On the Power of Public-key Encryption in Secure Computation

Mohammad Mahmoody[1], Hemanta K. Maji[2], and Manoj Prabhakaran[3]

[1] University of Virginia[⋆]
mohammad@cs.virginia.edu
[2] University of California, Los Angeles[⋆⋆]
hmaji@cs.ucla.edu
[3] University of Illinois, Urbana-Champaign[⋆⋆⋆]
mmp@cs.uiuc.edu

Abstract. We qualitatively separate semi-honest secure computation of non-trivial secure-function evaluation (SFE) functionalities from existence of key-agreement protocols. Technically, we show the existence of an oracle (namely, PKE-oracle) relative to which key-agreement protocols exist; but it is useless for semi-honest secure realization of symmetric 2-party (deterministic finite) SFE functionalities, i.e. any SFE which can be securely performed relative to this oracle can also be securely performed in the plain model.

Our main result has following consequences.

- There exists an oracle which is useful for some 3-party deterministic SFE; but useless for semi-honest secure realization of any general 2-party (deterministic finite) SFE.
- With respect to semi-honest, standalone or UC security, existence of key-agreement protocols (if used in black-box manner) is only as useful as the commitment-hybrid for general 2-party (deterministic finite) SFE functionalities.

This work advances (and conceptually simplifies) several state-of-the-art techniques in the field of black-box separations:

1. We introduce a general *common-information learning* algorithm (CIL) which extends the "eavesdropper" in prior work [1,2,3], to protocols whose message can depend on information gathered by the CIL so far.
2. With the help of this CIL, we show that in a secure 2-party protocol using an idealized PKE oracle, surprisingly, decryption queries are useless.
3. The idealized PKE oracle with its decryption facility removed can be modeled as a collection of *image-testable random-oracles*. We extend the analysis approaches of prior work on random oracle [1,2,4,5,3] to apply to this class of oracles. This shows that these oracles are useless for semi-honest 2-party SFE (as well as for key-agreement).

These information theoretic impossibility results can be naturally extended to yield black-box separation results (cf. [6]).

[⋆] Research done while at Cornell and supported in part by NSF Awards CNS-1217821 and CCF-0746990, AFOSR Award FA9550-10-1-0093, and DARPA and AFRL under contract FA8750-11-2- 0211.

[⋆⋆] Supported by NSF CI Postdoctoral Fellowship.

[⋆⋆⋆] Research supported in part by NSF grants 1228856 and 0747027.

1 Introduction

Public-key encryption (PKE) is an important security primitive in a system involving *more than two parties*. In this work, we ask if PKE could be useful for protecting two mutually distrusting parties against each other, *if there is no other party involved*. More specifically, we ask if the existence of PKE can facilitate 2-party secure function evaluation (SFE). Informally, our main result in this work shows the following:

> *The existence of PKE (as a computational complexity assumption, when used in a black-box manner) is useless for semi-honest secure evaluation of any finite, deterministic 2-party function.*

Here, a complexity assumption being "useless" for a task means that the task can be realized using that assumption alone (in a black-box manner) if and only if it can be realized unconditionally (i.e., information-theoretically).[4] As is typical in this line of research, our focus is on deterministic functions whose domain-size is finite. (However, all our results extend to the case when the domain-size grows *polynomially* in the security parameter; our proofs (as well as the results we build on) do not extend to exponentially growing domain-sizes, though.) Technically, we show an "oracle-separation" result, by presenting a randomized oracle which enables PKE in the information-theoretic setting, but does not enable SFE for any 2-party function for which SFE was impossible without the oracle. Then, using standard techniques, this information theoretic impossibility result is translated into the above black-box separation result [6]. While the above statement refers to semi-honest security, as we shall shortly see, a similar statement holds for security against active corruption, as well.

It is instructive to view our result in the context of "cryptographic complexity" theory [7]: with every (finite, deterministic) multi-party function f, one can associate a computational intractability assumption that there exists a secure computation protocol for f that is secure against semi-honest corruption.[5] Two assumptions are considered distinct unless they can be black-box reduced to each other. Then, the above result implies that secure key agreement (i.e., the interactive analog of PKE) does not belong to the universe of assumptions associated with 2-party functions. However, it is not hard to see that there are 3-party functions f such that a semi-honest secure protocol for f (in the broadcast channel model) is equivalent to a key agreement protocol.[6] Thus we obtain the following important conclusion:

[4] The task here refers to 2-party SFE in the "plain" model. We do not rule out the possibility that PKE is useful for 2-party SFE in a "hybrid" model, where the parties have access to a trusted third party.

[5] This is the simplest form of assumptions associated with functionalities in [7], where a more general framework is presented.

[6] As an example, consider the 3-party function $f(x, y, z) = x \oplus y$. A semi-honest secure protocol π for f over a broadcast channel can be black-box converted to a key-agreement protocol between Alice and Bob, where, say, Alice plays the role of the first party in π with the key as its input, and Bob plays the role of the second and third parties with random inputs. Conversely, a key-agreement protocol can be used as a black-box in a semi-honest secure protocol for f, in which the first party sends its input to the second party encrypted using a key that the two of them generate using the key-agreement protocol.

> *The set of computational complexity assumptions associated (in the above sense) with 3-party functions is strictly larger than the set associated with 2-party functions.*

This answers an open question posed in [7], but raises many more questions. In particular, we ask if the same conclusion holds if we consider $(n+1)$-party functions and n-party functions, for every $n > 2$.

Another consequence of our main result is its implications for SFE secure against *active* corruption. Following a related work in [5], using characterizations of functions that have SFE protocols secure against semi-honest and active corruptions [8,9,10,11], we obtain the following corollary of our main result.

> *The existence of PKE (as a black-box assumption) is exactly as useful as a commitment functionality (given as a trusted third party) for secure evaluation of any finite, deterministic 2-party function. This holds for semi-honest security, standalone active security and UC-security.*

Note that for semi-honest security, the commitment functionality is not useful at all (since semi-honest parties can commit using a trivial protocol), and this agrees with the original statement. The interesting part of the corollary is the statement about active (standalone or UC) security. Commitment is a "minicrypt" functionality that can be implemented using one-way functions (in the standalone setting) or random oracles. PKE, on the other hand, is not a minicrypt primitive [1]. Yet, in the context of guaranteeing security for *two* mutually distrusting parties, computing a (finite, deterministic) function, without involving a trusted third party, PKE is no more useful than the commitment functionality.

In the rest of this section, we state our main results more formally, and present an overview of the techniques. But first we briefly mention some of the related work.

1.1 Related Work

Impagliazzo and Rudich [1] showed that random oracles are not useful against a computationally unbounded adversary for the task of secure key agreement. This analysis was recently simplified and sharpened in [2,3]. Haitner, Omri, and Zarosim [12,3] show that random oracles are essentially useless in any *inputless protocol*.[7]

Following [1] many other black-box separation results have appeared (e.g. [13,14,15,16,17]). In particular, Gertner et. al [18] insightfully asked the question of comparing oblivious-transfer (OT) and key agreement (KA) and showed that OT is strictly more complex (in the sense of [1]). Another trend of results has been to prove lower-bounds on the efficiency of the implementation in black-box constructions (e.g. [19,20,21,22,23,2,22]). A complementary approach has been to find black-box reductions when they exist (e.g. [24,25,26,27,28]). Also, results in the black-box separation framework of [1,6]

[7] Ideally, a result similar to that of [3] should be proven in our setting of secure function evaluation too, where parties do have private inputs, as it would extend to randomized functions as well. While quite plausible, such a result remains elusive.

have immediate consequences for computational complexity theory. Indeed, as mentioned above, separations in this framework can be interpreted as new worlds in Impagliazzo's universe [29].

Our work relies heavily on [5], where a similar result was proven for one-way functions instead of PKE. While we cannot use the result of [5] (which we strictly improve upon) in a black-box manner, we do manage to exploit the modularity of the proof there and avoid duplicating any significant parts of the proof.

1.2 Our Contribution

For brevity, in the following we shall refer to "2-party deterministic SFE functions with polynomially large domains" simply as SFE functions. Also, we consider security against semi-honest adversaries in the information theoretic setting, unless otherwise specified (as in Corollary 1).

Our main result establishes that there exists an oracle which facilitates key-agreement while being useless to 2-party SFE.

Theorem 1. *There exists an oracle $\mathbb{PKE}$ such that, the following hold:*

- *There is a secure key-agreement protocol (or equivalently, a semi-honest secure 3-party XOR protocol) using $\mathbb{PKE}$.*
- *A general 2-party deterministic function f, with a polynomially large domain, has a semi-honest secure protocol against computationally unbounded adversaries using $\mathbb{PKE}$ if and only if f has a perfectly semi-honest secure protocol in the plain model.*

As discussed below, this proof breaks into two parts — a compiler that shows that the decryption queries can be omitted, and a proof that the oracle without the decryption queries is not useful for SFE. For proving the latter statement, we heavily rely on a recent result from [5] for random oracles; however, this proof is modular, involving a "frontier analytic" argument, which uses a few well-defined properties regarding the oracles. Our contribution in this is to prove these properties for a more sophisticated oracle class (namely, family of image-testable random oracles), rather than random oracles themselves.

As in [5], Theorem 1 translates to a black-box separation of the primitive PKE from non-trivial SFE. Also, it yields the following corollary, that against active corruption, our $\mathbb{PKE}$ oracle is only as useful as the commitment-hybrid model, as far as secure protocols for 2-party SFE is concerned.

