Limits on the Power of Garbling Techniques for Public-Key Encryption

Sanjam Garg[1⋆], Mohammad Hajiabadi[1,2⋆⋆], Mohammad Mahmoody[2⋆⋆⋆], and Ameer Mohammed[2†]

[1] University of California, Berkeley
[2] University of Virginia

Abstract. Understanding whether public-key encryption can be based on one-way functions is a fundamental open problem in cryptography. The seminal work of Impagliazzo and Rudich [STOC'89] shows that black-box constructions of public-key encryption from one-way functions are impossible. However, this impossibility result leaves open the possibility of using non-black-box techniques for achieving this goal.

One of the most powerful classes of non-black-box techniques, which can be based on one-way functions (OWFs) alone, is Yao's garbled circuit technique [FOCS'86]. As for the non-black-box power of this technique, the recent work of Döttling and Garg [CRYPTO'17] shows that the use of garbling allows us to circumvent known black-box barriers in the context of identity-based encryption.

We prove that garbling of circuits that have OWF (or even random oracle) gates in them are insufficient for obtaining public-key encryption. Additionally, we show that this model also captures (non-interactive) zero-knowledge proofs for relations with OWF gates. This indicates that currently known OWF-based non-black-box techniques are perhaps insufficient for realizing public-key encryption.

1 Introduction

Public-key encryption (PKE) [15,33] is a fundamental primitive in cryptography and understanding what assumptions are sufficient for realizing it is a foundational goal. Decades of research have provided us with numerous constructions of

⋆ Research supported in part from DARPA/ARL SAFEWARE Award W911NF15C0210, AFOSR Award FA9550-15-1-0274, AFOSR YIP Award, DARPA and SPAWAR under contract N66001-15-C-4065, a Hellman Award and research grants by the Okawa Foundation, Visa Inc., and Center for Long-Term Cybersecurity (CLTC, UC Berkeley). The views expressed are those of the author and do not reflect the official policy or position of the funding agencies.

⋆⋆ Supported by NSF award CCF-1350939 and AFOSR Award FA9550-15-1-0274.

⋆⋆⋆ Supported by NSF CAREER award CCF-1350939, a subcontract on AFOSR Award FA9550-15-1-0274, and University of Virginia's SEAS Research Innovation Award.

† Supported by Kuwait University and the Kuwait Foundation for the Advancement of Science. Work done while at University of Virginia and visiting University of California, Berkeley.

PKE from a variety of assumptions; see a recent survey by Barak [5]. However, all known constructions of PKE require computational assumptions that rely on rich structure and are stronger than what is necessary and sufficient for *private*-key cryptography, namely the mere existence of one-way functions (OWF). The seminal work of Impagliazzo and Rudich [23] provides some evidence that this gap between the assumption complexity of private-key and public-key encryption may be inherent. In particular, the work of [23] shows that there is no *black-box* construction of PKE from OWFs.[3]

When studying the *impossibility* of basing PKE on OWFs, focusing on a class of constructions (e.g., black-box constructions as in [23]) is indeed necessary. The reason is that to rule out "OWFs implying PKE" in a *logical* sense, we have to first prove the existence of OWFs unconditionally (thus, proving $\mathbf{P} \neq \mathbf{NP}$) and then rule out the existence of PKE altogether (thus breaking all assumptions under which PKE exists). That is why this line of separation results focuses on ruling out the possibility of using certain *techniques* or generic *proof methods* (here black-box techniques) as possible natural paths from OWFs to PKE.

Garbled circuits. Over the past few decades, garbling techniques [35,25,9,1] (or randomized encodings [24] more generally) have been extensively used to build many cryptographic schemes. Roughly speaking, in a circuit garbling mechanism, a PPT encoder $\mathsf{Garb}(\mathsf{C})$ takes a circuit C as input, and outputs a *garbled circuit* $\widetilde{\mathsf{C}}$ and a set of *input labels* $\{\mathsf{label}_{i,b}\}_{i\in[m],b\in\{0,1\}}$ where m is the number of input wires of C. Using another algorithm $\mathsf{Eval}(\cdot)$, one can use the garbled circuit $\widetilde{\mathsf{C}}$ and input labels $\{\mathsf{label}_{i,x_i}\}_{i\in[m]}$ for an input $x = (x_1, ..., x_m)$, to compute $\mathsf{C}(x)$ without learning any other information. Note that if the original circuit C needs to run a cryptographic primitive f internally (e.g., a circuit C for a pseudorandom generator built from a OWF f), this use of garbling leads to a *non-black-box* construction. This is because the algorithm Garb needs to work with an actual circuit description of C, whose circuit description is in turn obtained by the circuit description of f, hence making non-black-box use of f.

Garbling, as a primitive, may itself be realized using one-way functions [35,25]. This puts forward the intriguing possibility of basing PKE solely on OWFs by making *black-box* use of garbling mechanisms over circuits that can run the one-way function. As stated above, such constructions will make non-black-box use of the underlying OWF (caused by garbling circuits that run the OWF internally) and hence the impossibility result of Impagliazzo and Rudich [23] has no bearing on such potential constructions. In fact, such non-black-box garbling techniques, combined with the Computational Diffie-Hellman assumption, have recently been used by Döttling and Garg [16] to circumvent black-box impossibility results [11,29] in the context of identity-based encryption (IBE). Thus, it is natural to ask:

Can non-black-box garbling techniques be used to realize PKE from OWFs?

[3] A (fully) black-box construction is one that treats the OWF as an oracle, and the security proof uses the OWF and the adversary both as oracles; see the surveys of [32,4] for formal definitions.

Our model. We study the above question in the model of Brakerski, Katz, Segev and Yerukhimovich [12] (see also follow up works [2,3,10]) which gives a general way of capturing non-black-box techniques via circuits with cryptographic gates (e.g., OWF gates). More formally, we will model the above-stated garbling-based non-black-box use of one-way functions as black-box use of garbling mechanisms that can take as input circuits C with one-way function (or even random oracle) gates planted in them. Such constructions are indeed *non-black-box* according to the taxonomy of [32] if viewed as standalone constructions solely based on the OWF itself. We stress that the allowed access to the garbling mechanism itself is black-box; the non-black-box feature arises from the fact that circuits with OWF gates may now be garbled.

A more sophisticated scenario is when the circuits being garbled have *garbling* gates, in addition to OWF gates, planted in them. We do not, however, consider such a recursive scenario and we leave it to future work. It is crucial to note that, to the best of our knowledge, all known constructions that make use of garbling schemes together with one-way functions (e.g., [8,26,17]) fall into our model, and thus, understanding the limitations of such techniques towards obtaining PKE is impactful.

1.1 Our Result

In this work, we show that black-box use of garbling mechanisms that allow circuits with OWF gates to be garbled is not sufficient for constructing PKE. More precisely, we prove the following.

Theorem 1 (Main result – informally stated). *There exists no black-box construction of public-key encryption schemes from any one-way function (or even a random oracle) f together with a garbling mechanism that can garble oracle-aided circuits with f-gates embedded in them.*

Comparison with prior work: Impossibility from weaker garbling. The work of Asharov and Segev [2] showed that secret-key functional encryption with one-way function gates cannot be used (as a black-box) to obtain public-key encryption (or even key agreement). This result implies that for the special case of such weaker garbling schemes, called *non-decomposable* garbling, where the entire input is considered as a single unit (rather that as bit-by-bit input labels) is insufficient for realizing PKE.

On the other hand, throughout this work we use garbling to refer to a notion that supports bit-by-bit input labels, a notion that Bellare, Hoang and Rogaway [9] refer to as *projective garbling* (a.k.a. decomposable garbling). Under projective garbling, for a circuit C of input size m, one generates two garbled label $\{\mathsf{label}_{i,b}\}_{i\in[m],b\in\{0,1\}}$ for the ith input wire of the circuit. An important property enabled by this bit-by-bit garbling is the decomposability property: one can pick a garbled label for each input wire to form a garbled input for a long string. In

contrast, under non-decomposable garbling, for each input X to the circuit, one independently generates a corresponding garbled input $\widetilde{X}$. As a result, different strings have independent garbled inputs.

We note that projective garbling is crucial for many applications of garbling. For example, even the most basic application of garbling in two party secure computation based on oblivious transfer uses the projective property. We refer the reader to [9, Figure 3] for a detailed list of applications that require projective garbling. As a recent example, we note that the IBE construction of Döttling and Garg [16] (that circumvents a black-box impossibility result of Papakonstantinou et al. [29] using garbling) uses projective garbling crucially. Specifically, in [DG17] the encryptor provides a sequence of garbled circuits with no knowledge of what input each of those garbled circuits are later evaluated on by the decryptor. This *input-obliviousness* property is enabled by the encryptor sending all the bit-by-bit garbled labels in some encrypted form to the decryptor. Later, the decryptor can open exactly one garbled label for each input wire, hence obtaining a garbled input for the whole string. This input-obliviousness technique cannot be enabled using non-decomposable garbling. This is because a whole garbled input cannot be formed there by putting together smaller pieces. As a result, the party who generates a garbled circuit must be aware of the input on which this garbled circuit is to be evaluated on, in order for him to be able to provide the corresponding garbled input.

1.2 Extensions

Extension to key agreements. Our proof extends to rule out any black-box construction of *constant-round* key-agreement protocols from OWFs and garbling schemes for oracle-aided circuits. However, the proof of the separation for key-agreement beyond the case of two message protocols (which are equivalent to PKE) becomes much more involved. Therefore, for clarity of the presentation, and because the most interesting special case of constant-round key-agreement protocols happens to be PKE itself, in this presentation, we focus on the case of separation for PKE. See Section the full version for more details.

Resolving an open question of [12]. The work of [12] proved non-black-box limitations for one-way functions when used as part of zero knowledge (ZK) proofs for relations with one-way function gates. They showed that key-agreement protocols with *perfect* completeness cannot be realized in a 'black-box' way from oracles that provide a one-way function f together with ZK proofs of satisfiability for f-aided circuits. They left ruling out the possibility of protocols with imperfect (e.g., negligible) completeness as an open problem, as their techniques indeed crucially relied on the perfect completeness assumption. We demonstrate the power of our new techniques in this work by resolving the open problem of [12] along the way, for the case of PKE schemes (or even constant-round key-agreement schemes). In particular, in the full version of the paper, we observe that the oracles we use for proving our separations for the case of

garbling, indeed imply the existence of NIZK proofs for satisfiability of circuits with OWF-gates. The extension of the result of [12] explained above then follows from the above observation.

1.3 Related Work and Future Directions

There are quite a few results that prove limitations for a *broad* class of non-black-box techniques [30,31,20], so long as the security reduction is black-box. In other words, these results are proved against basing certain primitives on any *falsifiable* assumption. However, when it comes to the case of non-black-box constructions of PKE from OWFs, no such general separations are known (and proving such results might in fact be impossible).

As described earlier, the works of [12,2] proved limitations of certain non-black-box constructions of PKE from OWFs. This is indeed the direction pursued in this work. The work of Dachman-Soled [14] takes yet another path, showing that certain non-black-box uses of one-way functions *in the security proof* are incapable of obtaining PKE from OWFs.

We note that we only consider a setting in which circuits with random oracle gates are garbled. We do not allow garbling of circuits which themselves include garbling gates. Such techniques are captured by the so called monolithic model of Garg, Mahmoody, and Mohammed [18,19]. We leave open the problem of ruling out such constructions.

Finally, as noted above, the extension of our results to the key-agreement setting (discussed in the full version) only cover the *constant-round* case. The reason is that, during the proof of our main result, we modify the protocol iteratively, once for each round, which increases the parameters of the protocol by a polynomial factor each time. We leave the extension to general polynomial-round protocols as an interesting future direction.

