}}^{\mathcal{C}}$ on common input

c, with Alice acting as the prover providing input σ. Alice obtains (c^*, σ^*) and Bob obtains c^*. Then Alice gives $(\text{RESP}, c^*, \sigma^*)$ and Bob gives (RESP, c^*) to $\widetilde{\mathcal{F}}_\pi$. Finally, when $\widetilde{\mathcal{F}}_\pi$ gives output (from π_A or π_B), the appropriate party reports it as their own output.

The security of this protocol follows from the binding and hiding properties of the commitment scheme, as well as the security properties of ρ and π. Proof is deferred to the full version.

6 FBB Zero-Knowledge Proofs (and Relaxations)

If f is a deterministic function, then $\mathcal{F}_{\text{FZK}}^f$ (defined in Section 3) is essentially a zero-knowledge proof functionality for the preimage relation $R_f(x, w) = 1 \Leftrightarrow f(w) = x$.

Theorem 4. *Define $\mathcal{C}_{\text{OWF}} = \{f \mid f \text{ is a OWF}\}$. If injective one-way functions exist, then there is no standalone-secure, FBB, honest-verifier witness-hiding protocol for $\mathcal{F}_{\text{FZK}}^{\mathcal{C}_{\text{OWF}}}$, even in the presence of an arbitrary trusted setup.*

We emphasize that only plain standalone-secure security is required, and that the protocol is not assumed to be a proof (argument) of knowledge. In particular, the above theorem rules out essentially any interesting relaxation of zero-knowledge proofs for the class of relations in question.

Proof. Suppose such a protocol exists, and call its algorithms (P^f, V^f). When the statement "$\exists w : R_f(x, w)$" is being proven, the honest prover runs $P^f(1^n, w)$ and the honest verifier runs $V^f(1^n, x)$.

Let f be an injective, length-increasing OWF f. In an interaction involving an honest verifier with input x, let $\mathcal{E}$ denote the event that his view contains a query to the oracle on a preimage of x. If $\Pr_w[\mathcal{E} \mid P^f(1^n, w) \rightleftarrows V^f(1^n, f(w))]$ is non-negligible in n (over choice of random $w \leftarrow \{0,1\}^n$), then we contradict the (honest verifier) witness-hiding property of the protocol.

So assume that this probability is negligible. Furthermore, in the interaction $P^f(1^n, w) \rightleftarrows V^f(1^n, f(w))$, the verifier accepts with overwhelming probability due to the completeness property.

Then by an averaging argument, there exists a negligible function δ and strings $w_1, w_2, \ldots$, with $|w_k| = k$, such that $\Pr[\mathcal{E} \mid P^f(1^n, w_n) \rightleftarrows V^f(1^n, f(w_n))] \leq \delta(n)$. Now for each n, let z_n be a string not in the image of $f(\{0,1\}^n)$, since f is length-increasing. Define:

$$f'_n(s) = \begin{cases} z_n & \text{if } s = w_n \\ f(s) & \text{else} \end{cases}.$$

Now f'_n is also a OWF and hence the security properties of the $\langle P, V \rangle$ protocol hold when instantiated with f'_n. Note that $R_f(f(w_n), w_n) = 1$ but $R_{f'_n}(f(w_n), w_n) = 0$, since f is injective.

Conditioned on V *not* querying its oracle on input w_n (an event which we assume happens with negligible probability), the outcomes $P^f(1^n, w_n) \rightleftarrows V^f(1^n, f(w_n))$ and $P^f(1^n, w_n) \rightleftarrows V^{f'_n}(1^n, f(w_n))$ are identical. That is, the verifier accepts with overwhelming probability when instantiated with either f or f'_n. Then when the prover runs the honest protocol with oracle f and input w_n, it constitutes a violation of soundness against an honest verifier instantiated with f'_n. More specifically, in such an interaction the honest verifier is instantiated with oracle f'_n yet accepts a statement about $R_{f'_n}$ that is false.