Corollary 1. *There exists an oracle $\mathbb{PKE}$ such that, the following hold:*

- *There is a secure key-agreement protocol (or equivalently, a semi-honest secure 3-party XOR protocol) using $\mathbb{PKE}$.*
- *A general 2-party deterministic function f, with a polynomially large domain, has a statistically semi-honest, standalone or UC-secure protocol relative to $\mathbb{PKE}$ if and only if f has a perfectly, resp., semi-honest, standalone or UC-secure protocol in the commitment-hybrid.*

Apart from there results, and their implications to the complexity of 2-party and 3-party functions, we make important technical contributions in this work. As described below, our "common-information learner" is simpler than that in prior work. This also helps us handle a more involved oracle class used to model PKE. Another module in our proof is a compiler that shows that the decryption facility in PKE is not needed in a (semi-honest secure) protocol that uses PKE, even if the PKE is implemented using an idealized oracle.

1.3 Technical Overview

The main result we need to prove (from which our final results follow, using arguments in [5]) is that there is an oracle class $\mathbb{PKE}_\kappa$ relative to which secure public-key encryption (i.e., 2-round key agreement) protocol exists, but there is no secure protocol for any non-trivial SFE function relative to it.

The oracle class $\mathbb{PKE}_\kappa$ is a collection of following correlated oracles:

- $\mathsf{Gen}(\cdot)$: It is a (length-tripling injective) random oracle which maps secret keys sk to respective public keys pk.
- $\mathsf{Enc}(\cdot,\cdot)$: For each public key pk, it is an independently chosen (length tripling injective) random oracle which maps messages m to cipher texts c.
- $\mathsf{Dec}(\cdot,\cdot)$: Given a valid secret key sk and a valid cipher text c it outputs m such that message m was encrypted using public key $pk = \mathsf{Gen}(sk)$.
- Additionally, it provides test oracles Test which output whether a public key pk is a valid public key or not; and whether a cipher text c has been created using a public key pk or not.

Note that without the Test oracle, this oracle class can be used to semi-honest securely perform OT; hence, all 2-party SFE will be trivial relative to it (see discussion in [18,30]). The main technical contribution of this paper is the negative result which shows that the above mentioned oracle class $\mathbb{PKE}_\kappa$ is useless for 2-party SFE against semi-honest adversaries.

This is shown in two steps:

1. First, we show that the decryption oracle $\mathsf{Dec}(\cdot,\cdot)$ is not useful against semi-honest adversaries. That is, given a (purported) semi-honest secure protocol ρ for a 2-party SFE f we compile it into another semi-honest secure protocol Π (with identical round complexity, albeit polynomially more query complexity) which has slightly worse security but performs no decryption-queries.
2. Finally, we observe that the oracle class "$\mathbb{PKE}_\kappa$ minus the decryption oracle" is identical to image-testable random-oracles. And we extend the negative result of [5] to claim that this oracle class is useless for 2-party SFE.

The key component in both these steps is the *Common Information Learner* algorithm, relative to image-testable random oracle class. But we begin by introducing image-testable random oracles.

Image-testable Random-oracle Class. It is a pair of correlated oracles (R, T), where R is a (length-tripling injective) random oracle and test oracle T which outputs whether a point in range has a pre-image or not. We consider *keyed-version* of these oracle, where for each key in an exponentially large key-space $\mathbb{K}$ we have an independent copy of image-testable random oracle.

Note that the answer to an image test query can slightly change the distribution of exponentially many other queries; namely, when we know that y is not in the image of R, the answer to any query x for R will not be uniformly distributed (because it cannot be y). However, since the number of tested images are polynomial-size and the number of possible queries to R are exponentially large, this will affect the distribution of the answers by R only negligibly. Also, because of the expansion of the random oracle R, the fraction of the image-size of R is negligibly small relative to the range of R. So an algorithm, with polynomially-bounded query complexity, who queries the test oracle T has negligible chance of getting a positive answer (i.e. an image) without actually calling R. We emphasize that our whole analysis is conditioned on this event (i.e. accidentally discovering y in the image of the oracle) not taking place; and this requires careful accounting of events because it holds only for (polynomially) bounded query algorithms.

Common Information Learner. The common information learner is a procedure that can see the transcript of an oracle-based protocol between Alice and Bob, and by making a polynomial number of publicly computable queries to the oracle, derives sufficient information such that conditioned on this information, the views of Alice and Bob are almost independent of each other. Our common information learner is similar in spirit to those in [1,2,5,3] but is different and more general in several ways:

- **Handling Image-Testable Oracles.** Our common information learner applies to the case when the oracle is not just a random oracle, but an image-testable random oracle family.[8]
- **Interaction between Learner and the System.** It is important for the first part of our proof (i.e. compiling out the decryption queries) that the common information learner *interacts with the protocol execution itself*. That is, at each round the information gathered by the common information learner is used by the parties in subsequent rounds. We require the common information learner to still make only a polynomial number or oracle queries while ensuring that conditioned on the information it gathers, the views of the two parties remain almost independent. In showing that the common information learner is still efficient, we show a more general result in terms of an interactive process between an oracle system (the Alice-Bob system, in our case) and a learner, both with access to an arbitrary oracle (possibly correlated with the local random tapes of Alice and Bob).
- **Simpler Description of the Learner.** The common information learner in our work has a simpler description than that in [1,2,5]. Our learner queries the random oracle with queries that are most likely to be queried by Alice or Bob in a protocol execution. The learner in [2,5] is similar, but uses probabilities not based on the actual

[8] The work of [3] also handles a larger set of oracles than random oracles (called *simple* oracle), but that class is not known to include image-testable oracles as special case [31].

protocol, but a variant of it; this makes the description of their common information learner more complicated, and somewhat complicates the proofs of the query efficiency of the learner.[9]

Showing that Image-Testable Random Oracles are Useless for SFE. In [5] it was shown that random oracles are useless for SFE. This proof is modular in that there are four specific results that depended on the nature of the oracle and the common information learner. The rest of the proof uses a "frontier analytic" argument that is agnostic to the oracle and the common information learner. Thus, in this work, to extend the result of [5] to a family of image-testable random oracles, we need only ensure that these four properties continue to hold. The four properties are as follows:

1. Alice's message conditioned on the view of the CIL is almost independent of Bob's input, see Section 6.1 item 1.
2. Safety holds with high probability, see Section 6.1 item 2.
3. A strong independence property of Alice's and Bob's views conditioned on that of the CIL, see Section 6.1 item 3.
4. Finally, local samplability and oblivious rerandomizability of image-testable random oracles which permit simulation of alternate views, see Section 6.

Compiling Out the Decryption Queries. The main idea behind compiling out the decryption queries is that if Alice has created a ciphertext by encrypting a message using a public-key that was created by Bob, and she realizes that there is at least a small (but significant) probability that Bob would be querying the decryption oracle on this ciphertext (since he has the secret key), then she would preemptively send the actual message to him. We need to ensure two competing requirements on the compiler:

1. **Security.** Note that with some probability Alice might send this message even if Bob was not about to query the decryption oracle. To argue that this is secure, we need to argue that a curious Bob *could have* called the decryption oracle at this point, for the same ciphertext.
2. **Completeness.** We need to ensure that in the compiled protocol, Bob will never have to call the decryption oracle, as Alice would have sent him the required decryptions ahead of time.

For security, firstly we need to ensure that Alice chooses the set of encryptions to be revealed based *only* on the common information that Alice and Bob have. This ensures that Bob can sample a view for himself from the same distribution used by Alice to compute somewhat likely decryption queries, and obtain the ciphertext and secret-key from the decryption query made in this view. The one complication that arises here is the possibility that the secret-key in the sampled view is not the same as the secret-key in the actual execution. To rule this out, we rely on the independence of the views of the parties conditioned on the common information. This, combined with the fact that it is unlikely for a valid public-key to be discovered by the system without either party

[9] [1] uses an indirect mechanism to find the heavy queries, and reasoning about their common information learner is significantly more involved.

having actually called the key-generation oracle using the corresponding secret-key, we can show that it is unlikely for a sampled view to have a secret-key different from the actual one.

For completeness of the compiler, we again rely on the common information learner to ensure that if Alice uses the distribution based on common information to compute which decryption queries are likely, then it is indeed unlikely for Bob to make a decryption query that is considered unlikely by Alice.

1.4 Overview of the paper

The full version of the paper is available at [32]. In Section 2 we formally define all the relevant oracle classes. Section 3 introduces relevant definitions and notations for this paper. The efficiency of an algorithm which performs heavy-queries is argued in Section 4.1. This is directly used to provide an independence learner for protocols where parties do not have private inputs in Section 4.2. In Section 5 we show that for 2-party deterministic SFE Decryption queries in $\mathbb{PKE}_\kappa$ are useless. Next, in Section 6, we extend Lemma 2 to protocols where parties have private inputs. Finally, we prove our main result (Theorem 1) in Section 7.

2 Oracle Classes

General class of oracles shall be represented by $\mathbb{O}$. We are interested in three main classes of oracles, each parameterized by the security parameter κ.

2.1 Image-testable Random Oracle Class

The set $\mathbb{O}_\kappa$ consists of the all possible pairs of correlated oracles $O \equiv (R, T)$ of the form:

1. $R : \{0,1\}^\kappa \mapsto \{0,1\}^{3\kappa}$ is a function, and
2. $T : \{0,1\}^{3\kappa} \mapsto \{0,1\}$ is defined by: $T(\beta) = 1$ if there exists $\alpha \in \{0,1\}^\kappa$ such that $R(\alpha) = \beta$; otherwise $T(\beta) = 0$.