Organization. In Section 2 we give an overview of our approach and techniques. In Section 3 we give some definitions and basic lemmas. In Section 4 we will go over the proof steps of our main impossibility result. See the full version of the paper for full proofs of the main result and the extensions.

2 Technical Overview

For brevity, we refer to the primitive of a one-way function f and garbling circuits with f gates as GC-OWF. As usual in black-box separation results, we will prove our main theorem by providing an oracle O relative to which secure GC-OWF exists, but secure PKE does not.

2.1 Big Picture: Reducing the Problem to the Result of [23]

At a very high level, our approach is to reduce our problem to the result of [23]. Namely, we aim to show that one can always modify the PKE construction that

is based on the GC-OWF oracle O into a new one that is almost as secure, but which no longer uses the garbling part of the oracle O. In other words, we modify the construction so that it becomes a construction from an OWF oracle alone. Our main result, then, follows from the impossibility result of [23] which rules out the possibility of getting PKE from one-way (or even random) functions. We call this process 'compiling out the garbling part' from the PKE construction.

As a technical remark, our transformation does not result in a normal black-box construction of PKE from OWFs, but rather results in an *inefficient* one which nonetheless makes a polynomial number of queries to the OWF oracle. The key point is that the *proof* of the work of Impagliazzo and Rudich [23] allows us to break any such (even inefficient, but still polynomial-query) constructions of PKE in the random oracle model using a polynomial number of queries during the attack. Our actual result follows by combining our compilation result with the result of [23], to get a polynomial query attack against the security of the original PKE. This will be sufficient for a black-box separation.

At a high level, our approach also bears similarities to recent impossibility results for indistinguishability obfuscation [18] as we also compile out the more powerful (and structured) parts of the oracle, ending up with a scheme that uses a much simpler oracle, for which an impossibility is known. However, there is a subtle distinction here. Unlike the results of [13,28,27,18], when we compile out the garbling-related queries from the PKE construction, we end up with an inefficient scheme that potentially runs in exponential time but nevertheless makes a polynomial number of queries. However, as mentioned above, this is fine for deriving our separation, because we can still rely on the fact that the result of [23] does something stronger and handles inefficient constructions as well.

2.2 Our Separating Idealized GC-OWF Oracle

In this subsection, we will first describe our oracle O that gives an intuitive way of obtaining GC-OWFs. The natural first version of this oracle is too strong as it also implies virtual black-box (VBB) obfuscation. We will then add a careful weakening subroutine to this oracle O to prevent it from implying obfuscation. In the next subsection we describe the ideas behind how to compile out the garbling-related subroutines of O from the PKE construction, while keeping the PKE construction "secure".

Our 1st oracle for GC-OWF. Our first version of the separating oracle $O = (f, L^f)$ will consist of a random oracle f (giving the OWF part) as well a garbling part $L^f = (\mathsf{gc}, \mathsf{eval}^f)$ with two subroutines. The encoding/garbling subroutine $\mathsf{gc}(s, \mathsf{C})$ is simply a random (injective) function that takes a seed s and a circuit C and maps them into a garbled circuit $\widetilde{\mathsf{C}}$ as well as labels $\{\mathsf{label}_{i,b}\}_{i\in[n],b\in\{0,1\}}$ for the input wires of C where n is the number of input wires in C.[4] The evaluation subroutine eval^f takes as input a garbled circuit $\widetilde{\mathsf{C}}$

[4] In the main body, we will use two separate subroutines gc, gi for encoding circuits vs input labels, but for brevity here we combine them into one subroutine.

as well as a vector of input labels $\widetilde{X} = (\widetilde{x}_1 \cdots \widetilde{x}_n)$ and *only if* they were all encoded using the *same* seed s, eval^f returns the right output $\mathsf{C}^f(x_1, \ldots, x_n)$. Note that we include f in the representation of eval^f but not in that of gc; the reason is gc is simply a random oracle (independent of f), while eval^f needs to call f in order to compute $\mathsf{C}^f(x_1, \ldots, x_n)$.

Adding the weakening subroutine rev. It is easy to see that this first version of the oracle O as described above can realize a secure GC-OWF, but it can do much more than that! In fact this oracle implies even VBB obfuscation of circuits with f gates (which in turn *does* imply PKE [34]). We will, therefore, weaken the power of the oracle O later on by adding an extra subroutine to it (which we will call rev), which roughly speaking allows an attacker to break the garbling scheme if she has access to two labels for the same wire. We will describe this subroutine after it becomes clear how it will be useful for our main goal of compiling out the garbling aspect of O. Also note that, since we are defining our oracle *after* the (supposed) construction of PKE from GC-OWF is fixed, without loss of generality, the PKE construction from GC-OWF does *not* call the extra subroutine rev. This separation technique was also used before in the work of [21] and is reminiscent of the "two-oracle" approach of [22].

2.3 Compiling Out the Garbling Power of O from the Construction

Suppose $\mathcal{E}^O = (G^O, E^O, D^O)$ is a fully black-box construction of PKE using the oracle O described above. Our goal here is to 'reduce' our problem (of breaking $\mathcal{E}^O$ using a polynomial number of queries) to the result of [23] by compiling out the 'garbling power' of the oracle O from the scheme $\mathcal{E}^O$. But what subroutines do we have to compile out from $\mathcal{E}$? As it turns out, we do not have to eliminate both gc and eval^f subroutines; removing only eval^f queries will suffice.

Compiling out eval^f queries from PKE constructions $\mathcal{E}^O$. If we make sure that (the modified but "equally"-secure version of) $\mathcal{E}$ does not make any calls to the eval^f subroutine of the oracle O, it would be sufficient for our purposes, because the oracle $O' = (f, \mathsf{gc})$ is just a random oracle, and by the result of [23] we know that such oracle is not enough for getting PKE in a black-box way. Therefore, in what follows, our goal reduces to (solely) removing the eval^f queries from PKE constructions $\mathcal{E}^O$ in a way that we can argue the new construction is 'as secure' as the original one.

In order to make our proof more modular, we compile out eval^f queries from the different components (i.e., key-generation G, encryption E, and decryption D) of the construction $\mathcal{E}^O = (G^O, E^O, D^O)$ one at a time. First, we may easily see that G^O does not need to call eval^f at all. This is because G^O is the first algorithm to get executed in the system, and so G^O knows all the generated garbled circuits/labels. Therefore, G^O can, instead of calling eval^f queries, simply

run $\mathsf{C}^f(X)$ on its own by further calls to f.[5] Now, we proceed to compile out eval^f queries from the remaining two subroutines E and D in two steps. In each step, we assume that we are starting off with a construction that has no eval^f queries in some of its subroutines, and then we modify the construction to remove eval^f queries from the next subroutine.

- **Step 1:** Starting with $\mathcal{E} = (G^{f,\mathsf{gc}}, E^O, D^O)$, we will compile $\mathcal{E}$ into a new scheme $\dot{\mathcal{E}} = (\dot{G}^{f,\mathsf{gc}}, \dot{E}^{f,\mathsf{gc}}, D^O)$, removing eval^f queries asked by E^O. We have to make sure $\dot{\mathcal{E}}$ is 'almost as secure' as the original scheme $\mathcal{E}$. This step is detailed in Section 4.1.
- **Step 2:** Given $\mathcal{E} = (G^{f,\mathsf{gc}}, E^{f,\mathsf{gc}}, D^O)$, we compile $\mathcal{E}$ into a new scheme $\ddot{\mathcal{E}} = (\ddot{G}^{f,\mathsf{gc}}, \ddot{E}^{f,\mathsf{gc}}, \ddot{D}^{f,\mathsf{gc}})$, removing eval^f queries asked by D^O. Again, we have to make sure $\ddot{\mathcal{E}}$ is 'almost as secure' as the original one. This step is detailed in Section 4.2.

Once we accomplish both of the steps above, we will combine them into a single compiler that removes eval^f queries from $\mathcal{E}^O$, obtaining another PKE construction that is secure in the random oracle model (which we already know is impossible by the result of [23]).

Overview of Step 1. Let us start by looking at eval queries of the encryption algorithm $E^O(pk, b)$. Since the subroutine gc of oracle O is just a random mapping, for any eval query on inputs $(\widetilde{\mathsf{C}}, \widetilde{X})$, denoted $qu = ((\widetilde{\mathsf{C}}, \widetilde{X}) \xrightarrow[\mathsf{eval}]{} ?)$, made by E^O and whose answer is not trivially $\perp$, we must have either of the following cases. Either (a) $\widetilde{\mathsf{C}}$ was produced as a result of a gc query during the execution of E itself or (b) $\widetilde{\mathsf{C}}$ was produced during the execution of G which has led to the generation of the public key pk. If case (a) holds, then E does not need to make that particular eval query at all. If case (b) holds, then in order to allow $\dot{E}^{f,\mathsf{gc}}$ to simulate E^O without calling eval^f, the algorithm $\dot{E}^{f,\mathsf{gc}}$ will resort to some 'hint list' H attached to pk by $\dot{G}^{f,\mathsf{gc}}$. That is, a compiled public key $\dot{pk}$ produced by $\dot{G}^{f,\mathsf{gc}}$ will now contain the original pk as well as a hint list H. Below, we further explain how the hint H is formed.

How $\dot{G}^{f,\mathsf{gc}}$ forms the hint list H. A naive idea is to let H contain all the query/response pairs made by $G^{f,\mathsf{gc}}$ to generate pk. This method hurts security. A better idea is to provide in H answers to individual eval queries like $\mathsf{eval}(\widetilde{\mathsf{C}}, \widetilde{X})$ that are *likely* to be asked by $E^O(pk, b)$, and where $\widetilde{\mathsf{C}}$ was generated by $G^{f,\mathsf{gc}}$. That is, $\dot{G}^{f,\mathsf{gc}}$ would run $E^O(pk, b)$ many times and would let H contain all encountered eval queries as well as their answers. Note that $\dot{G}^{f,\mathsf{gc}}$ could simulate almost perfectly a random execution of $E^O(pk, b)$ without calling eval since $\dot{G}$ knows the randomness seeds of all the garbled circuits so far. However, this approach also fails! To see the difficulty, recall that a whole garbled input $\widetilde{X} =$

[5] More formally, because of the huge output space of gc, calling eval^f on a garbled circuit $\widetilde{\mathsf{C}}$ that is produced on the fly is bound to be responded with $\perp$ with overwhelming probability as $\widetilde{\mathsf{C}}$ will not be an encoding of any circuit.

$(\widetilde{x}_1, \ldots, \widetilde{x}_m)$ is made up of a sequence of garbled labels $\widetilde{x}_i$, one for each input wire. Now imagine that for a garbled circuit $\widetilde{\mathsf{C}}$ that was generated by $\dot{G}$, any new execution of $E^O(pk, b)$ calls the oracle eval on $\widetilde{\mathsf{C}}$ and on a new garbled input $\widetilde{X}$ that is formed by picking each of the garbled labels uniformly at random from the set of two lables for that corresponding input wire. If E behaves this way, then no matter how many (polynomial) times we sample from $E^O(pk, b)$, we cannot hope to predict the garbled-input part of the eval query of the next execution of $E^O(pk, b)$. We refer to this as the *garbled-input unperdictability* problem, which stems from the decomposability nature of our garbling oracle. This is what makes our results different from those of [2], which dealt with non-decomposable garbling, for which such a complication is absent.