References

1. Abadi, M., Feigenbaum, J., Kilian, J.: On hiding information from an oracle. J. Comput. Syst. Sci. 39(1), 21–50 (1989)
2. Allender, E.: New surprises from self-reducibility. In: Ferreira, F., Löwe, B., Mayordomo, E., Gomes, L.M. (eds.) CiE 2010, Abstract and handout booklet, pp. 1–5 (2010)
3. Applebaum, B., Ishai, Y., Kushilevitz, E.: How to garble arithmetic circuits. In: Ostrovsky, R. (ed.) FOCS, pp. 120–129. IEEE (2011)
4. Barak, B., Goldreich, O., Impagliazzo, R., Rudich, S., Sahai, A., Vadhan, S.P., Yang, K.: On the (Im)possibility of Obfuscating Programs. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 1–18. Springer, Heidelberg (2001)
5. Beaver, D., Feigenbaum, J.: Hiding Instances in Multioracle Queries. In: Choffrut, C., Lengauer, T. (eds.) STACS 1990. LNCS, vol. 415, pp. 37–48. Springer, Heidelberg (1990)
6. Beaver, D., Feigenbaum, J., Kilian, J., Rogaway, P.: Locally random reductions: Improvements and applications. J. Cryptology 10(1), 17–36 (1997)
7. Bellare, M., Cash, D.: Pseudorandom Functions and Permutations Provably Secure against Related-Key Attacks. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 666–684. Springer, Heidelberg (2010)
8. Bellare, M., Kohno, T.: A Theoretical Treatment of Related-Key Attacks: RKA-PRPs, RKA-PRFs, and Applications. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 491–506. Springer, Heidelberg (2003)
9. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation. In: STOC, pp. 1–10. ACM (1988)
10. Boldyreva, A.: Threshold Signatures, Multisignatures and Blind Signatures Based on the Gap-Diffie-Hellman-Group Signature Scheme. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 31–46. Springer, Heidelberg (2003)
11. Boneh, D., Lynn, B., Shacham, H.: Short Signatures from the Weil Pairing. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 514–532. Springer, Heidelberg (2001)
12. Brassard, G.: Cryptology - column 4: hiding information from oracles. SIGACT News 21(2), 5 (1990)
13. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. Electronic Colloquium on Computational Complexity (ECCC) TR01-016 (2001); Previous version "A unified framework for analyzing security of protocols" availabe at the ECCC archive TR01-016. Extended abstract in FOCS 2001 (2001)

14. Canetti, R., Lindell, Y., Ostrovsky, R., Sahai, A.: Universally composable two-party and multi-party secure computation. In: STOC, pp. 494–503. ACM (2002)
15. Chaum, D.: Blind signatures for untraceable payments. In: Chaum, D., Rivest, R.L., Sherman, A.T. (eds.) CRYPTO, pp. 199–203. Plenum Press, New York (1982)
16. Chaum, D., Crépeau, C., Damgård, I.: Multiparty unconditionally secure protocols (extended abstract). In: STOC, pp. 11–19. ACM (1988)
17. Choi, S.G., Dachman-Soled, D., Malkin, T., Wee, H.: Simple, Black-Box Constructions of Adaptively Secure Protocols. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 387–402. Springer, Heidelberg (2009)
18. Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions. J. ACM 33(4), 792–807 (1986)
19. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or a completeness theorem for protocols with honest majority. In: STOC, pp. 218–229. ACM (1987)
20. Haitner, I., Ishai, Y., Kushilevitz, E., Lindell, Y., Petrank, E.: Black-box constructions of protocols for secure computation. SIAM J. Comput. 40(2), 225–266 (2011)
21. Impagliazzo, R., Rudich, S.: Limits on the provable consequences of one-way permutations. In: STOC, pp. 44–61. ACM (1989)
22. Ishai, Y., Prabhakaran, M., Sahai, A.: Founding Cryptography on Oblivious Transfer – Efficiently. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 572–591. Springer, Heidelberg (2008)
23. Kilian, J.: Founding cryptography on oblivious transfer. In: STOC, pp. 20–31. ACM (1988)
24. Kolesnikov, V., Schneider, T.: Improved Garbled Circuit: Free XOR Gates and Applications. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 486–498. Springer, Heidelberg (2008)
25. Lindell, Y.: Techniques for efficient secure two-party computation with malicious adversaries. Technical talk, The Check Point Institute Crypto and Security Day (2010)
26. Pass, R., Wee, H.: Black-Box Constructions of Two-Party Protocols from One-Way Functions. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 403–418. Springer, Heidelberg (2009)
27. Pinkas, B., Schneider, T., Smart, N.P., Williams, S.C.: Secure Two-Party Computation Is Practical. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 250–267. Springer, Heidelberg (2009)
28. Reingold, O., Trevisan, L., Vadhan, S.P.: Notions of Reducibility between Cryptographic Primitives. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 1–20. Springer, Heidelberg (2004)
29. Selke, J.: Autoreducibility and friends: About measuring redundancy in sets. Master's thesis, Gottfried-Wilhelm-Leibniz-Universität, Hannover (2008)
30. Yao, A.C.: How to generate and exchange secrets. In: FOCS, pp. 162–167. IEEE (1986)