This class of oracles is known as *image-testable random oracle* class. Based on the length of the query string we can uniquely determine whether it is a query to R or T oracle. We follow a notational convention. Queries to R oracle shall be denoted by α and its corresponding answer shall be denoted by β.

2.2 Keyed Version of Image-testable Random Oracle Class

Given a class $\mathbb{K}$ of keys,[10] consider the following oracle $O^{(\mathbb{K})}$: For every $k \in \mathbb{K}$, let $O^{(k)} \in \mathbb{O}_\kappa$. Given a query $\langle k, q \rangle$, where $k \in \mathbb{K}$ and q is the query to an oracle in $\mathbb{O}_\kappa$, answer it with $O^{(k)}(q)$. Let $\mathbb{O}_\kappa^{(\mathbb{K})}$ be the set of all possible oracles $O^{(\mathbb{K})}$. This class of oracle $\mathbb{O}_\kappa^{(\mathbb{K})}$ is called *keyed-version of image-testable random oracle* class.

[10] Note that the description of the keys in $\mathbb{K}$ is $\mathsf{poly}(\kappa)$; so the size of the set $\mathbb{K}$ could possibly be exponential in κ.

2.3 Public-key Encryption Oracle

We shall use a "PKE-enabling" oracle similar to the one used in [18]. With access to this oracle, a semantically secure public-key encryption scheme can be readily constructed,[11] yet we shall show that it is useless for SFE. This oracle, which we will call $\mathbb{PKE}_\kappa$ (or simply $\mathbb{PKE}$, when κ is understood), is a collection of the oracles $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Test}_1, \mathsf{Test}_2, \mathsf{Dec})$ defined as follows:

- Gen: It is a length-tripling random oracle from the set of inputs $\{0,1\}^\kappa$ to $\{0,1\}^{3\kappa}$. It takes as input a secret key sk and provides a public-key pk corresponding to it, i.e. $\mathsf{Gen}(sk) = pk$.
- Enc: This is an "encryption" oracle. It can be defined as a collection of length-tripling random oracles, keyed by strings in $\{0,1\}^{3\kappa}$. For each key $pk \in \{0,1\}^{3\kappa}$, the oracle implements a random function from $\{0,1\}^\kappa$ to $\{0,1\}^{3\kappa}$. When queried with a (possibly invalid) public key pk, and a message $m \in \{0,1\}^\kappa$, this oracle provides the corresponding cipher text $c \in \{0,1\}^{3\kappa}$ for it, i.e. $\mathsf{Enc}(pk, m) = c$.
- Test_1: It is a test function which tests the validity of a public key, i.e. given a public-key pk, it outputs 1 if and only if there exists a secret key sk such that $\mathsf{Gen}(sk) = pk$.
- Test_2: It is a test function which tests the validity of a public key and cipher text pair, i.e. given a public-key pk and cipher text c, it outputs 1 if and only if there exists m such that $\mathsf{Enc}(pk, m) = c$.
- Dec: This is the decryption oracle, from $\{0,1\}^\kappa \times \{0,1\}^{3\kappa}$ to $\{0,1\}^\kappa \cup \{\perp\}$, which takes a secret-key, cipher-text pair (sk, c) and returns the lexicographically smallest m such that $\mathsf{Enc}(\mathsf{Gen}(sk), m) = c$. If no such m exists, it outputs $\perp$.

We note that the encryption oracle produces cipher texts for public keys pk irrespective of whether there exists sk satisfying $\mathsf{Gen}(sk) = pk$. This is crucial because we want to key set $\mathbb{K}$ to be defined independent of the Gen oracle.

$\mathbb{PKE}_\kappa$ *Without* Dec. We note that if we remove the oracle Dec, the above oracle is exactly the same as the image-testable random oracle $\mathbb{O}_\kappa^{(\mathbb{K})}$, with $\mathbb{K} = \{0,1\}^{3\kappa} \cup \{\perp\}$. Here we identify the various queries to $\mathbb{PKE}_\kappa$ with queries to $\mathbb{O}_\kappa^{(\mathbb{K})}$ as follows: $\mathsf{Gen}(sk)$ corresponds to the query $\langle \perp, sk \rangle$, $\mathsf{Enc}(pk, m)$ corresponds to $\langle pk, m \rangle$, $\mathsf{Test}_1(pk)$ corresponds to $\langle \perp, pk \rangle$ and $\mathsf{Test}_2(pk, c)$ corresponds to $\langle pk, c \rangle$.

3 Preliminaries

We say $a = b \pm c$ if $|a - b| \leq c$. We shall use the convention that a random variable shall be represented by a bold face, for example $\mathbf{X}$; and a corresponding value of the random variable without bold face, i.e. X in this case. We say that two distributions $\mathcal{D}_1$ and $\mathcal{D}_2$ are ε-close to each other if $\mathbf{\Delta}\left(\mathcal{D}_1, \mathcal{D}_2\right) \leq \varepsilon$.

[11] To encrypt a message of length, say, $\kappa/2$, a random string of length $\kappa/2$ is appended to it, and passed to the "encryption" oracle, along with the public-key.

Two-party Secure Function Evaluation. Alice and Bob have inputs $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ and are interested in evaluating $f(x, y)$ securely, where f is a deterministic function with output space $\mathcal{Z}$.

Protocols and Augmented Protocols. We shall consider two-party protocols π between Alice and Bob relative to an oracle class. Alice and Bob may or may not have private inputs for the parties. An augmentation of the protocol with a third party Eve, represented as π^+, is a three party protocol where parties have access to a broadcast channel and speak in following order: Alice, Eve, Bob, Eve, and so on. In every round one party speaks and then Eve speaks.

Views of Parties. We shall always consider Eve who have no private view; her complete view is public. Such Eve shall be referred to as *public query-strategy*. Transcript message sent by Eve in a round is her sequence of oracle query-answer pairs performed in that round. The oracle query-answer sets of Alice, Bob and Eve are represented by P_A, P_B and P_E, respectively. The transcript is represented by m (note that m only contains messages from Alice and Bob). View of Eve is $V_E = (m, P_E)$; view of Alice is $V_A = (x, r_A, m, P_A, P_E)$ (where x is Alice's private input and r_A is her local random tape; in input-less protocols x is not present) and view of Bob is $V_B = (y, r_B, m, P_B, P_E)$.

If i is odd then Alice performs local query-answers $P_{A,i}$ and sends the message m_i in that round followed by Eve message $P_{E,i}$. If i is even then Bob sends the message m_i, followed by Eve message $P_{E,i}$. View of Alice up to round i is represented by $V_A^{(i)} = (x, r_A, m^{(i)}, P_A^{(i)}, P_E^{(i)})$, where $m^{(i)} = m_1 \dots m_i$; and $P_A^{(i)}$ and $P_E^{(i)}$ are similarly defined.

Relative to oracle class $\mathbb{O}_\kappa^{(\mathbb{K})}$. Our sample space is distribution over complete Alice-Bob joint views when: $r_A \xleftarrow{\$} \mathbf{U}$, $r_B \xleftarrow{\$} \mathbf{U}$ and $O \xleftarrow{\$} \mathbb{O}_\kappa^{(\mathbb{K})}$.

Definition 1 (Canonical). *A canonical sequence of query-answer pairs is a sequence of query-answer pairs such that an R-query of form $\langle k, \alpha \rangle$ is immediately followed by a T-query of form $\langle k, \beta \rangle$, where the query $\langle k, \alpha \rangle$ was answered by β.*

Definition 2 (Normal Form for Protocols). *A three party protocol between Alice, Bob and (public query strategy) Eve is in normal form, if:*

1. *In every round Alice or Bob sends a message; followed by a sequence of query-answer pairs from Eve. We allow Alice and Bob to base their messages on prior messages broadcast by Eve.*
2. *In rounds $i = 1, 3, \dots$ Alice sends the message m_i; and in $i = 2, 4, \dots$ Bob sends a message.*
3. *In every round i after Alice/Bob has sent the message m_i, Eve broadcasts $P_{E,i}$.*
4. *Alice, Bob and Eve always perform canonical queries.*

4 Common Information Learner

In this section we shall introduce a *Heavy-query Performer* algorithm (see Fig. 2). Using this heavy querier, we shall *augment* any two-party protocol with a third party algorithm. Relative to the oracle class $\mathbb{O}_\kappa^{(\mathbb{K})}$ we show that the distribution of Alice-Bob joint views is (nearly) independent of each other conditioned on the transcript of the augmented protocol. Thus, the third party is aptly called an *common information learner* (see Eve_π in Fig. 3).

4.1 Heavy-query Performer

In this section we shall introduce a *Heavy-query Performer* algorithm. Let $\mathbb{O}$ be a finite class of oracles with finite domain D. Our experiment is instantiated by an oracle system Σ and a deterministic "Heavy-query Performer" $\mathcal{H}$ (with implicit parameter σ, see Fig. 2).

The oracle system Σ takes a random tape as input which has finite length. Let $\mathbb{S}$ be the set of pairs of random tape r for Σ and oracle $O \in \mathbb{O}$. The system Σ could possibly be computationally unbounded; but its round complexity is finite.