In short, we could only hope to predict the garbled circuit part of an eval query of $E^O(pk, b)$, and not necessarily the garbled-input part. To fix this garbled-input unpredictability problem, $\dot{G}^{f,\mathsf{gc}}$ will do the following trick: while sampling many executions of $E^O(pk, b)$, if $\dot{G}^{f,\mathsf{gc}}$ comes across two different eval queries $\mathsf{eval}(\widetilde{\mathsf{C}}, \widetilde{X_1}), \mathsf{eval}(\widetilde{\mathsf{C}}, \widetilde{X_2})$ that are both answered with a value that is not $\perp$ (i.e., both are valid garbled circuits and inputs), then $\dot{G}^{f,\mathsf{gc}}$ releases the corresponding seed s and the *plain* circuit C of $\widetilde{\mathsf{C}}$. That is, if $\mathsf{gc}(s, \mathsf{C}) = (\widetilde{\mathsf{C}}, \cdots)$, then $\widetilde{G}^{f,\mathsf{gc}}$ puts the tuple $(s, \mathsf{C}, \widetilde{\mathsf{C}})$ into the hint list H. If, however, during these sampling, $\widetilde{\mathsf{C}}$ is evaluated upon *at most* one matching $\widetilde{X}$, then $\dot{G}^{f,\mathsf{gc}}$ simply provides the answer to the query $\mathsf{eval}(\widetilde{\mathsf{C}}, \widetilde{X})$ in H.

Looking ahead, the algorithm $\dot{E}^{f,\mathsf{gc}}((pk, H), b)$, when facing an eval query $qu = \mathsf{eval}(\widetilde{\mathsf{C}}, \widetilde{X})$, will check whether qu is already answered in H, or whether the corresponding seed s and plain-circuit C of $\widetilde{\mathsf{C}}$ could be retrieved from H. If so, $\dot{E}^{f,\mathsf{gc}}$ will reply to qu accordingly; otherwise, it will reply to qu with $\perp$.

Using the weakening subroutine rev to reduce the security of $\dot{\mathcal{E}}$ to $\mathcal{E}$. Note that $\dot{G}^{f,\mathsf{gc}}$ does not query any oracle subroutines beyond f and gc in order to form the hint list H attached to pk. This is because $\dot{G}^{f,\mathsf{gc}}$ has all the (otherwise-hidden) query-answer pairs used to produce pk, and thus for any encountered valid garbled circuit $\widetilde{\mathsf{C}}$ during those sampled executions of E, $\dot{G}^{f,\mathsf{gc}}$ already knows the corresponding seed s and plain circuit C. Now we are left to show that this additional information H attached to pk does not degrade the security of the compiled scheme significantly. To this end, we will use the new weakening oracle intended to capture the natural use of garbling: the security of a garbled circuit $\widetilde{\mathsf{C}}$ is guaranteed to hold so long as $\widetilde{\mathsf{C}}$ is evaluated only on one garbled input. Capturing this, our new oracle rev takes as input a garbled circuit $\widetilde{\mathsf{C}}$ and two garbled inputs $\widetilde{X_1}$ and $\widetilde{X_2}$, and if all of $\widetilde{\mathsf{C}}$, $\widetilde{X_1}$ and $\widetilde{X_2}$ are encoded using the same seed s, then rev simply outputs (s, C), where $\mathsf{gc}(s, \mathsf{C}) = \widetilde{\mathsf{C}}$. For security, we will show that any adversary against the semantic security of $\dot{\mathcal{E}}$ may be used in a black-box way, along with oracle access to $(f, \mathsf{gc}, \mathsf{eval}, \mathsf{rev})$, to mount an attack against the original scheme $\mathcal{E}$. This shows that the leakage caused by revealing H was also attainable in the original scheme (in which all parties including the attacker do have access to eval) if, in addition, access to

the oracle rev — which reflects the intuitive way in which garbled circuits are supposed to be used — was also granted to the adversary. In our security proof we will crucially make use of the rev subroutine in order to construct tuples $(s, \mathsf{C}, \widetilde{\mathsf{C}})$ to store in the simulated hint list whenever $\widetilde{\mathsf{C}}$ should be evaluated on two different inputs. Tuples of the form $(\widetilde{\mathsf{C}}, \widetilde{X}, y)$ can in turn be simulated using oracle access to eval.

Overview of Step 2. The main idea is similar to Step (1): $\ddot{E}^{f,\mathsf{gc}}(pk, b)$ would first run $E^{f,\mathsf{gc}}(pk, b)$ to get the ciphertext c and then appropriately attach a hint H to c. The idea is that H should allow the eval-free algorithm $\ddot{D}^{f,\mathsf{gc}}((sk, \mathsf{H}), c)$ to simulate $D^{f,\mathsf{gc},\mathsf{eval}}(sk, c)$ well enough. Again, since we cannot simply copy the entire private view of $\ddot{E}^{f,\mathsf{gc}}(pk, b)$ into H (as that cannot be simulated by the security reduction, and therefore would hurt security) we should instead ensure that w.h.p. all eval queries during the execution of $D^{O}(sk, c)$, whose garbled circuits were generated by $\ddot{E}^{f,\mathsf{gc}}(pk, b)$, can be answered using H. Let us call these eval queries $\ddot{E}$-*tied* queries. Unfortunately, when implementing this idea, we run into the following problem: $\ddot{E}^{f,\mathsf{gc}}(pk, b)$ cannot simply run $D^{O}(sk, c)$ to get a sense of eval queries because sk is private; this was absent in Step (1).

In order to resolve this new challenge, the algorithm $\ddot{E}^{f,\mathsf{gc}}(pk, b)$ needs to do some more offline work in order to get an idea of $\ddot{E}$-tied eval queries that come up during $D^{O}(sk, c)$. The main idea is that although the true secret key sk is unknown to $\ddot{E}^{f,\mathsf{gc}}(pk, b)$, in the eyes of $\ddot{E}^{f,\mathsf{gc}}(pk, b)$, the value of sk is equally likely to be any sk' that agrees with the entire view of $\ddot{E}^{f,\mathsf{gc}}(pk, b)$. Put differently, the probability that an $\ddot{E}$-tied garbled circuit comes up during $D^{O}(sk, c)$ is close to the probability that it comes up during the execution of $D^{O'}(sk', c)$, where O' is an offline oracle that agrees with all the private information of $\ddot{E}$, and also relative to which (pk, sk') is valid public-key/secret-key. As a result, such a fake sk' that is consistent with the view of $\ddot{E}^{f,\mathsf{gc}}(pk, b)$ will be used to learn the answers of the evaluation queries asked by $D^{O'}(sk', c)$[6].

Putting things together. Taken together, Steps (1) and (2) in conjunction with the result of Imagliazzo and Rudich [23] imply the following.

Lemma 2 (Informal). *The (claimed) semantic security of any candidate PKE construction $\mathcal{E}^{f,\mathsf{gc},\mathsf{eval}}$ can be broken by a poly-query adversary $\mathcal{A}^{f,\mathsf{gc},\mathsf{eval},\mathsf{rev}}$.*

Moreover, we can show that the oracle rev does not break the one-wayness or the garbling-security aspects of $(f, \mathsf{gc}, \mathsf{eval})$.

Lemma 3 (Informal). *The function f is one way against all poly-query adversaries with oracle access $(f, \mathsf{gc}, \mathsf{eval}, \mathsf{rev})$. Moreover, there exists a garbling scheme $L^{f,\mathsf{gc},\mathsf{eval}}$ for garbling circuits with f gates that remains secure against all poly-query adversaries $\mathcal{B}^{f,\mathsf{gc},\mathsf{eval},\mathsf{rev}}$.*

Now Lemmas 2 and 3 imply our main theorem, Theorem 1.

[6] The process of discovering such an sk' is what makes $\dot{E}$ an inefficient algorithm.

3 Preliminaries

We use κ for the security parameter. By PPT we mean a probabilistic polynomial time algorithm. By an *oracle* PPT/algorithm we mean a PPT that may make oracle calls. For any oracle algorithm A that has access to some oracle O, we denote a query qu asked by A to a subroutine T of O as $(qu \underset{T}{\longrightarrow} ?)$. If the returned answer is β, then we denote the resulting query-answer pair as $(qu \underset{T}{\longrightarrow} \beta)$. For a set S of query/answer pairs, we will use intuitive notation such as $(* \underset{T}{\longrightarrow} \beta) \in \mathsf{S}$ to mean that there exists a query qu such that $(qu \underset{T}{\longrightarrow} \beta) \in \mathsf{S}$. We use $||$ to concatenate strings and we use "," for attaching strings in a way they could be retrieved. Namely, one can uniquely identify x and y from (x, y). For example $(00||11) = (0011)$, but $(0, 011) \neq (001, 1)$. For any given string x, we denote x_i to be the i'th string of x. For (family of) random variables $\{X_\kappa, Y_\kappa\}_\kappa$, by $X \overset{\text{c}}{\approx} Y$ we denote that they are computationally indistinguishable; namely, for any $\text{poly}(\kappa)$-time adversary A there is a negligible function $\text{negl}(\kappa)$ such that $|\Pr[A(X_\kappa) = 1] - \Pr[A(Y_\kappa) = 1]| \leq \text{negl}(\kappa)$. When writing the probabilities, by putting an algorithm A in the subscript of the probability (e.g., $\Pr_A[\cdot]$) we emphasize that the probability is over A's randomness. For any given probability distribution $\mathbf{D}$, we denote $x \leftarrow \mathbf{D}$ as sampling from this distribution and obtaining a sample x from the support of $\mathbf{D}$. We may also use $x \in \mathbf{D}$ to mean that x is in the support of $\mathbf{D}$. For any two random variables X, Y, we denote $\Delta(X, Y)$ to be the statistical distance between the two random variables. Throughout the paper, whenever we write $f_1(\kappa) \leq f_2(\kappa)$ we mean that this inequality holds asymptotically; i.e., there exists κ_0 such that for all $\kappa \geq \kappa_0$, $f_1(\kappa) \leq f_2(\kappa)$.

3.1 Some Useful Lemmas

The following lemma shows that hitting the image of a sparse injective random function without having called the function on the corresponding preimage happens with negligible probability.

Lemma 4 (Hitting the image of random injective function). *Let A be an arbitrary polynomial-query algorithm with access to an oracle $O : \{0,1\}^\kappa \to \{0,1\}^{2\kappa}$ chosen uniformly at random from the set of all injective functions from $\{0,1\}^\kappa$ to $\{0,1\}^{2\kappa}$. We have*

$$\Pr[y \leftarrow A^O(1^\kappa) \mid \text{ for some } x\colon y = O(x) \wedge (* \underset{O}{\longrightarrow} y) \notin \mathsf{Q_A}] \leq 2^{-\kappa/2},$$

where the probability is taken over the random choice of O as well as $\mathcal{A}$'s random coins, and where $\mathsf{Q_A}$ is the set of all A's query-answer pairs.

We will also use the following standard information theoretic lemma frequently in the paper.

Lemma 5. *Let $X_1, \ldots, X_{t+1}$ be independent, Bernoulli random variables, where $Pr[X_i = 1] = p$, for all $i \leq t + 1$. Then*

$$\Pr[X_1 = 0 \wedge \cdots \wedge X_t = 0 \wedge X_{t+1} = 1] \leq \frac{1}{t}.$$

3.2 Standard Primitives

The definition of a single-bit public key encryption scheme (G, E, D) with $(\frac{1}{2} + \delta)$-correctness is standard. For $\gamma = \gamma(\kappa)$ we say that an adversary $\mathcal{A}$ γ-breaks (G, E, D) if the advantage of the adversary in the standard semantic-security game is at least γ. See the full version for formal definitions.

Oracle aided circuits. A binary-output oracle-aided circuit C is a circuit with Boolean gates as well as oracle gates, and where the output of the circuit is a single bit. The input size, $\mathsf{inpsize}(\mathsf{C})$, is the number of input wires. The circuit size, denoted $|\mathsf{C}|$, denotes the number of gates and input wires of the circuit. For a fixed function f we write C^f to denote the circuit C when the underlying oracle is fixed to f.

Definition 6 (Garbling schemes for oracle-aided circuits). *Fix a function f. A circuit garbling scheme for oracle-aided circuits relative to f (or with f gates) is a triple of algorithms $(\mathsf{Garb}, \mathsf{Eval}, \mathsf{Sim})$ defined as follows:*

- $\mathsf{Garb}\,(1^\kappa, \mathsf{C})$*: takes as input a security parameter κ, an oracle-aided circuit C and outputs a garbled circuit $\widetilde{\mathsf{C}}$ with a set of labels $\{\mathsf{label}_{i,b}\}_{i\in[m],b\in\{0,1\}}$, where $m = \mathsf{inpsize}(\mathsf{C})$.*
- $\mathsf{Eval}^f\left(\widetilde{\mathsf{C}}, \{\mathsf{label}_{i,b_i}\}_{i\in[m]}\right)$*: takes as input a garbled circuit $\widetilde{\mathsf{C}}$ and a sequence of garbled input labels $\{\mathsf{label}_{i,b_i}\}_{i\in[m]}$ and outputs $y \in \{0,1\}^* \cup \{\perp\}$.*

  We define the following notions.

- **Correctness.** *For any oracle-aided circuit C and input $x \in \{0,1\}^m$, where $m = \mathsf{inpsize}(\mathsf{C})$:*

$$\Pr\left[\mathsf{C}^f(x) = \mathsf{Eval}^f\left(\widetilde{\mathsf{C}}, \{\mathsf{label}_{i,x_i}\}_{i\in[m]}\right)\right] = 1$$

  where the probability is taken over $\mathsf{Garb}\,(1^\kappa, \mathsf{C}) \mapsto (\widetilde{\mathsf{C}}, \{\mathsf{label}_{i,b}\}_{i\in[m],b\in\{0,1\}})$.
- **Security.** *For any polynomial $m = m(\kappa)$, any poly-size oracle circuit C with input size m, and any input $x \in \{0,1\}^m$:*

$$\left(\widetilde{\mathsf{C}}, \{\mathsf{label}_{i,x_i}\}_{i\in[m]}\right) \overset{\mathrm{c}}{\approx} \mathsf{Sim}\left(1^{|\mathsf{C}|}, m, \mathsf{C}^f(x)\right)$$

  where $(\widetilde{\mathsf{C}}, \{\mathsf{label}_{i,b}\}_{i\in[m],b\in\{0,1\}}) \leftarrow \mathsf{Garb}\,(1^\kappa, \mathsf{C})$.

3.3 Black-box Constructions

Now, we recall the standard notion of black-box constructions [23,32,4]. We do so in the context of building PKE from one-way functions and garbling.

Definition 7 (Black-box constructions of PKE from GC-OWF). *A fully black-box construction of a PKE scheme from a one-way function and a garbling scheme for circuits with one-way function gates (shortly, from GC-OWF) consists of a triple of PPT oracle algorithms (G, E, D) and a PPT oracle security-reduction $S = (S_1, S_2)$ such that for any function f and any correct garbling scheme $L = (\mathsf{Garb}, \mathsf{Eval}, \mathsf{Sim})$ relative to f, both the following hold:*

- ***Correctness:*** $\mathcal{E}^{f,L} = (G^{f,L}, E^{f,L}, D^{f,L})$ *is a* $(1 - \frac{1}{2^\kappa})$*-correct PKE scheme. (See the remark after this definition.)*
- ***Security:*** *For any adversary any* A *that breaks the semantic security of the PKE scheme* $\mathcal{E}^{f,L}$*, either*
  - $S_1^{f,L,A}$ *breaks the one-wayness of* f*; or*
  - $S_2^{f,L,A}$ *breaks the security of the scheme* $L = (\mathsf{Garb}, \mathsf{Eval}, \mathsf{Sim})$ *relative to* f*. That is, for some oracle-aided circuit* C *and input* x*,* $S_2^{f,L,A}$ *can distinguish between the tuple* $\left(\widetilde{\mathsf{C}}, \{\mathsf{label}_{i,x_i}\}_{i\in[m]}\right)$ *and* $\mathsf{Sim}\left(1^{|\mathsf{C}|}, 1^{|x|}, \mathsf{C}^f(x)\right)$*, where* $m = \mathsf{inpsize}(\mathsf{C})$ *and* $(\widetilde{\mathsf{C}}, \{\mathsf{label}_{i,b}\}_{i\in[m],b\in\{0,1\}}) \leftarrow \mathsf{Garb}\left(1^\kappa, \mathsf{C}\right)$*.*

Remark about the correctness condition in Definition 7. In Definition 7, for correctness we require that the constructed PKE be $(1 - \frac{1}{2^\kappa})$ correct. This is without loss of generality since one may easily boost correctness using standard techniques; i.e., let the new public key be a tuple of public keys under the original scheme. Encrypt a given plaintext bit under each individual public key. For decryption, we decrypt all the ciphertexts and go with the majority bit. The semantic security of this expanded scheme reduces to that of the base scheme using a hybrid argument, which is a fully-black-box reduction.

Calling the base primitives on the same security parameter. For simplicity of exposition, for any given black-box construction $\mathcal{E}^{f,L}$ we assume that $\mathcal{E}^{f,L}$ on the security parameter 1^κ always calls f and L on the same security parameter 1^κ. There are standard techniques for doing away with this restriction, but those extensions will only complicate the proofs further. Looking ahead, when we define our oracles $(O', \mathsf{rev})_{\kappa,n}$ in Definition 10, which are parameterized over a security parameter κ and a circuit size $n = n(\kappa)$, the above restriction means that $\mathcal{E}^{O'}$ on the security parameter 1^κ always calls O' on parameters such as (κ, n_1), (κ, n_2), etc. That is, the value of κ will be the same across all queries, but each query may use a different value for n.

4 Separating Public-Key Encryption from OWF-based Garbling

In this section, we state our main impossibility result and describe at a high-level the steps that we will take in order to prove our main theorem.

Theorem 8 (Main theorem). *There exists no fully black-box construction of a public-key encryption scheme from GC-OWFs; namely garbling schemes that garble circuits with one-way function gates in them (see Definition 7).*

Our theorem above follows from the following lemma.

Lemma 9. *There exists an oracle* $O = (f, \mathsf{gc}, \mathsf{gi}, \mathsf{eval}, \mathsf{rev})$ *for which the following holds (in what follows, let* $O' = (f, \mathsf{gc}, \mathsf{gi}, \mathsf{eval})$*):*

1. *f is one-way relative to (O', rev). That is, f is one-way against all polynomial query (and even sub-exponential query) adversaries $\mathcal{A}^{O',\mathsf{rev}}$.*
2. *There exists a PPT GC-OWF construction $(\mathsf{Garb}^{O'}, \mathsf{Eval}^{O'}, \mathsf{Sim}^{O'})$ for f-aided circuits that is secure against any poly-query adversary $\mathcal{A}^{O',\mathsf{rev}}$.*
3. *For any PKE construction $\mathcal{E}^{O'}$ with access to the oracle O', there exists an attacker $\mathcal{A}^{O',\mathsf{rev}}$ that breaks the semantic security of $\mathcal{E}^{O'}$ using a polynomial number of queries.*

Note that Lemma 9 immediately implies Theorem 8.

Roadmap: Proof of Lemma 9. As common in black-box impossibility results, we will show the existence of the oracles required by Lemma 9 by proving results with respect to oracles chosen randomly according to a distribution. We will describe our oracle distribution below and will then outline the main steps we will take in order to prove Lemma 9.

Definition 10 (The ideal model/oracle). *Let $O = (f, \mathsf{gc}, \mathsf{gi}, \mathsf{eval}, \mathsf{rev})_{\kappa,n}$ be an ensemble of oracles parameterized by (κ, n), where κ denotes the security parameter and n denotes the size of a circuit which we want to garble. We describe the distribution $\mathbf{O}$ from which these oracles are sampled for fixed (κ, n).*

- *$f : \{0,1\}^\kappa \to \{0,1\}^\kappa$: a uniformly chosen random function.*
- *$\mathsf{gc}(s, F) : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^{2(\kappa+n)}$: an injective random function that, given a key $s \in \{0,1\}^\kappa$ and a single-bit-output oracle-aided circuit F, outputs an encoding $\widetilde{F}$.*
- *$\mathsf{gi}(s, i, x_i) : \{0,1\}^\kappa \times \{0,1\}^{\log n} \times \{0,1\} \to \{0,1\}^{2(\kappa+\log n)}$: an injective random function that, given a key $s \in \{0,1\}^\kappa$, an index $i \in \{0,1\}^{\log n}$, an input-wire bit value $x_i \in \{0,1\}$, outputs an encoding $\widetilde{x}_i$. As notation, for any $X = (x_1, ..., x_n)$, we denote $\mathsf{gi}(s, X) := (\mathsf{gi}(s, i, x_i))_{i\in[n]} = \widetilde{X}$.*
- *$\mathsf{eval}(\widetilde{F}, \widetilde{X})$: given as input $\widetilde{F}$ and $\widetilde{X} = (\widetilde{x}_1, ..., \widetilde{x}_m)$, if there is a string $s \in \{0,1\}^\kappa$ and circuit F such that $\mathsf{gc}(s, F) = \widetilde{F}$, that $m = \mathsf{inpsize}(F)$ and that for every $i \in [m]$ there exists $x_i \in \{0,1\}$ such that $\mathsf{gi}(s, i, x_i) = \widetilde{x}_i$, then it outputs $F^f(x_1 || \cdots || x_m)$. Otherwise, it outputs $\perp$.*
- *$\mathsf{rev}(\widetilde{F}, \widetilde{X}, \widetilde{X}')$: if there exists $s \in \{0,1\}^\kappa$ and circuit F such that $\mathsf{gc}(s, F) = \widetilde{F}$ and that there exists $X, X' \in \{0,1\}^{\mathsf{inpsize}(F)}$ such that $X \neq X'$, $\mathsf{gi}(s, X) = \widetilde{X}$ and $\mathsf{gi}(s, X') = \widetilde{X}'$, then it outputs (s, F). Otherwise, it outputs $\perp$.*

Remark 11. The size of a garbled circuit outputted by the gc oracle is roughly twice the size of the corresponding input circuit. Current garbled circuits constructions are not capable of achieving such a short expansion factor. We are able to do this as we model the garbling mechansim as a totally random function. Nonetheless, working with such a short size expansion is without loss of generality, because a general black-box PKE construction out of GC-OWF should work with respect to any oracle that implements the GC-OWF securely. We should also mention that all our results hold (without having to make any changes) if the output of gc is bigger than the one specified in Definition 10.

First, we show that a random oracle $O = (f, \mathsf{gc}, \mathsf{gi}, \mathsf{eval}, \mathsf{rev})$ chosen according to the distribution $\mathbf{O}$ allows us to implement an ideal version of garbling for circuits with f gates. This is not surprising as O is indeed an idealized form of implementing this primitive.