Consider the experiment in Fig. 1.

1. Let $\mathcal{D}_\mathbb{S}$ be a distribution over $\mathbb{S}$ such that $\mathsf{Supp}(\mathcal{D}_\mathbb{S}) = \mathbb{S}$. Sample $(r, O) \sim \mathcal{D}_\mathbb{S}$.
2. Start an interactive protocol between $\Sigma^O(r)$, i.e. the oracle system Σ with access to oracle O and local random tape r, and the heavy-query performer $\mathcal{H}$.

Fig. 1: Protocol between Oracle system Σ and the Heavy-query Performer $\mathcal{H}$.

We emphasize that the heavy-query performer $\mathcal{H}$ never performs a query unless its answer is uncertain. If the answer to the query q^* in uncertain, we say that the answer to this query has (max) entropy. Let $\mathcal{Q}_\Sigma\left(\left\langle \Sigma^O(r), \mathcal{H}\right\rangle\right)$ represent the query-answer set of the oracle system Σ when its local random tape is r, has oracle access to O and is interacting with the heavy-query performer $\mathcal{H}$. Similarly, $\mathcal{Q}_\mathcal{H}\left(\left\langle \Sigma^O(r), \mathcal{H}\right\rangle\right)$ represents the query-answer set of the heavy-query performer $\mathcal{H}$ which were actually performed to the oracle when Σ has local random tape r and has oracle access to O. Note that $\mathcal{Q}_\Sigma\left(\left\langle \Sigma^O(r), \mathcal{H}\right\rangle\right)$ and $\mathcal{Q}_\mathcal{H}\left(\left\langle \Sigma^O(r), \mathcal{H}\right\rangle\right)$ could possibly be correlated to each other.

Efficiency of the Heavy-query Performer. We argue that the expected query complexity of the heavy-query performer cannot be significantly larger than the query complexity of the system Σ itself:

Lemma 1 (Efficiency of Heavy-query Performer). *Let $\mathcal{D}_\mathbb{S}$ be a joint distribution over the space $\mathbb{S}$ as defined above. For every (randomized) oracle system Σ, the expected query complexity of the heavy-query performer $\mathcal{H}$ (presented an Fig. 2) is at most $\frac{1}{\sigma}$*

After every message sent by the oracle system Σ, perform the following step:

- Repeatedly call Heavy-Query-Finder to obtain a query-answer pair (q^*, a^*); and add the query-answer pair (q^*, a^*) to the transcript T. Until it reports that there are no more heavy queries left.

Heavy-Query-Finder: Let T be the transcript between the oracle system Σ and heavy-query performer $\mathcal{H}$. The messages added by Σ are represented by T_Σ and the set of query-answer pairs added by $\mathcal{H}$ are represented by $T_\mathcal{H}$. It has an implicit parameter σ, which is used to ascertain whether a query is heavy or not.

1. For every $q \in D \setminus T_\mathcal{H}$, compute the probability that Σ performs the query q when $(\tilde{r}, \widetilde{O}) \sim \mathcal{D}_\mathbb{S}$ conditioned on transcript T.
2. If there is no query q with probability $\geq \sigma$ then report that there are no more heavy queries left and quit. Otherwise, let q^* be the lexicographically smallest such query.
3. If the answer to q^* is uncertain (given the transcript T) then query O at q^* and obtain the answer a^*. Otherwise, let a^* be the fixed answer to q^*.
4. Return (q^*, a^*).

Fig. 2: Heavy-Query-Performer $\mathcal{H}$.

times the expected query complexity of the oracle system Σ in the experiment shown in Fig. 1. Formally,

$$\mathop{\mathbb{E}}_{(r,O)\sim\mathcal{D}_\mathbb{S}}\left[\left|\mathcal{Q}_\mathcal{H}\left(\left\langle\Sigma^O(r),\mathcal{H}\right\rangle\right)\right|\right] \leq \frac{\mathbb{E}_{(r,O)\sim\mathcal{D}_\mathbb{S}}\left[\left|\mathcal{Q}_\Sigma\left(\left\langle\Sigma^O(r),\mathcal{H}\right\rangle\right)\right|\right]}{\sigma}$$

In particular, the probability that $\mathcal{H}$ asks more than $\frac{\mathbb{E}_{(r,O)\sim\mathcal{D}_\mathbb{S}}\left[\left|\mathcal{Q}_\Sigma\left(\left\langle\Sigma^O(r),\mathcal{H}\right\rangle\right)\right|\right]}{\sigma^2}$ queries is at most σ.

The proof is provided in the full version of the paper [32]. We mention some highlights of the current proof. The proof is significantly simpler and is more general than the ones presented in [2,3] because our learner is directly working with heavy queries rather than concluding the heaviness of the queries being asked by the learner. Also note that in our setting the oracles might be correlated with local random tape of the system Σ; and the future messages of the oracle system Σ could, possibly, depend on prior messages of $\mathcal{H}$. We also note that the same proof also works in the setting where Σ cannot read the transcript T [12] but $\mathcal{H}$ also considers *queries performed in the future* by Σ while computing the set of heavy-queries.[13] We emphasize that it is possible that the future messages of the oracle system Σ could possible depend on the prior messages sent by the heavy-query performer $\mathcal{H}$. This property is inherited by Lemma 2, which (in turn) is crucially used by Theorem 2.

[12] More specifically, it cannot read $T_\mathcal{H}$; note that Σ already knows the part T_Σ generated by Σ itself.

[13] Note that if Σ can also read from T then the distribution of future queries is not well defined. But if Σ cannot read T, then future queries are well defined after (r, O) is instantiated.

Specific to Image-testable Random-oracles. Relative to the oracle class $\mathbb{O}_\kappa^{(\mathbb{K})}$, we can make an assumption that after performing a R-query $\langle k, \alpha \rangle$ and receiving β as answer, it immediately performs the next query as $\langle k, \beta \rangle$. Note that this query has no entropy (because this query will surely be answered 1); and, hence, need not be performed to the oracle.

4.2 Common Information Learner for Input-less Protocols

In this section we shall consider two-party protocols where parties have access to an oracle $O \in \mathbb{O}_\kappa^{(\mathbb{K})}$. For a two-party input-less protocol π, we augment it with the following eavesdropper strategy, referred as Eve_π, to obtain π^+:

1. Interpret the two-party oracle protocol π as the oracle system Σ in Fig. 1. Messages produced by Alice or Bob in round i is interpreted as the message of Σ.
2. Define $\mathbb{O} = \mathbb{O}_\kappa^{(\mathbb{K})}$ and $\mathcal{D}_\mathbb{S}$ as the uniform distribution over $\mathbb{O}$ and the space of local random tapes of Alice and Bob.
3. Let Eve_π be the heavy-query performer algorithm in Fig. 2 instantiated with a suitably small parameter σ.

Fig. 3: Eavesdropper strategy to augment an input-less protocol π.

Note that the query-complexity of Eve_π is $\mathsf{poly}(\kappa)$ with $1 - {}^1\!/_{\mathsf{poly}(\kappa)}$ probability, if σ is set to $1/\mathsf{poly}(\kappa)$ and the query complexity of the parties in π is (at most) $\mathsf{poly}(\kappa)$ (due to Lemma 1).

Lemma 2 (Common Information Learner for Input-less Protocols). *Let π be an input-less protocol in normal form between Alice and Bob relative to $\mathbb{O}_\kappa^{(\mathbb{K})}$, and Eve_π be as defined in Fig. 3. Let the distributions $\mathbf{V}_{AB}^{(i)}$ and $\mathbf{V}_{A\times B}^{(i)}$ for each round i of π be as follows:*

1. $\mathbf{V}_{AB}^{(i)} = (\mathbf{V}_A^{(i)}, \mathbf{V}_B^{(i)})$;
2. *Distribution $\mathbf{V}_{A\times B}^{(i)}$ defined as: Sample $V_E^{(i)} \sim \mathbf{V}_E^{(i)}$ and output $(V_A, V_B) \sim (\mathbf{V}_A^{(i)} | V_E^{(i)}) \times (\mathbf{V}_B^{(i)} | V_E^{(i)})$.*

For every $\varepsilon = 1/\mathsf{poly}(\kappa)$, there exists a choice of Eve_π's parameter $\sigma = 1/\mathsf{poly}(\kappa)$ such that, for every i,

$$\mathbf{\Delta}\left(\mathbf{V}_{AB}^{(i)}, \mathbf{V}_{A\times B}^{(i)}\right) \leq \varepsilon.$$

Below, we sketch the ideas behind proving this lemma. Interested reader may refer to the full version of this paper [32] for the proof.

The case of random oracles. First we consider the case of random oracles without image-testing. This case was already analyzed in [1,2], but it will be helpful to rephrase this proof, so that we can extend it to the case when image-testing is present. At the beginning of the execution, the views of Alice and Bob are indeed independent of each other. As the execution progresses, at each round, we introduce a "tidy" distribution over (V_A, V_B, V_E), which has the following properties: a tidy distribution is obtained simply by restricting the support of the real execution to "good" tuples. Below, (P_A, P_B, P_E) stand for the query-answer sets of (V_A, V_B, V_E).

Definition 3 (Good). *Three query-answer sets P_A, P_B and P_E are called* good*, represented by* $\mathsf{Good}(P_A, P_B, P_E)$*, if* $\mathsf{Consistent}(P_A \cup P_B \cup P_E)$ *and* $P_A \cap P_B \subseteq P_E$*.*

This has the consequence that a tidy distribution is identical to a "conditional product distribution" – i.e., a distribution which, when conditioned on each Eve view in its support, is a product distribution – when restricted to the same support as the tidy distribution.