Lemma 12 (Secure OWF and garbling exists relative to O). *Let $O = (f, \mathsf{gc}, \mathsf{gi}, \mathsf{eval}, \mathsf{rev})$ be as in Definition 10 and let $O' = (f, \mathsf{gc}, \mathsf{gi}, \mathsf{eval})$. Then, with probability (measure) one over the choice of O, the function f is one-way relative to O — i.e., f is one-way against any PPT oracle adversary with access to the oracle O. Moreover, there exists a PPT GC-OWF construction $(\mathsf{Garb}^{O'}, \mathsf{Eval}^{O'})$ for f-aided circuits which is secure relative to O with probability one over the choice of $O \leftarrow \mathbf{O}$.*

Proof. The fact that f is one-way relative to O with probability one over the choice of O is now standard (see [23]). Given any oracle $O = (O', \mathsf{eval})$, we now show how to construct a PPT garbling scheme $L^{O'} = (\mathsf{Garb}^{O'}, \mathsf{Eval}^{O'}, \mathsf{Sim}^{O'})$ for f-aided circuits. The algorithm $\mathsf{Garb}^{O'}$ on input $(1^\kappa, C)$ samples $s \leftarrow \{0,1\}^\kappa$, sets $m = \mathsf{inpsize}(C)$ and outputs the garbled circuit $\widetilde{C} = \mathsf{gc}(s, C)$ as well as a sequence of garbled inputs $(\widetilde{x}_{1,0}, \widetilde{x}_{1,1}, \ldots, \widetilde{x}_{m,0}, \widetilde{x}_{m,1})$, where for $i \in m$ and $b \in \{0,1\}$ we have $\widetilde{x}_{i,b} = \mathsf{gi}(s, i, b)$.

The algorithm $\mathsf{Eval}^{O'}(\widetilde{C}, \widetilde{x}_1 || \cdots || \widetilde{x}_m)$ simply outputs $\mathsf{eval}(\widetilde{C}, \widetilde{x}_1 || \cdots || \widetilde{x}_m)$. Correctness holds by definition of the oracle.

For security, we will define $\mathsf{Sim}^{O'}$ as follows: on input $(1^\kappa, n, m, y \in \{0,1\})$, where n denotes the size of the circuit, m denotes the number of input wires and y denotes the output value, we set C_0 to be a canonical circuit of size n and with m input wires that always outputs y. Sample $s \leftarrow \{0,1\}^\kappa$ and let $\widetilde{C} = \mathsf{gc}(s, C)$ and $\widetilde{X} = \mathsf{gi}(s, 0^m)$. Output $(\widetilde{C}, \widetilde{X})$. Simulation security follows from the random nature of the oracles. That is, for any polynomial-query distinguisher $\mathcal{A}^{O', \mathsf{rev}}$, for any n, m and any circuit C of size n and of input size m and any input $X \in \{0,1\}^m$, we have

$$\left| \Pr[\mathcal{A}^{O', \mathsf{rev}}(\widetilde{C}, \widetilde{X}) = 1] - \Pr[\mathcal{A}^{O', \mathsf{rev}}(\widetilde{C'}, \widetilde{X'}) = 1] \right| = \mathrm{negl}(\kappa), \tag{1}$$

where $s \leftarrow \{0,1\}^\kappa$, $\widetilde{C} = \mathsf{gc}(s, C)$, $\widetilde{X} = \mathsf{gi}(s, X)$, $(\widetilde{C'}, \widetilde{X'}) \leftarrow \mathsf{Sim}^{O'}(1^\kappa, n, m, C^f(X))$. We omit the details of the proof of Equation 1 as it can be obtained through a simple information theoretic argument.

We are left with proving Part 3 of Lemma 9. Proving this part is the main technical contribution of our paper, and is done via an oracle *reducibility* technique. In order to state this reducibility statement formally, we first need to define the notions of correctness and attack advantage in the *ideal model*.

Definition 13 (Correctness in the ideal model). *For a polynomial $p = p(\kappa)$ we say that a single-bit PKE scheme $\mathcal{E}^O = (G^O, E^O, D^O)$ is $\frac{1}{2} + \frac{1}{p}$ correct in the ideal model if for both $b \in \{0,1\}$:*

$$\Pr[D^O(sk, c) = b] \geq \frac{1}{2} + \frac{1}{p}, \tag{2}$$

where the probability is over $O \leftarrow \mathbf{O}$, $(pk, sk) \leftarrow G^O(1^\kappa)$, $c \leftarrow E^O(pk, b)$.

Definition 14 (Ideal model attack advantage). *We say that an adversary* $\mathcal{A}$ *breaks the semantic security of a single-bit PKE* (G^O, E^O, D^O) *in the ideal model with probability* γ *(or with advantage* γ*) if* $\Pr[A(pk, c) = b] \geq \gamma$*, where the probability is taken over* $O \leftarrow \mathbf{O}$, $(pk, sk) \leftarrow G^O(1^\kappa)$, $b \leftarrow \{0, 1\}$, $c \leftarrow E^O(pk, b)$ *and over* $\mathcal{A}$*'s random coins.*

We are now ready to describe our oracle reducibility lemma.

Lemma 15 (Reducibility to the random oracle model). *Let* $\mathcal{E}$ *be a given PKE construction possibly making use of all the oracles* $O' = (f, \mathsf{gc}, \mathsf{gi}, \mathsf{eval})$*. There exists a compilation procedure and a polynomial-query security-reduction* Red *such that the compilation transforms* $\mathcal{E}^{O'}$ *into a new polynomial-query PKE construction* $\ddot{\mathcal{E}}^{f,\mathsf{gc},\mathsf{gi}}$*, where* $\ddot{\mathcal{E}}$ *makes no* eval *queries and for which both the following hold:*

- ***Correctness****: If* $\mathcal{E}^{O'}$ *is* $(1 - \frac{1}{2^\kappa})$ *correct in the ideal model, the compiled scheme* $\ddot{\mathcal{E}}^{f,\mathsf{gc},\mathsf{gi}}$ *has at least* $(1 - \frac{1}{\kappa^7})$ *correctness in the ideal model.*
- ***Security reduction****. For any constant* c *the following holds: if there exists an adversary* $\mathcal{A}$ *that breaks the semantic security of* $\ddot{\mathcal{E}}^{f,\mathsf{gc},\mathsf{gi}}$ *in the ideal model with probability* η*, the algorithm* $\mathrm{Red}^{O',\mathsf{rev},\mathcal{A}}$ *breaks the semantic security of* $\mathcal{E}^{O'}$ *in the ideal model with probability* $\eta - \frac{1}{\kappa^c}$*.*

Let us first show how to use lemmas 12 and 15 to establish Lemma 9.

Completing proof of Lemma 9 and Theorem 8. Let $\mathcal{E} = (G, E, D)$ be a candidate PKE construction. We will show that with probability one over the choice of $(O', \mathsf{rev}) \leftarrow \mathbf{O}$, the PKE construction $\mathcal{E}^{O'}$ can be broken by a polynomial number of queries to (O', rev). Let us first show how to use this claim to complete the proof of Theorem 8, and we will then prove this claim. By Lemma 12, we know that with probability one over the choice of O we have (a) f is one-way relative to (O', rev) and (b) $(\mathsf{Garb}^{O'}, \mathsf{Eval}^{O'}, \mathsf{Sim}^{O'})$ is a secure GC-OWF construction for f-aided circuits against all polynomial-query adversaries with access to the oracles (O', rev). Thus, the foregoing claim coupled with Lemma 12 implies Lemma 9. In what follows we prove the foregoing claim.

By Definition 7 we know that $\mathcal{E}^{O'}$ has $(1-\frac{1}{2^\kappa})$-correctness in the ideal model. Thus, by Lemma 15 there exists a compiled scheme $\ddot{\mathcal{E}}^{f,\mathsf{gc},\mathsf{gi}}$ that has at least $(1 - \frac{1}{\kappa^7})$-correctness in the ideal model. Note that the oracles f, gc and gi are nothing but three independent random oracles. By the results of [23,7] there exists a polynomial query adversary $\mathcal{A}^{f,\mathsf{gc},\mathsf{gi}}$ which breaks the semantic security of $\ddot{\mathcal{E}}^{f,\mathsf{gc},\mathsf{gi}}$ in the ideal model with probability $(1-\frac{1}{\kappa^6})$.[7] (See Definition 14 for the notion of "break in the ideal model.") Invoking Lemma 15 again and choosing

[7] The results of [23,7] show how to break the semantic security of any key exchange (and hence PKE) construction in the random oracle model with a probability that is at most $\frac{1}{\kappa^{c'}}$ less than the correctness probability, for any arbitrary constant $c' > 0$.

the constant c appropriately, we will obtain a polynomial query adversary $\mathcal{B}^{O',\mathsf{rev}}$ which breaks the semantic security of $\mathcal{E}^{O'}$ in the ideal model with probability $(1-\frac{1}{\kappa^5})$. That is,

$$\Pr_{O=(O',\mathsf{rev}),pk,b,c}\left[\mathcal{B}^{O',\mathsf{rev}}(pk,c)=b\right]\geq 1-\frac{1}{\kappa^5}, \tag{3}$$

where $(pk,sk)\leftarrow G^{O'}(1^\kappa)$, $b\leftarrow\{0,1\}$ and $c\leftarrow E^{O'}(pk,b)$.

Using a simple averaging argument we have

$$\Pr_{O=(O',\mathsf{rev})}\left[\Pr_{pk,b,c}\left[\mathcal{B}^{O',\mathsf{rev}}(pk,c)=b\right]\geq 1-\frac{1}{\kappa^3}\right]\geq 1-\frac{1}{\kappa^2}. \tag{4}$$

Equation 4 implies that for at most $\frac{1}{\kappa^2}$ fraction of the oracles $O=(O',\mathsf{rev})$, the adversary $\mathcal{B}^{O',\mathsf{rev}}(pk,c)$, on security parameter κ, recovers b with probability less than $1-\frac{1}{\kappa^3}$. Since $\sum_{i=1}^{\infty}\frac{1}{i^2}$ converges, by the Borel-Cantelli Lemma we have that for a measure-one fraction of oracles $O=(O',\mathsf{rev})\leftarrow\mathbf{O}$, the adversary $\mathcal{B}^{O',\mathsf{rev}}$ breaks the semantic security of $\mathcal{E}^{O'}$. The proof is now complete. □

Roadmap for the proof of Lemma 15. Finally, all that remains is proving Lemma 15 which shows that we can compile out eval queries from any PKE scheme without significantly hurting correctness or security. In the remainder of this paper, we show that such a compilation procedure exists. We obtain the compiled eval-free scheme $(\ddot{G}^{f,\mathsf{gc},\mathsf{gi}},\ddot{E}^{f,\mathsf{gc},\mathsf{gi}},\ddot{D}^{f,\mathsf{gc},\mathsf{gi}})$ in two steps. First, in Section 4.1, we show how to compile out eval queries from $E^{O'}$ only. In particular, we will prove the following lemma.

Lemma 16 (Compiling out eval from E). *Let δ be an arbitrary polynomial and parse $O=(O',\mathsf{rev})$. There exists a compilation procedure that achieves the following for any constant c. Given any $(\frac{1}{2}+\delta)$-ideally-correct PKE scheme $\mathcal{E}=(G^{O'},E^{O'},D^{O'})$, the compiled PKE scheme $\dot{\mathcal{E}}=(\dot{G}^{f,\mathsf{gc},\mathsf{gi}},\dot{E}^{f,\mathsf{gc},\mathsf{gi}},D^{O'})$ is $(\frac{1}{2}+\delta-\frac{1}{\kappa^c})$-ideally-correct. Moreover, there exists a polynomial-query algorithm $\mathtt{SecRed}$ that satisfies the following: for any adversary $\mathcal{A}$ that breaks the semantic security of $\dot{\mathcal{E}}$ in the ideal model with advantage η, the adversary $\mathtt{SecRed}^{\mathcal{A},O}$ breaks the semantic security of $\mathcal{E}$ in the ideal model with advantage at least $\eta-\frac{1}{\kappa^c}$.*

Then, in Section 4.2 we show how to compile out eval from $D^{O'}$, assuming neither of the algorithms G and E call eval. That is, we prove the following.