When the execution evolves for one step (an Alice or Bob round), we start with the tidy distribution at that step, but will end up with a distribution that is not tidy. This distribution is again tidied up to obtain a new tidy distribution.

Then we argue the following:

1. Claim: At any point, the tidy distribution is close to a "conditional product distribution" – i.e., a distribution which, when conditioned on each Eve view in its support, is close to a conditional product distribution.

This closeness property is maintained inductively. Indeed, during an Eve round, it is easy to see that the distance from the conditional product distribution can only decrease. In an Alice or Bob step, we bound the *additional* distance from a conditional product distribution using the fact that, since Eve had just finished its step before the beginning of the current step, every query not in Eve's view was of low probability for either party ("lightness" guarantee). A lightness threshold parameter for Eve controls this additional distance.

2. Claim: After each Alice or Bob step, the statistical difference incurred in modifying the resulting distribution to become a tidy distribution is small.

Note that in an Alice or Bob step, even tough we start from a tidy distribution, after that step, tuples that are not good can indeed be introduced. But their probability mass can be bounded by the fact that the tidy distribution was close to a conditional product distribution.

Thus at each step, the statistical difference from the actual execution incurred by tidying up can be bounded, as well as the distance of the tidy distribution from a conditional product distribution. By choosing the lightness threshold parameter for Eve to be sufficiently small, after a polynomial number of steps, we obtain that the distribution of (V_A, V_B, V_E) in the actual execution is close to a tidy distribution, which is in turn close to a conditional product distribution.

We remark that, the "lightness" guarantee in [2] was ensured directly for the tidy distribution. However, it is enough to ensure that the lightness holds for the original distribution, since the tidy distribution is obtained by restricting the support of the actual distribution (without changing the relative probabilities within the support). This allows for a more modular description of the Eve's dropper's algorithm, independent of the definition of the tidy distributions. This turns out to be helpful when we move to the setting of image-testable random oracles, where the tidy distributions are much more complicated.

The case of image-testable random oracles. To adapt the above argument to accommodate test queries, we need to change several elements from above. Firstly, we replace the notion of good tuples, with a more refined notion of "nice" tuples, which takes into account the presence of *positive* test queries. (As it turns out, negative test queries by themselves have a negligible effect in the probability of individual views.)

Given a query-answer set P relative to $\mathbb{O}_\kappa^{(\mathbb{K})}$, we say that a query $q = \langle k, \beta \rangle \in \mathbb{K} \times \{0,1\}^{3\kappa}$ is *unexplained* if $(q, 1) \in P$ (i.e. $T(q) = 1$) but there is no $q' = \langle k, \alpha \rangle \in \mathbb{K} \times \{0,1\}^{\kappa}$ such that $(q, \beta) \in P$ (i.e. $R(q') = \beta$). We define $\mathsf{T_1Guess}(P)$ as the total number of unexplained queries in P.

Definition 4 (Typical and Nice Views). *A query-answer set P relative to $\mathbb{O}_\kappa^{(\mathbb{K})}$ is* typical*, represented by* $\mathsf{Typical}(P)$*, if* $\mathsf{T_1Guess}(P|_k) = 0$*, for every* $k \in \mathbb{K}$.

Alice, Bob and Eve views in a normal protocol are called nice*, represented as* $\mathsf{Nice}(V_A, V_B, V_E)$ *if:*

1. $\mathsf{Consistent}(P_A, P_B, P_E)$*,* $\mathsf{Good}(P_A, P_B, P_E)$*, and*
2. $\mathsf{Typical}(P_A \cup P_B \cup P_E)$*,* $\mathsf{Typical}(P_A \setminus P_E)$ *and* $\mathsf{Typical}(P_B \setminus P_E)$.

Apart from replacing goodness with niceness, the tidy distributions we use are different in a few other important ways. Firstly, a tidy distribution's support would typically not contain all the nice tuples in the actual execution; we remove certain kinds of nice tuples too from the support, to ensure that test queries do not lead to increased distance from a conditional product distribution. Secondly, to ensure that a tidy distribution is identical to a conditional product distribution, when the latter is restricted to the supported of the former, we let it be different from the actual distribution restricted to the same support. The definition of niceness however, ensures that this difference is at most a $1 \pm \mathsf{negl}$ factor point-wise. (The $1 \pm \mathsf{negl}$ factor corresponds to the *negative* test queries in the actual distribution, which are ignored in defining the probabilities in a tidy distribution.)

More formally, let $\mathcal{A}_i$, $\mathcal{B}_i$ and $\mathcal{E}_i$ denote the set of possible views of Alice, Bob and Eve respectively, after i steps of the augmented protocol execution (where each "round" consists of an Alice or Bob step, and an Eve step). To specify a tidy distribution Γ over the views after i steps of the augmented protocol we need to specify two sets $\mathcal{S}_A^{(i)} \subseteq \mathcal{A}_i \times \mathcal{E}_i$ and $\mathcal{S}_B^{(i)} \subseteq \mathcal{B}_i \times \mathcal{E}_i$. Then the distribution is defined as follows:

$$\Gamma(V_A, V_B, V_E) = \begin{cases} Z \cdot \gamma(V_A, V_E) \cdot \gamma(V_B, V_E) & \text{if } (V_A, V_E) \in \mathcal{S}_A^{(i)}, (V_B, V_E) \in \mathcal{S}_B^{(i)}, \\ & \quad \text{and } \mathsf{Nice}(V_A, V_B, V_E) \\ 0 & \text{otherwise} \end{cases}$$

Here Z is a normalization factor, and $\gamma(V_A, V_E) = 2^{-r} N^{-3w}$ where r is the length of the random tape in V_A and w is the number of random oracle queries (not including test queries) in $P_A \setminus P_E$. Note that by restricting to $\mathsf{Nice}(V_A, V_B, V_E)$, the positive test queries are taken into account by the definition of Γ, but the number of negative test queries in the views are not accounted for. But the probability of (V_A, V_B, V_E) in an actual distribution of any protocol, when restricted to the support of Γ, can be shown to be $\Gamma(V_A, V_B, V_E)(1 \pm \mathsf{negl})$.

Another major difference in our proof is the tidying up operation itself. Unlike in the random oracle case, we need to introduce a tidying up step even during the Eve round. In fact, this tidying up is done per query that Eve makes. Before each fresh query that Eve makes, we tidy up the distribution to ensure that at most one of Alice or Bob could have made that query previously. Further, after an Eve test query that is answered positively for which Eve does not have an explanation (i.e., none of the random oracle queries that Eve has made so far returned the image being tested), we remove the possibility that neither party has an explanation. (By tidying up before this query, we would have already required that *at most one party* had made that query previously; the current tidying up ensures that, exactly one party has an explanation for this query.)

Though this tidying up is carried per query that Eve makes, we ensure that the entire statistical difference incurred by the tidying up process during one round of Eve's execution is bounded in terms of the distance to the conditional product distribution at the start of this round. Indeed, this latter distance can only decrease through out Eve's round.

The tidying up procedure when Alice or Bob makes a query is similar to that in the case of the random oracle setting. It ensures that the tidied up distribution is close to a conditional product distribution, and the *additional* distance can be bounded in terms of the lightness threshold parameter for Eve, as before.

With these modifications, the resulting proof follows the outline mentioned above. At each round we tidy up the distribution over (V_A, V_B, V_E), by incurring a statistical difference related to the distance between the previous tidy distribution and a conditional product distribution. In turn, we bound the increase in the latter distance (during Alice's and Bob's rounds) in terms of the lightness guarantee by Eve.

As a direct consequence of Lemma 2, we can conclude that:

Corollary 2. *There is no key-agreement protocol relative to $\mathbb{O}_\kappa^{(\mathbb{K})}$, for any key set $\mathbb{K}$.*

5 Compiling out Decryption Queries

In this section we show that a family of PKE-enabling oracles is only as useful as a family of image testable random oracles, for semi-honest SFE. Combined with the result that this image testable random oracle family is useless for SFE, we derive the main result in this paper, that PKE is useless for semi-honest SFE.

As pointed out by [18], care must be taken in modeling such an oracle so that it does not allow oblivious transfer. In our case, we need to separate it from not just oblivious transfer but any non-trivial SFE.

In our proof we shall use the oracle $\mathbb{PKE}_\kappa$ defined in Section 2.3. This oracle facilitates public-key encryption (by padding messages with say $\kappa/2$ random bits before calling Enc), and hence key agreement. But, as mentioned before, by omitting the Dec oracle, the collection $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Test}_1, \mathsf{Test}_2)$ becomes an image-testable random oracle family $\mathbb{O}^{(\mathbb{K})}$. As we will see in Section 6, an image-testable random oracle is not useful for SFE or key agreement. The challenge is to show that even given the decryption oracle Dec, which does help with key-agreement, the oracle remains useless for SFE. [18] addressed this question for the special case of oblivious transfer, relying on properties that are absent from weaker (yet non-trivial) SFE functionalities. Our approach is to instead show that the decryption facility is completely useless in SFE, by giving a carefully compiled protocol whereby the parties help each other in finding decryptions of ciphertexts without accessing Dec oracle, while retaining the security against honest-but-curious adversaries. We show the following.