Lemma 17 (Compiling out eval from D). *Let δ be an arbitrary polynomial. There exists a compilation procedure that achieves the following for any constant c. Given any $(\frac{1}{2}+\delta)$-ideally-correct PKE scheme $\mathcal{E}=(G^{f,\mathsf{gc},\mathsf{gi}},E^{f,\mathsf{gc},\mathsf{gi}},D^{f,\mathsf{gc},\mathsf{gi},\mathsf{eval}})$, the compiled PKE scheme $\ddot{\mathcal{E}}=(\ddot{G}^{f,\mathsf{gc},\mathsf{gi}},\ddot{E}^{f,\mathsf{gc},\mathsf{gi}},\ddot{D}^{f,\mathsf{gc},\mathsf{gi}})$ is $(\frac{1}{2}+\delta-\frac{1}{\kappa^c})$-ideally-correct. Moreover, there exists a polynomial-query algorithm $\mathtt{SecRed}$ that satisfies the following: for any adversary $\mathcal{A}$ that breaks the semantic security of $\ddot{\mathcal{E}}$ in the ideal model with advantage η, the adversary $\mathtt{SecRed}^{\mathcal{A},O}$ breaks the semantic security of $\mathcal{E}$ in the ideal model with advantage at least $\eta-\frac{1}{\kappa^c}$.*

Th proof of Lemma 15 immediately follows from Lemmas 16 and 17.

4.1 Removing Garbling Evaluation Queries from Encryption

In this section, we will prove Lemma 16. Namely, we will show how to compile the PKE scheme $\mathcal{E} = (G^{f,\mathsf{gc},\mathsf{gi},\mathsf{eval}}, E^{f,\mathsf{gc},\mathsf{gi},\mathsf{eval}}, D^{f,\mathsf{gc},\mathsf{gi},\mathsf{eval}})$ into a new PKE scheme $\dot{\mathcal{E}} = (\dot{G}^{f,\mathsf{gc},\mathsf{gi}}, \dot{E}^{f,\mathsf{gc},\mathsf{gi}}, D^{f,\mathsf{gc},\mathsf{gi},\mathsf{eval}})$ with correctness and security comparable to the original scheme $\mathcal{E}$, but where $\dot{E}$ will not ask any eval queries. First, we may assume without loss of generality that G does not make queries to eval — it can predict the answer itself. Thus, we will focus on removing eval queries from E assuming that G does not make any eval queries.

Before describing the compilation process, we need to give some definitions.

Definition 18 (Valid outputs). *For any oracle $O = (f, \mathsf{gc}, \mathsf{gi}, \mathsf{eval}, \mathsf{rev})$, we say that $\widetilde{F}$ is a valid garbled circuit with respect to O if there exists (s, F) such that $\mathsf{gc}(s, F) = \widetilde{F}$. Similarly, we say that $\widetilde{X}$ is a valid garbled input with respect to O if there exists (s, X) such that $\mathsf{gi}(s, X) = \widetilde{X}$.*

We also define the notion of normal form with respect to oracle-aided algorithms. At a high-level, a normal form algorithm avoids asking any redundant queries if it already knows the answer to such queries.

Definition 19 (Normal form). *Let A be an oracle algorithm that accepts as input a query-answer set Q_S and let Q_A be the query-answer pairs that A has asked so far. We say that A is in* normal-form *if it satisfies these conditions:*

1. *A never asks duplicate queries.*
2. *Before it asks an $((\widetilde{F}, \widetilde{X}) \xrightarrow[\mathsf{eval}]{} ?)$ query qu, A first checks if there exists a query-answer pair $((s, F) \xrightarrow[\mathsf{gc}]{} \widetilde{F})$ in $\mathsf{Q}_\mathsf{A} \cup \mathsf{Q}_\mathsf{S}$. If that is the case then it would not issue qu to the oracle but would instead run $F^f(X)$ on its own where X can be obtained bit-by-bit by searching $\mathsf{gi}(s, i, x_i)$ for every index position $i \in n$ and every bit $x_i \in \{0, 1\}$.*

Recall that our goal is to remove eval queries from E to obtain an eval-free algorithm $\dot{E}$. To make this transformation possible, the new algorithm $\dot{E}$ needs some help from its associated key generation algorithm $\dot{G}$ so as to make up for its lack of access to eval. This help is sent to $\dot{E}$ as part of a hint list H, attached to the public key, by the key generation algorithm $\dot{G}$. The following definition describes how $\dot{G}$ forms the hint list H based on its inside information Aux and based on information Q that G has collected about random executions of E.

Definition 20 (Building helper tuples). *We define a function* `ConstHelp` *that takes as input a query-answer set Q along with some query-answer set Aux and outputs a set H as follows:*

- *If there exists $((\widetilde{F}, \widetilde{X}) \xrightarrow[\mathsf{eval}]{} y \neq \bot) \in \mathsf{Q}$ such that for no $\widetilde{X}' \neq \widetilde{X}$ do we have $((\widetilde{F}, \widetilde{X}') \xrightarrow[\mathsf{eval}]{} y' \neq \bot) \in \mathsf{Q}$, then add $((\widetilde{F}, \widetilde{X}) \xrightarrow[\mathsf{eval}]{} y)$ to H.*

– *If for two distinct* $\widetilde{X}_1$ *and* $\widetilde{X}_2$ *we have* $((\widetilde{F},\widetilde{X}_1) \xrightarrow[\mathsf{eval}]{} y_1 \neq \perp) \in \mathsf{Q}$ *and* $((\widetilde{F},\widetilde{X}_2) \xrightarrow[\mathsf{eval}]{} y_2 \neq \perp) \in \mathsf{Q}$*, then if for some* (s,F) *we have* $((s,F) \xrightarrow[\mathsf{gc}]{} \widetilde{F}) \in \mathsf{Aux}$*, add* $((s,F) \xrightarrow[\mathsf{gc}]{} \widetilde{F})$ *to* H.

Having a hint list H, we give the following definition that describes the idea of how the receiving algorithm $\dot{E}$ may use it to avoid making eval queries. In the following definition one may think of Q as a hint list.

Definition 21 (Emulating eval queries). *For* $O = (f, \mathsf{gc}, \mathsf{gi}, \mathsf{eval}, \mathsf{rev})$*, we define the function* $\texttt{HandleEval}^{f,\mathsf{gi}}$ *to be a subroutine that takes as input a set* Q *of query-answer pairs to* O *and a query* qu *of the form* $((\widetilde{F},\widetilde{X}) \xrightarrow[\mathsf{eval}]{} ?)$ *then performs the following steps to answer* qu*:*

– *If there exists a tuple* $((\widetilde{F},\widetilde{X}) \xrightarrow[\mathsf{eval}]{} y)$ *in* Q*, then output* y.
– *If there exists* $((s,F) \xrightarrow[\mathsf{gc}]{} \widetilde{F}) \in \mathsf{Q}$*, then find* X *such that* $\mathsf{gi}(s,X) = \widetilde{X}$ *and output* $y = F^f(X)$.
– *If neither of the above cases happen, then return* $\perp$ *as the answer to* qu.

We will also define the notion of a mixed oracle that uses O on non-eval queries but uses $\texttt{HandleEval}$ to answer eval queries without resorting to O. This oracle is constructed and used in the newly compiled algorithms when we want to avoid asking eval queries to O.

Definition 22 (Mixed oracle). *For an oracle* $O = (f, \mathsf{gc}, \mathsf{gi}, \mathsf{eval})$ *and a set of query-answer pairs* S*, we denote* $O[S]$ *to be an* Eval-mixed *oracle that answers all* f, gc*, and* gi *queries by forwarding them to the real oracle* O*, but for any* eval *query* qu *it will emulate the answer by calling and returning* $y = \texttt{HandleEval}^{f,\mathsf{gi}}(S, qu)$.

Compilation procedure. Let $\mathcal{E} = (G^{f,\mathsf{gc},\mathsf{gi}}, E^{f,\mathsf{gc},\mathsf{gi},\mathsf{eval}}, D^{f,\mathsf{gc},\mathsf{gi},\mathsf{eval}})$ be the give construction for which we want to remove eval queries from E. Without loss of generality, we assume that all the algorithms of $\mathcal{E}$ are in normal form (see Definition 19). For simplicity, we keep O as a superscript to all the algorithms of $\mathcal{E}$, but it is understood that the actual oracle access is of the form above.

We need the following definition as we will need to choose parameters in the compilation construction based on the query complexity of the construction.

Definition 23 (Parameter $q = q(\kappa)$: size-upperbound). *Throughout this section, fix* $q = q(\kappa)$ *to be an arbitrary polynomial that satisfies the following.*

1. $q \geq \kappa$;
2. q *is greater than the total number of queries that each of the algorithms* (G^O, E^O, D^O) *make on inputs corresponding to the security parameter* 1^κ *and on* $O \leftarrow \mathbf{O}$*; and*

3. *q is greater than the size of any query made by any of (G^O, E^O, D^O) on inputs corresponding to the security parameter 1^κ and on $O \leftarrow \mathbf{O}$.*

Construction 24 (compiled scheme $\dot{\mathcal{E}}$) *The compiled scheme $(\dot{G}, \dot{E}, D)$ is parameterized over a function $t = t(\kappa)$, which we will instantiate later.*

- *$\dot{G}(1^\kappa)$: Perform the following steps:*
  1. *Run $(pk, sk) \leftarrow G^O(1^\kappa)$. Add all query-answer pairs generated in this step to OrigG.*
  2. ***Generating helper set H for $\dot{E}$:** Set $\mathsf{LocalE} = \varnothing$.*
     (a) *Do the following t times: Run $E^{O[\mathsf{OrigG}]}(pk, 0)$ and $E^{O[\mathsf{OrigG}]}(pk, 1)$ and keep adding all the resulting query-answer pairs to LocalE.*
     (b) *Set $\mathsf{H} := \mathtt{ConstHelp}(\mathsf{LocalE}, \mathsf{OrigG} \cup \mathsf{LocalE})$.*
  3. *Output $\dot{pk} = (pk, \mathsf{H})$ and $\dot{sk} = sk$.*
- *$\dot{E}(\dot{pk}, b)$: Parse $\dot{pk} = (pk, \mathsf{H})$. Run $\dot{c} \leftarrow E^{O[\mathsf{H}]}(pk, b)$ and add all the query-response pairs to OrigE. Return $\dot{c}$.*

Remark about $\dot{E}$. We note that, by the definition of $O[\mathsf{H}]$, all the eval queries of $E^{O[\mathsf{H}]}(pk, b)$ will be emulated using H. Thus, $\dot{E}$ will not issue any eval queries.

Query complexity of $\dot{\mathcal{E}}$. It is immediate to see that the query complexity of each of the compiled algorithms is polynomial in q and t, where q the query complexity of (G, E, D). Thus, we have the following lemma.

Lemma 25. *Let q be the size-upperbound of (G, E, D) as given in Definition 23. The query complexity of $\dot{\mathcal{E}} = (\dot{G}, \dot{E}, D)$ is at most $q + (2q^2)t \leq 3tq^2$.*

Correctness and security. We now give the correctness and security statements regarding the compiled scheme $\dot{\mathcal{E}} = (\dot{G}, \dot{E}, D)$ and prove them. By doing so, we complete the proof of Lemma 16.

Lemma 26 (Correctness of $\dot{\mathcal{E}}$). *Suppose the original scheme (G, E, D) is $(\frac{1}{2} + \delta)$ correct in the ideal model. The compiled scheme $(\dot{G}^O, \dot{E}^O, D^O)$ has at least $(\frac{1}{2} + \delta - \frac{2q}{t} - \mathrm{negl}(\kappa))$ correctness in the ideal model, where t is the number of iterations performed in $\dot{G}$.*

In particular, for any constant $c > 0$ by taking $t = q^{c+2}$, the compiled scheme $(\dot{G}^O, \dot{E}^O, D^O)$ has at least $(\frac{1}{2} + \delta - \frac{1}{\kappa^c})$ correctness.

Lemma 27 (Security of $\dot{\mathcal{E}}$). *There exists a polynomial-query algorithm $\mathtt{SecRed}$ that satisfies the following. For any adversary $\mathcal{A}$ that breaks the semantic security of $(\dot{G}^O, \dot{E}^O, \dot{D}^O)$ in the ideal model with probability at least γ, the algorithm $\mathtt{SecRed}^{\mathcal{A},O}$ breaks the semantic security of (G^O, E^O, D^O) with probability at least $\gamma - \frac{1}{2^{\kappa/4}} - \frac{1}{\kappa^c}$ for any constant $c > 0$.*

Proof of correctness for $\dot{\mathcal{E}}$. We first prove Lemma 26, which states that $\dot{\mathcal{E}} = (\dot{G}, \dot{E}, D)$ is still a correct PKE after removing the eval queries from E.

Parsing $\dot{pk} = (pk, H)$, recall that $\dot{E}^O(\dot{pk}, b)$ simply runs $E^{O[\mathsf{H}]}(pk, b)$. With this in mind, to prove Lemma 26, we give the following lemma, which shows that the distribution of outpouts of $E^{O[\mathsf{H}]}(pk, b)$ and $E^O(pk, b)$ are close.

Lemma 28. *For $b \in \{0,1\}$ we have*

$$\Pr_{O,r,pk,\mathsf{H}}[E^O(pk, b; r) \neq E^{O[\mathsf{H}]}(pk, b; r)] \leq \frac{2q}{t} + \frac{1}{2^{\kappa/3}}$$

where $O \leftarrow \mathbf{O}$, $((pk, H), sk) \leftarrow \dot{G}^O(1^\kappa)$ and $r \leftarrow \{0,1\}^*$.

We first show how to derive Lemma 26 from Lemma 28.

Proof (of Lemma 26). Parse $\dot{pk} = (pk, \mathsf{H})$. All the probabilities below are taken over the random choices of $\dot{pk}$, O and r. We have

$$\begin{aligned}\Pr[D^O(sk, \dot{E}^O(\dot{pk}, b; r)) \neq b] &= \Pr[D^O(sk, E^{O[\mathsf{H}]}(pk, b; r)) \neq b] \\ &\leq \Pr[D^O(sk, E^O(pk, b; r)) \neq b] + 2q/t + \mathrm{negl}(\kappa) \\ &\leq \frac{1}{2} - \delta + 2q/t + \mathrm{negl}(\kappa)\end{aligned}$$

where the first inequality follows from Lemma 28. □

We now focus on proving Lemma 28. Fix $b \in \{0,1\}$. For compactness, we define the following experiment that outputs some random variables that will be later used to define some events.

Experiment $\mathbf{Expr}(1^\kappa)$ for fixed $b \in \{0,1\}$: Output the tuple of variables $\mathrm{Vars} = (pk, \mathsf{OrigG}, \mathsf{LocalE}, \mathsf{H}, r)$, where $pk, \mathsf{OrigG}, \mathsf{LocalE}$ and H are sampled as in $\dot{G}(1^\kappa)$ and $r \leftarrow \{0,1\}^*$ is the randomness to $\dot{E}(pk, b)$.

We define the following bad events. Note that all these bad events as well as those that appear later are defined based on the output of Vars, and so we make this dependence implicit henceforth.

- Bad_1: The event that $E^O(pk, b; r)$ makes a query $qu = ((\widetilde{F}, \widetilde{X}) \xrightarrow[\mathsf{eval}]{} ?)$, where $((*, *) \xrightarrow[\mathsf{gc}]{} \widetilde{F}) \notin \mathsf{OrigG}$ and $\mathsf{eval}(\widetilde{F}, \widetilde{X}) \neq \perp$.
- Bad_2: The event that the execution of $E^O(pk, b; r)$ queries $qu = ((\widetilde{F}, \widetilde{X}) \xrightarrow[\mathsf{eval}]{} ?)$ for which we have $((*, *) \xrightarrow[\mathsf{gc}]{} \widetilde{F}) \in \mathsf{OrigG}$, $\mathsf{eval}(\widetilde{F}, \widetilde{X}) \neq \perp$ and $O[\mathsf{H}](qu) = \mathtt{HandleEval}(\mathsf{H}, qu) = \perp$.

Roadmap for the proof of Lemma 28. The proof of Lemma 28 now follows from the following lemmas.

Lemma 29. $\Pr_{O,\mathrm{Vars}}[E^{O}(pk,b;r) \neq E^{O[\mathsf{H}]}(pk,b;r)] \leq \Pr[\mathsf{Bad}_1 \vee \mathsf{Bad}_2]$ *where* $O \leftarrow \mathbf{O}$ *and* $\mathrm{Vars} = (pk, \mathsf{OrigG}, \mathsf{LocalE}, \mathsf{H}, r) \leftarrow \mathbf{Expr}(1^\kappa)$.

Lemma 30. $\Pr_{O,\mathrm{Vars}}[\mathsf{Bad}_1] \leq \frac{1}{2^{\kappa/3}}$ *where* $O \leftarrow \mathbf{O}$ *and* $\mathrm{Vars} \leftarrow \mathbf{Expr}(1^\kappa)$.

Lemma 31. $\Pr_{O,\mathrm{Vars}}[\mathsf{Bad}_2 \wedge \overline{\mathsf{Bad}_1}] \leq \frac{2q}{t}$ *where* $O \leftarrow \mathbf{O}$ *and* $\mathrm{Vars} \leftarrow \mathbf{Expr}(1^\kappa)$

The proof of Lemma 28 follows immediately from Lemmas 29, 30, and 31. We now prove all these lemmas below.

Proof (of Lemma 29). Let Bad be the event $E^{O}(pk,b;r) \neq E^{O[\mathsf{H}]}(pk,b;r)$. We show that whenever Bad holds, then either Bad_1 happens or Bad_2 happens, hence proving the lemma. Notice that the only difference between the executions of $E^{O}(pk,b;r)$ and $E^{O[\mathsf{H}]}(pk,b;r)$ is how eval queries are handled. Specifically, in $E^{O[\mathsf{H}]}(pk,b;r)$, the eval queries are simulated with respect to the set H whereas in $E^{O}(pk,b;r)$, the real oracle O is used to reply to these queries. All of f, gc, and gi queries will be handled identically in both experiments by forwarding them to O. Thus, we only need to consider what happens in either execution when a new query $qu = ((\widetilde{F}, \widetilde{X}) \xrightarrow[\mathsf{eval}]{} ?)$ is asked.

Suppose Bad holds and let $qu = ((\widetilde{F}, \widetilde{X}) \xrightarrow[\mathsf{eval}]{} ?)$ be the first eval query that will be answered differently between the two executions. That is, qu will be replied to with $\perp$ under $O[\mathsf{H}]$, but receives an answer $y \neq \perp$ from the real oracle O. We will now show that either Bad_1 or Bad_2 must hold. Consider two cases:

1. $((*,*) \xrightarrow[\mathsf{gc}]{} \widetilde{F}) \notin \mathsf{OrigG}$: In this case, the fact that $\mathsf{eval}(\widetilde{F}, \widetilde{X}) \neq \perp$ implies that Bad_1 holds.
2. $((*,*) \xrightarrow[\mathsf{gc}]{} \widetilde{F}) \in \mathsf{OrigG}$: In this case the facts that $\mathsf{eval}(\widetilde{F}, \widetilde{X}) \neq \perp$, that qu is a query during the execution of $E^{O}(pk,b;r)$, and that $O[H](qu) = \perp$ imply that Bad_2 holds.

□

Proof (of Lemma 30). The proof of this lemma follows by a simple reduction to Lemma 4. Letting $\alpha = \Pr[\mathsf{Bad}_1]$, we will show how to build an adversary $\mathcal{A}^{f,\mathsf{gc},\mathsf{gi}}(1^\kappa)$ in the sense of Lemma 4 that will win with probability $\alpha \cdot \frac{1}{\mathrm{poly}(\kappa)}$.

Let i be the index of the first query qu during the execution of $E^{O}(pk,b;r)$ for which the event Bad_1 holds. Note that up to the query index i, the executions of $E^{O}(pk,b;r)$ and $E^{O[\mathsf{OrigG}]}(pk,b;r)$ are identical. With this in mind, we build the adversary $\mathcal{A}^{f,\mathsf{gc},\mathsf{gi}}(1^\kappa)$ as follows.

The adversary $\mathcal{A}^{f,\mathsf{gc},\mathsf{gi}}(1^\kappa)$ samples $(pk,sk) \leftarrow G^{f,\mathsf{gc},\mathsf{gi}}(1^\kappa)$, forming the set of query/response pairs OrigG. Then $\mathcal{A}^{f,\mathsf{gc},\mathsf{gi}}$ guesses $i \leftarrow [q]$ and runs $E^{O[\mathsf{OrigG}]}(pk,b;r)$ for a random r. Notice that $\mathcal{A}$ makes no queries to eval whatsoever, as it handles eval queries using OrigG. If the ith query of this execution is $((\widetilde{F},*) \xrightarrow[\mathsf{eval}]{} ?)$ for some $\widetilde{F}$, then $\mathcal{A}^{f,\mathsf{gc},\mathsf{gi}}$ returns $\widetilde{F}$; otherwise, $\mathcal{A}$ returns $\perp$.

$\mathcal{A}^{f,\mathsf{gc},\mathsf{gi}}(1^\kappa)$ wins with probability at least $\alpha \cdot \frac{1}{q}$. On the other hand, by Lemma 4 we know $\mathcal{A}$'s success probability is at most $\frac{1}{2^{\kappa/2}}$. Thus, we have $\alpha \leq \frac{1}{2^{\kappa/3}}$, and the proof is complete. □

Proof (of Lemma 31). We claim that whenever the event $\mathsf{Bad}_2 \wedge \overline{\mathsf{Bad}_1}$ holds then the event Miss, defined as follows, also holds. Miss is the event that during the execution of $E^{O[\mathsf{OrigG}]}(pk, b; r)$ there is a query $qu = ((\widetilde{F}, \widetilde{X}) \xrightarrow[\mathsf{eval}]{} ?)$, such that

1. $((*,*) \xrightarrow[\mathsf{gc}]{} \widetilde{F}) \in \mathsf{OrigG}$;
2. $\mathsf{eval}(\widetilde{F}, \widetilde{X}) \neq \perp$;
3. $((*,*) \xrightarrow[\mathsf{gc}]{} \widetilde{F}) \notin \mathsf{H}$ and $((\widetilde{F}, \widetilde{X}) \xrightarrow[\mathsf{eval}]{} *) \notin \mathsf{H}$.