Theorem 2. *Suppose $\mathit{\Pi}$ is an N-round 2-party protocol with input domain $\mathcal{X} \times \mathcal{Y}$, that uses the oracle $\mathbb{PKE}_\kappa$. Then for any polynomial* poly*, there is an N-round protocol $\mathit{\Pi}^*$ using the oracle $\mathbb{O}_\kappa^{(\mathbb{K})}$ that is as secure as $\mathit{\Pi}$ against semi-honest adversaries, up to a security error of* $|\mathcal{X}||\mathcal{Y}|/\mathsf{poly}(\kappa)$.

Below we present the compiler used to prove this theorem, and sketch why it works. The full proof appears in the full version of the paper [32].

The Idea Behind the Compiler. For ease of presentation, we assume here that the oracles $\mathsf{Gen}(\cdot)$ and $\mathsf{Enc}(\cdot,\cdot)$ are injective (which is true, except with negligible probability, because they are length tripling random oracles). Conceptually the compiler is simple: each party keeps track of the ciphertexts that it *created* that the other party becomes "capable of" decrypting and sends the message in the ciphertext across at the right time. This will avoid the need for calling the decryption oracle. But we need to also argue that the compilation preserves security: if the original protocol was a secure protocol for some functionality, then so is the compiled protocol. To ensure this, a party, say Bob, should reveal the message in an encryption it created only if there is a high probability that Alice (or a curious adversary with access to Alice) *can* obtain that message by decryption. Further, the fact that Bob found out that Alice could decrypt a ciphertext should not compromise Bob's security. This requires that just based on common information between the two parties it should be possible to accurately determine which ciphertexts each party can possibly decrypt. This is complicated by the fact that the protocol can have idiosyncratic ways of transferring ciphertexts and public and private keys between the parties, and even if a party *could* carry out a decryption, it may choose to not extract the ciphertext or private key implicit in its view. By using the common information learner for image testable random oracles, it becomes possible to

Definition of the Compiler. Given a 2-party protocol $\mathit{\Pi}$, with input domains $\mathcal{X} \times \mathcal{Y}$, we define the compiled protocol $\mathit{\Pi}^*$ below. $\mathit{\Pi}$ has access to $\mathbb{PKE}_\kappa$, where as $\mathit{\Pi}^*$ will have access to the interface of $\mathbb{PKE}_\kappa$ consisting only of $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Test}_1, \mathsf{Test}_2)$ (or equivalently, to $\mathbb{O}_\kappa^{(\mathbb{K})}$ as described in Section 2). For convenience, we require a normal form for $\mathit{\Pi}$ that before making a decryption query $\mathsf{Dec}(sk, c)$ a party should make queries $\mathsf{Gen}(sk)$, $\mathsf{Test}_1(pk)$ and $\mathsf{Test}_2(pk, c)$ where pk was what was returned by $\mathsf{Gen}(sk)$.

We define $\mathit{\Pi}^*$ in terms of a 3-party protocol involving $\mathsf{Alice}_0, \mathsf{Bob}_0, \mathsf{Eve}$, over a broadcast channel. In the following we will define Alice_0 and Bob_0; this then defines an (inputless) system $\mathit{\Sigma}$ which consists of them interacting with each other internally, while interacting with an external party; in $\mathit{\Sigma}$, the inputs to Alice_0 and Bob_0 are picked uniformly at random. Then, Eve is defined to be $\mathcal{H}$ for the system Σ, as defined in Fig. 3: after each message from Σ (i.e., from Alice or Bob), Eve responds with a set of publicly computable queries to the oracle. Finally, Π^* is defined as follows: Alice runs Alice$_0$ and Eve internally, and Bob runs Bob$_0$ and Eve.[14]

So to complete the description of the compiled protocol, it remains to define the programs Alice$_0$ and Bob$_0$. We will define Alice$_0$; Bob$_0$ is defined symmetrically.

Alice$_0$ internally maintains the state of an execution of Alice's program in Π (denoted by Alice$_\Pi$). In addition, Alice$_0$ maintains a list L_A of entries of the form (m, pk, c), one for every call $\mathsf{Enc}(pk, m) = c$ that Alice$_\Pi$ has made so far, along with such triples from the (secondary) messages from Bob$_0$.
Corresponding to a single message m_i from Alice in Π, Alice$_0$ will send out two messages — a primary message m_i and a secondary message $C_{A,i}$ (with an intermediate message from Eve) — as follows. (For the sake of brevity we ignore the boundary cases $i = 1$ and $i - 1$ being the last message in the protocol; they are handled in a natural way.)
The list $L_{A,}$, before receiving the $i - 1^{\text{st}}$ message, is denoted by $L_{A,i-2}$.

- On receiving m_{i-1} and $C_{B,i-1}$ from Bob$_0$ (and the corresponding messages from Eve), first Alice sets $L_{A,i-1} := L_{A,i-2} \cup C_{B,i-1}$ (where $C_{B,i-1}$ is parsed as a set of entries of the form (m, pk, c)).
- Then Alice$_0$ passes on m_{i-1} to Alice$_\Pi$, and Alice$_\Pi$ is executed. During this execution Alice$_\Pi$ is given direct access to $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Test1}, \mathsf{Test2})$; but for every query of the form $\mathsf{Dec}(sk, c)$ from Alice$_\Pi$, Alice$_0$ obtains $pk = \mathsf{Enc}(sk)$ and checks if any entry in $L_{A,i-1}$ is of the form (m, pk, c) for some m. If it is, Alice$_0$ will respond to this query with m. Otherwise Alice$_0$ responds with $\perp$. At the end of this computation, the message output by Alice$_\Pi$ is sent out as m_i.
  Also Alice updates the list $L_{A,i-1}$ (which was defined above as $L_{A,i-2} \cup C_{B,i-1}$) to $L_{A,i}$ by including in it a tuple (m, pk, c) for each encryption query $\mathsf{Enc}(pk, m) = c$ that Alice$_\Pi$ made during the above execution.
- Next it reads a message from Eve. Let $T^{(i)}$ denote the entire transcript at this point (including messages sent by Alice$_0$, Bob$_0$ and Eve). Based on this transcript Alice$_0$ computes a set $D_{\mathrm{B}}^{T^{(i)}}$ of ciphertexts that Bob is "highly likely" to be able to decrypt in the next round, but has not encrypted itself,[a] and then creates a message $C_{A,i}$ that would help Bob decrypt all of them without querying the decryption oracle. The algorithm Assist_A used for this is detailed below in Fig. 5. Alice finishes her turn by sending out $C_{A,i}$.

[a] The threshold δ used in defining $D_{\mathrm{B}}^{T^{(i)}}$ by itself does not make it *highly likely* for the honest Bob to be able to decrypt a ciphertext. However, as we shall see, this will be sufficient for a *curious* Bob to be able to decrypt with high probability.

Fig. 4: Definition of Alice$_0$ procedure.

[14] Note that Eve follows a deterministic public-query strategy, and can be run by both parties. Alternately, in Π^*, one party alone could have run Eve. But letting both parties run Eve will allow us to preserve the number of rounds exactly, when consecutive messages from the same party are combined into a single message.

For each possible view V_B of Bob_{Π} at the point $T^{(i)}$ is generated let,

$$\begin{aligned} d_{\mathrm{B}}(V_B) := \{(pk, c) | \exists sk \text{ s.t. } [\mathsf{Gen}(sk) = pk], [\mathsf{Test}_1(pk) = 1], [\mathsf{Test}_2(pk, c) = 1] \in V_B \\ \text{and } \nexists m \text{ s.t. } [\mathsf{Enc}(pk, m) = c] \in V_B\}. \end{aligned}$$

We define the set

$$D_{\mathrm{B}}^{T^{(i)}} := \{(pk, c) | \Pr[(pk, c) \in d_{\mathrm{B}}(V_B) | T^{(i)}] > \delta\} \qquad (1)$$

where the probability is over a view V_B for Bob_{Π} sampled conditioned on $T^{(i)}$, in the interaction between Σ (i.e., Alice_0 and Bob_0 with a random input pair) and Eve.[a] The threshold δ which will be set to an appropriately small quantity (larger than, but polynomially related to, σ associated with Eve).

The message $C_{A,i}$ is a set computed as follows: for each $(pk, c) \in D_{\mathrm{B}}^{T^{(i)}}$, if there is an m such that $\mathsf{Enc}(pk, m) = c$ appears in $L_{A,i}$, then the triple (m, pk, c) is added to $C_{A,i}$. If for any (pk, c), if there is no such m, then the entire protocol is aborted.

[a] Even though we define Alice_0 in terms of a probability that is in terms of the behavior of a system involving Alice_0, we point out that this probability is well-defined. This is because the probability computed in this round refers only to the behavior of the system up till this round. Also Eve, which is also part of the system generating $T^{(i)}$, depends only on the prior messages from Alice_0.

Fig. 5: Procedure Assist_A for computing $C_{A,i}$.

Security of the Compiled Protocol. To formally argue the security of the compiled protocol we must show an honest-but-curious *simulator* with access to either party in an execution of Π, which can simulate the view of an honest-but-curious adversary in Π^*. Here we do allow a small (polynomially related to σ), but possibly non-negligible simulation error. We give a detailed analysis of such a simulation in the full version of the paper. Below we sketch some of the important arguments.

Firstly, it must be the case that the probability of Alice_0 aborting in Π^* while computing a secondary message $C_{A,i}$, is small. Suppose, with probability p Alice fails to find an encryption for some $(pk, c) \in D_{\mathrm{B}}^{T^{(i)}}$. Then, by the independence property guaranteed by Lemma 2, with about probability δp this Alice execution takes place in conjunction with a Bob view V_B such that $(pk, c) \in d_{\mathrm{B}}(V_B)$. This would mean that with close to probability δp we get an execution of the original protocol Π in which (pk, c) is present in the parties' views, but neither Alice nor Bob created this ciphertext. This probability must then be negligible.

The more interesting part is to show that it is safe to reveal an encrypted message, when there is only a small (but inverse polynomial) probability that the other party would have decrypted it. For concreteness, consider when the honest-but-curious adversary has access to Alice. In the execution of Π^* it sees the messages $C_{B,i}$ that are sent by Bob (assuming Bob does not abort). These contain the messages for each (pk, c) pair in D_{A}^{T} where T is the common information so far. So we need to show that the simulator would be able to extract all these messages as well. Consider a $(pk, c) \in D_{\mathrm{A}}^{T}$. If Alice's view contains an Enc query that generates c, or a Gen query that generates pk, then the simulator can use this to extract the encrypted message. Otherwise it samples a view A' for Alice consistent with T, but conditioned on $(pk, c) \in d_{\mathrm{A}}(A')$ (such A' must exist since $(pk, c) \in D_{\mathrm{A}}^{T}$). Then A' does contain a secret key sk' for pk that Alice will use to decrypt the ciphertext.

However, note that the view of the oracle in A' need not be consistent with the given oracle. Thus it may not appear meaningful to use sk' as a secret key. But intuitively, if it is the case that with significant probability Alice did not generate pk herself, then it must have been generated by Bob, and then the only way Alice could have carried out the decryption is by extracting Bob's secret key from the common information. Thus this secret key is fixed by the common information. Further, by sampling an Alice view in which a secret key for pk occurs, this secret key must, with high probability agree with the unique secret key implicit in the common information. Formalizing this intuition heavily relies on the independence characterization: otherwise the common information need not fix the secret key, even if it fixes the public key.

In the full version of this paper we give a detailed proof of security of Π^*, by defining a complete simulation, and using a coupled execution to analyze how good the simulation is. We show that the compiled protocol is as secure as the original protocol up to a security error of $O|\mathcal{X}||\mathcal{Y}|(N(\sigma/\delta + \delta)) = O(1/\mathsf{poly})$ by choosing appropriate parameters, assuming $|\mathcal{X}||\mathcal{Y}|$ is polynomial.

The proof relies on Lemma 2. It shows that even when the protocol allows the parties to use the information from the common information learner, it holds that the views of the two parties (in an inputless version of the protocol considered in the proof) are nearly independent of each other's, conditioned on the common information gathered by Eve.

6 Limits of Image-testable Random Oracles

Applying the compiler from the above section, we can convert a protocol using the $\mathbb{PKE}$ oracle to one using an image-testable random oracle $\mathbb{O}^{(\mathbb{K})}$. Then, to complete the proof of Theorem 1 it will suffice to prove the following result, which asserts that no protocol ρ using $\mathbb{O}^{(\mathbb{K})}$ can be a secure realization of f, if f is semi-honest non-trivial.

Lemma 3. *Suppose ρ is a $1-\lambda(\kappa)$ semi-honest secure protocol (with round complexity N) for 2-party finite semi-honest non-trivial f relative to oracle class $\mathbb{O}_{\kappa}^{(\mathbb{K})}$, for any key set $\mathbb{K}$. There exists $\Lambda = 1/\mathsf{poly}(\cdot)$ such that, for infinitely many κ, we have $\lambda(\kappa) > \Lambda(N, \kappa)$.*

The proof of this lemma follows the proof of a similar result of [5], which considered the class of random oracles instead of image-testable random-oracles. The proof in [5] uses a detailed frontier analysis, in which the following two properties of the random oracles (informally stated here) were used, which we extend to the case of image-testable random oracles.

1. Local Samplability: We need Bob to sample hypothetical Bob view V_B', based on his actual view V_B, but without exactly knowing what view V_A Alice has. A crucial

step in this is to sample a new query-answer set P'_B which is consistent with P_E; and this sampling has to be independent of the exact query-answer set P_A of Alice.

2. Oblivious Re-randomizability: Once Bob has sampled a hypothetical Bob view, it needs to simulate the view further (for just one round, before the next message from Alice arrives). This simulation includes answering further queries to the (hypothetical) oracle. A crucial step in this is to answer these new queries with answers which are consistent with Alice's query-answer set P_A, but are otherwise independent of P_B. That is, in simulating answers to further oracle queries, Bob should *rerandomize* the part of the oracle which is consistent with the actual Bob query-answer pairs P_B.

Local samplability is a direct consequence of Lemma 2, which characterized the views in the actual execution of the prototocl as close to a conditional product distribution. For proving the oblivious rerandomization property, we need to specify the rerandomization procedure. Such a procedure for the case of random oracles was provided in [5]. In Fig. 6 we extend this to the case of image-testable random oracles.

Suppose Alice has private query-answer sequence P_A, Eve has P_E and Bob has P_B. Assume that Bob has been provided with P'_B; and $\mathsf{Typical}(P_A \cup P \cup P_E)$ and $\mathsf{Good}(P_A, P, P_E)$ hold, for $P \in \{P_B, P'_B\}$.
Let D be the set of R-queries in P_B which are not already included in $\mathcal{Q}(P_E \cup P'_B)$.
We re-emphasize that the queries in $\mathcal{Q}(P_B) \cap \mathcal{Q}(P'_B)$ outside $\mathcal{Q}(P_E)$ could possible by inconsistently answered.
Initialize a global set $R_{\mathsf{local}} = \emptyset$.
$\mathsf{Query\text{-}Answering}$ (q) :

1. If q is answered in $P_E \cup P'_B$ use that answer.
2. If $q = \langle k, \alpha \rangle$ is a new R-query and $q \in D$, answer with $a \xleftarrow{\$} \{0,1\}^{3\kappa}$. Add $\langle k, a \rangle$ to R_{local}.
3. If q is a T-query which is already in R_{local} then answer 1.
4. Otherwise (i.e. if the conditions above are not met) forward the query to the actual oracle and obtain the answer a.

Fig. 6: Bob's algorithm to answer future queries using re-randomization (oblivious to P_A).

We need to show that the result of the simulated execution using the rerandomized oracle is close to that of an actual execution with the hypothetical view V'_B that was sampled. Formally, the analysis requires the following "safety" property to hold with high probability, when Bob samples V'_B at a point when the views are (V_A, V_B, V_E).

Definition 5 (Safety). *For Alice view V_A, pair of Bob views (V_B, V'_B) and Eve view V_E, we define the following predicate:*

$$\mathsf{Safety}(V_A, (V_B, V'_B), V_E) := \mathsf{Nice}(V_A, V_B, V_E) \ \wedge \ \mathsf{Nice}(V_A, V'_B, V_E).$$

In the full version we show that at all rounds of the protocol, the safety condition is satisfied with high probability. The proof, again, is a consequence of the fact that the actual distributions are close to tidy distributions, and the tidy distributions are close to conditional product distributions.

6.1 Extending to Protocols with Inputs

The final ingredient we need to extend the proof in [5] to the case of image-testable random oracles is to extend our common information learner to protocols with private inputs for Alice and Bob. As was done in [5], such a common information learner can be easily reduced to one for inputless protocols, *as long as the domain size of the inputs is polynomial in the security parameter*.[15] For this we transform the given protocol to an inputless protocol by randomly sampling inputs for Alice and Bob. We augment the protocol ρ where parties have private inputs with an eavesdropper strategy as guaranteed by Lemma 2 when we assume that parties have picked their input uniformly at random. This augmented protocol is referred to as ρ^+.

By choosing the threshold parameter σ of the eavesdropper to be suitably small, we can ensure the following strong independence properties:

1. Suppose i is an even round in the augmented protocol ρ^+. If $x \in \mathcal{X}$ and $y, y' \in \mathcal{Y}$ are likely inputs at $V_E^{(i)}$ (transcript of the augmented protocol), then the message sent by Alice is nearly independent of Bob's private input being y or y'.
2. Suppose i is an even round in the augmented protocol ρ^+. If $x \in \mathcal{X}$ and $y, y' \in \mathcal{Y}$ are likely inputs at $V_E^{(i)}$, then sample a Alice-Bob joint view $(V_A^{(i+1)}, V_B^{(i)})$ just after Alice has sent the message m_{i+1}. Conditioned on the transcript $V_E^{(i)}$, message sent by Alice m_{i+1} and Bob input being y', sample a new Bob view $V'^{(i)}_B$. With high probability: $\mathsf{Safety}(V_A^{(i+1)}, (V_B^{(i)}, V'^{(i)}_B), V_E^{(i)})$ holds, i.e. $\mathsf{Nice}(V_A^{(i+1)}, V_B^{(i)}, V_E^{(i)})$ and $\mathsf{Nice}(V_A^{(i+1)}, V'^{(i)}_B, V_E^{(i)})$.
3. For an even round i, and likely inputs $x \in \mathcal{X}$ and $y, y' \in \mathcal{Y}$ the distribution of $(V_A^{(i+1)}, V_B^{(i)} V'^{(i)}_B)$ is close to a product distribution where each component is independently sampled conditioned on $(V_E^{(i)}, m_{i+1})$.

Analogous conditions hold when i is odd. For a formal version of this result, refer to the full version of the paper [32].

Given the above results, the frontier analysis of [5] can be carried out for image-testable random oracles.

7 Putting Things Together

Now we show how to complete the proof of Theorem 1. Suppose f is a 2-party finite semi-honest non-trivial SFE. Assume that there exists a $1 - \mathsf{negl}(\kappa)$ secure protocol ρ relative to oracle class $\mathbb{PKE}_\kappa$, with round complexity N. By Theorem 2, we construct a

[15] Note that we depend on the input domains being of polynomial size also for applying the decomposability characterization of [8,10].

$1-\lambda^*(\kappa)$ secure protocol ρ^* relative to oracle class $\mathbb{O}_\kappa^{(\mathbb{K})}$, where λ^* could be arbitrarily small $1/\mathsf{poly}$ and $\mathbb{K} = \{0,1\}^{2\kappa} \cup \{\perp\}$.

Now, if we choose λ^* in Lemma 3 to be sufficiently small so that $\lambda^*(\kappa) < \Lambda(N, \kappa)$, ρ^* contradicts Lemma 3 and hence also the assumption that ρ is a $(1 - \mathsf{negl}(\kappa))$ secure protocol for semi-honest non-trivial f relative to $\mathbb{PKE}_\kappa$. Note that this result crucially relies on the fact that Theorem 2 preserves round-complexity and the simulation error exhibited in Lemma 3 is function of only round complexity (and independent of the query complexity).

8 Conclusions and Open Problems

As mentioned in the introduction, our result can be set in the larger context of the "cryptographic complexity" theory of [7]: with every (finite, deterministic) multi-party function f, one can associate a computational intractability assumption that there exists a secure computation protocol for f that is secure against semi-honest corruption. The main result of this work shows that the set of such assumptions associated with 3-party functions is strictly larger than the set associated with 2-party functions. However, *we do not characterize this set either for the 3-party case or for the 2-party case.*

It remains a major open problem in this area to understand what all computational intractability assumptions could be associated with multi-party functions. For the 3-party case, this question is far less understood than that for 2-party functions. Intuitively, there are many more "modes of secrecy" when more than two parties are involved, and these modes will be associated with a finer gradation of intractability assumptions. Our result could be seen as a first step in understanding such a finer gradation. It raises the question whether there are further modes of secrecy for larger number of parties, and if they always lead to "new" complexity assumptions.

Stepping further back, the bigger picture involves randomized and reactive functionalities, various different notions of security, and "hybrid models" (i.e., instead of considering each multi-party function f and a secure protocol for it in plain model, we can consider a pair of functions (f, g) and consider a secure protocol for f given ideal access to g). The cryptographic complexity questions of such functions remain wide open.

References

1. Impagliazzo, R., Rudich, S.: Limits on the provable consequences of one-way permutations. In Johnson, D.S., ed.: STOC, ACM (1989) 44–61
2. Barak, B., Mahmoody, M.: Merkle puzzles are optimal - an $O(n^2)$-query attack on any key exchange from a random oracle. In Halevi, S., ed.: CRYPTO. Volume 5677 of Lecture Notes in Computer Science., Springer (2009) 374–390
3. Haitner, I., Omri, E., Zarosim, H.: Limits on the usefulness of random oracles. Theory of Cryptography Conference (TCC, to appear) (2013)
4. Dachman-Soled, D., Lindell, Y., Mahmoody, M., Malkin, T.: On black-box complexity of optimally-fair coin-tossing. In: Theory of Cryptography Conference - TCC 2011. (2011)
5. Mahmoody, M., Maji, H.K., Prabhakaran, M.: Limits of random oracles in secure computation. CoRR **abs/1205.3554** (2012) To appear in ITCS 2014.

6. Reingold, O., Trevisan, L., Vadhan, S.P.: Notions of reducibility between cryptographic primitives. In Naor, M., ed.: TCC. Volume 2951 of Lecture Notes in Computer Science., Springer (2004) 1–20
7. Maji, H.K., Prabhakaran, M., Rosulek, M.: Cryptographic complexity classes and computational intractability assumptions. In Yao, A.C.C., ed.: ICS, Tsinghua University Press (2010) 266–289
8. Kushilevitz, E.: Privacy and communication complexity. [33] 416–421
9. Beaver, D.: Perfect privacy for two-party protocols. In Feigenbaum, J., Merritt, M., eds.: Proceedings of DIMACS Workshop on Distributed Computing and Cryptography. Volume 2., American Mathematical Society (1989) 65–77
10. Maji, H.K., Prabhakaran, M., Rosulek, M.: Complexity of multi-party computation problems: The case of 2-party symmetric secure function evaluation. In Reingold, O., ed.: TCC. Volume 5444 of Lecture Notes in Computer Science., Springer (2009) 256–273 Full version available from IACR Eprint Archive: http://eprint.iacr.org.
11. Maji, H.K., Prabhakaran, M., Rosulek, M.: A unified characterization of completeness and triviality for secure function evaluation. In Galbraith, S.D., Nandi, M., eds.: INDOCRYPT. Volume 7668 of Lecture Notes in Computer Science., Springer (2012) 40–59
12. Haitner, I., Omri, E., Zarosim, H.: On the power of random oracles. Electronic Colloquium on Computational Complexity (ECCC) **19** (2012) 129
13. Simon, D.R.: Finding collisions on a one-way street: Can secure hash functions be based on general assumptions? In: EUROCRYPT. (1998) 334–345
14. Gertner, Y., Malkin, T., Reingold, O.: On the impossibility of basing trapdoor functions on trapdoor predicates. In: FOCS. (2001) 126–135
15. Boneh, D., Papakonstantinou, P.A., Rackoff, C., Vahlis, Y., Waters, B.: On the impossibility of basing identity based encryption on trapdoor permutations. In: FOCS, IEEE Computer Society (2008) 283–292
16. Katz, J., Schröder, D., Yerukhimovich, A.: Impossibility of blind signatures from one-way permutations. [34] 615–629
17. Matsuda, T., Matsuura, K.: On black-box separations among injective one-way functions. [34] 597–614
18. Gertner, Y., Kannan, S., Malkin, T., Reingold, O., Viswanathan, M.: The relationship between public key encryption and oblivious transfer. In: FOCS, IEEE Computer Society (2000) 325–335
19. Kim, J.H., Simon, D.R., Tetali, P.: Limits on the efficiency of one-way permutation-based hash functions. In: FOCS. (1999) 535–542
20. Gennaro, R., Gertner, Y., Katz, J., Trevisan, L.: Bounds on the efficiency of generic cryptographic constructions. SIAM J. Comput. **35**(1) (2005) 217–246
21. Lin, H.C., Trevisan, L., Wee, H.: On hardness amplification of one-way functions. In Kilian, J., ed.: TCC. Volume 3378 of Lecture Notes in Computer Science., Springer (2005) 34–49
22. Haitner, I., Hoch, J.J., Reingold, O., Segev, G.: Finding collisions in interactive protocols - a tight lower bound on the round complexity of statistically-hiding commitments. In: FOCS, IEEE Computer Society (2007) 669–679
23. Barak, B., Mahmoody, M.: Lower bounds on signatures from symmetric primitives. In: FOCS: IEEE Symposium on Foundations of Computer Science (FOCS). (2007)
24. Impagliazzo, R., Luby, M.: One-way functions are essential for complexity based cryptography (extended abstract). [33] 230–235
25. Ostrovsky, R.: One-way functions, hard on average problems, and statistical zero-knowledge proofs. In: Structure in Complexity Theory Conference. (1991) 133–138
26. Ostrovsky, R., Wigderson, A.: One-way functions are essential for non-trivial zero-knowledge. Technical Report TR-93-073, International Computer Science Institute, Berke-

ley, CA (November 1993) Preliminary version in Proc. 2nd Israeli Symp. on Theory of Computing and Systems, 1993, pp. 3–17.

27. Haitner, I.: Semi-honest to malicious oblivious transfer - the black-box way. In: TCC. (2008) 412–426
28. Haitner, I., Nguyen, M.H., Ong, S.J., Reingold, O., Vadhan, S.P.: Statistically hiding commitments and statistical zero-knowledge arguments from any one-way function. SIAM J. Comput. **39**(3) (2009) 1153–1218
29. Impagliazzo, R.: A personal view of average-case complexity. In: Structure in Complexity Theory Conference. (1995) 134–147
30. Lindell, Y., Omri, E., Zarosim, H.: Completeness for symmetric two-party functionalities - revisited. In Wang, X., Sako, K., eds.: ASIACRYPT. Volume 7658 of Lecture Notes in Computer Science., Springer (2012) 116–133
31. Haitner, I. personal communication (Jan. 21, 2013)
32. Mahmoody, M., Maji, H.K., Prabhakaran, M.: On the power of public-key encryption in secure computation. Electronic Colloquium on Computational Complexity (ECCC) **20** (2013) 137
33. 30th Annual Symposium on Foundations of Computer Science, 30 October-1 November 1989, Research Triangle Park, North Carolina, USA. In: FOCS, IEEE (1989)
34. Ishai, Y., ed.: Theory of Cryptography - 8th Theory of Cryptography Conference, TCC 2011, Providence, RI, USA, March 28-30, 2011. Proceedings. In Ishai, Y., ed.: TCC. Volume 6597 of Lecture Notes in Computer Science., Springer (2011)