The reason for the above claim is that if $\mathsf{Bad}_2 \wedge \overline{\mathsf{Bad}_1}$ holds, then $\overline{\mathsf{Bad}_1}$ must necessarily hold, and thus the two executions $E^{O}(pk, b; r)$ and $E^{O[\mathsf{OrigG}]}(pk, b; r)$ are identical. The rest follows by the definition of the event Bad_2. We will prove

$$\Pr[\mathsf{Miss}] \leq \frac{2q}{t}, \tag{5}$$

which yields the proof of this lemma. Thus, we focus on proving Equation 5.

We break Miss into smaller events. We give some notation first. Let $i \in [n]$, $d \in \{0,1\}$, F be circuit with input size n and let $\widetilde{F} = \mathsf{gc}(s, F)$, for some s. We say a garbled input $\widetilde{X} = (\widetilde{x}_1, \ldots, \widetilde{x}_n)$ is an (i,d)-*match* for $\widetilde{F}$ if $\widetilde{X}$ is a valid garbled input of $\widetilde{F}$ and the ith garbled bit of $\widetilde{X}$ corresponds to the bit d. Formally,

- for all $j \in [n]$ and $j \neq i$: $\widetilde{x}_j = \mathsf{gi}(s, j, 0)$ or $\widetilde{x}_j = \mathsf{gi}(s, j, 1)$;
- $\widetilde{x}_i = \mathsf{gi}(s, i, d)$.

We say that a set of query/response pairs U *contains an* (i,d)*-match* for $\widetilde{F}$ if there exists $((\widetilde{F}, \widetilde{X}) \xrightarrow[\mathsf{eval}]{} *) \in \mathsf{U}$ such that $\widetilde{X}$ is an (i,d)-match for $\widetilde{F}$.

We also give the following notation. Recalling the way in which LocalE is constructed in $\dot{G}$ through t iterations, for $i \in [t]$ let LocalE_i be the set formed after the i-th iteration. Also, let OrigE^* be the set of all query/response pairs during the execution of $E^{O[\mathsf{OrigG}]}(pk, b; r)$.

We now define a series of events, $\mathsf{Miss}_{i,d}$, for $i \in [q]$ and $d \in \{0,1\}$, and will show that if Miss holds then for some i and d the event $\mathsf{Miss}_{i,d}$ must hold.

Event $\mathsf{Miss}_{i,d}$ is the event that for some $\widetilde{F}$ that $((*,*) \xrightarrow[\mathsf{gc}]{} \widetilde{F}) \in \mathsf{OrigG}$, both the following hold:

1. OrigE^* contains an (i,d)-match for $\widetilde{F}$;
2. none of the sets $\mathsf{LocalE}_1, \cdots, \mathsf{LocalE}_t$ do contain an (i,d)-match for $\widetilde{F}$.

We claim that if Miss holds then $\mathsf{Miss}_{i,d}$ must hold, for some $i \in [q]$ and $d \in \{0,1\}$. Suppose the event Miss holds for the query $qu = ((\widetilde{F}, \widetilde{X}) \xrightarrow[\mathsf{eval}]{} ?)$ (see above for the definition of Miss). We consider all possible cases:

- For no (i,d) does the set $\mathsf{LocalE} = \mathsf{LocalE}_1 \cup \cdots \cup \mathsf{LocalE}_t$ contain an (i,d)-match for $\widetilde{F}$. Since the set OrigE^* contains $((\widetilde{F}, \widetilde{X}) \xrightarrow[\mathsf{eval}]{} *)$, there would be an (i,d)-match for $\widetilde{F}$ for all $i \in [q]$, so $\mathsf{Miss}_{i,d}$ holds for some d and all $i \in [q]$.

– There is one and only one garbled input $\widetilde{X_1}$ which is valid for $\widetilde{F}$ and for which we have $((\widetilde{F}, \widetilde{X_1}) \xrightarrow[\mathsf{eval}]{} ?) \in \mathsf{LocalE}$. In this case, we must have $\widetilde{X_1} \neq \widetilde{X}$, because otherwise we would have $((\widetilde{F}, \widetilde{X}) \xrightarrow[\mathsf{eval}]{} *) \in \mathsf{H}$, a contradiction to the fact that Miss holds. Thus, for some (i, d) both the following must hold: (A) $\widetilde{X}$ is an (i, d)-match for $\widetilde{F}$ and (B) $\widetilde{X_1}$ is not an (i, d)-match for $\widetilde{F}$. Thus, for some i and d, the event $\mathsf{Miss}_{i,d}$ must hold.
– There are at least two different garbled inputs $\widetilde{X_1}$ and $\widetilde{X_2}$ which both are valid for $\widetilde{F}$ and which $((\widetilde{F}, \widetilde{X_1}) \xrightarrow[\mathsf{eval}]{} ?) \in \mathsf{LocalE}$ and $((\widetilde{F}, \widetilde{X_2}) \xrightarrow[\mathsf{eval}]{} ?) \in \mathsf{LocalE}$: This case cannot happen because otherwise we would have $(*, * \xrightarrow[\mathsf{gc}]{} \widetilde{F}) \in \mathsf{H}$, a contradiction to the fact that Miss holds.

Having proved $\Pr[\mathsf{Miss}] \leq \sum_{i,d} \Pr[\mathsf{Miss}_{i,d}]$, we bound the probability of each individual $\mathsf{Miss}_{i,d}$. To bound the probability of the event $\mathsf{Miss}_{i,d}$, note that since all of $\mathsf{LocalE}_1, \ldots, \mathsf{LocalE}_t$ and OrigE^* are obtained via independent and identical processes, by Lemma 3.1 we have

$$\Pr[\mathsf{Miss}_{i,d}] \leq \frac{1}{t}.$$

Using a union bound, $\Pr[\mathsf{Miss}] \leq \frac{2q}{t}$, and Equation 5 is now proved. This completes the proof. □

Proof of security for $\dot{\mathcal{E}}$. We now give the proof of security.

Proof (of Lemma 27). To define the reduction algorithm `SecRed` we need to introduce the following procedure, overloading the definition of `ConstHelp` (Definition 20). In Definition 20 the procedure `ConstHelp` was given as input an auxiliary information set Aux which helps the procedure in finding answers to the eval queries provided in the given set Q. In the definition below, however, there is no auxiliary information set, but the procedure could use the oracle rev.

Definition 32. *Procedure* `ConstHelp`*:*

– ***Input:*** *A set of query/answer pairs* Q.
– ***Oracle:*** $O = (f, \mathsf{gc}, \mathsf{gi}, \mathsf{eval}, \mathsf{rev})$.
– ***Output:*** *A "hint" set* H *formed as follows:*
  • *If there exists* $((\widetilde{F}, \widetilde{X}) \xrightarrow[\mathsf{eval}]{} y \neq \bot) \in \mathsf{Q}$ *such that for no* $\widetilde{X}' \neq \widetilde{X}$ *do we have* $((\widetilde{F}, \widetilde{X}') \xrightarrow[\mathsf{eval}]{} y' \neq \bot) \in \mathsf{Q}$*, then add* $((\widetilde{F}, \widetilde{X}) \xrightarrow[\mathsf{eval}]{} y)$ *to* H.
  • *If for two distinct* $\widetilde{X}_1$ *and* $\widetilde{X}_2$ *we have* $((\widetilde{F}, \widetilde{X}_1) \xrightarrow[\mathsf{eval}]{} y_1 \neq \bot) \in \mathsf{Q}$ *and* $((\widetilde{F}, \widetilde{X}_2) \xrightarrow[\mathsf{eval}]{} y_2 \neq \bot) \in \mathsf{Q}$*, then add* $(((s, F)) \xrightarrow[\mathsf{gc}]{} \widetilde{F})$ *to* H*, where* $(s, F) = \mathsf{rev}(\widetilde{F}, \widetilde{X_1}, \widetilde{X_2})$.

We will now describe the attack oracle-aided algorithm $\mathtt{SecRed}$ against the semantic security of (G^O, E^O, D^O). The input to $\mathtt{SecRed}$ is pair of challenge (pk, c) sampled under $\mathcal{E}^O$. Moreover, $\mathtt{SecRed}$ has oracle access to O as well as an adversary against $\dot{\mathcal{E}}^O$.

Description of $\mathtt{SecRed}^{\mathcal{A},O}(pk, c)$:

1. Initialize $\mathsf{LocalE}^* = \varnothing$. For $i = [1, t]$, do the following: Run $E^O(pk, 0)$ and $E^O(pk, 1)$ and add all the resulting query-answer pairs to LocalE^*.
2. Set $\mathsf{H}^* \leftarrow \mathtt{ConstHelp}^O(\mathsf{LocalE}^*)$.
3. Return $b' \leftarrow \mathcal{A}(pk, \mathsf{H}^*, c)$.

We will now show that the following holds for both $b = 0$ and $b = 1$: The distribution $\text{Dist1} = (pk, \mathsf{H}^*, c)$ is statistically close to $\text{Dist2} = (\dot{pk}, \dot{c})$, where $(pk, sk) \leftarrow G^O(1^\kappa)$, $c \leftarrow E^O(pk, b)$, and $(\dot{pk}, *) \leftarrow \dot{G}^O(1^\kappa)$ and $\dot{c} \leftarrow \dot{E}^O(\dot{pk}, b)$. Also, H^* is sampled as in the execution of the security reduction $\mathtt{SecRed}^{\mathcal{A},O}(pk, c)$. Let all the variables that appear below be sampled as in the above. First, it is easy to show that

$$\Delta((pk, \mathsf{H}^*), \dot{pk}) \leq \text{poly}(\kappa) \times \frac{1}{2^{\kappa/2}} \leq \frac{1}{2^{\kappa/3}}.$$

Moreover, by Lemma 29 we have

$$\Delta(c, \dot{c}) \leq \frac{2q}{t} + \frac{1}{2^{\kappa/3}}. \tag{6}$$

Thus, $\mathtt{SecRed}^{\mathcal{A},O}(pk, c)$ breaks the semantic security of (G^O, E^O, D^O) with probability at least $\gamma - \frac{2q}{t} - \frac{1}{2^{\kappa/4}}$. □

4.2 Removing Garbling Evaluation Queries from Decryption

In this section, we will prove Lemma 17. Namely, we will present a procedure that compiles a PKE scheme $\mathcal{E} = (G^{f,\mathsf{gc},\mathsf{gi}}, E^{f,\mathsf{gc},\mathsf{gi}}, D^{f,\mathsf{gc},\mathsf{gi},\mathsf{eval}})$ into a new PKE scheme $\ddot{\mathcal{E}} = (\ddot{G}^{f,\mathsf{gc},\mathsf{gi}}, \ddot{E}^{f,\mathsf{gc},\mathsf{gi}}, \ddot{D}^{f,\mathsf{gc},\mathsf{gi}})$ with correctness and security comparable to the original scheme $\mathcal{E}$, but where $\ddot{D}$ will not ask any eval queries.

Again, for simplicity we use the following convention where we keep the entire oracle O as a superscript to all the algorithms (G^O, E^O, D^O) as well as $(\ddot{G}^O, \ddot{E}^O, \ddot{D}^O)$ with the understanding that the actual oracle access is of the form given above. We also make the following assumption without loss of generality.

Assumption 33 *We assume that all the algorithms (G, E, D) are in normal form (Definition 19). Also, we assume w.l.o.g. that the secret key outputted by G contains all the query-response pairs made by G.*

Definition 34 (Query set). *For an oracle algorithm A^O, $\text{Query}(A^O(x; r))$ denotes the set of all queries asked during the execution of A^O on input x and randomness r. We write $\text{Query}(A^O(x))$ to indicate the random variable formed by returning $\text{Query}(A^O(x; r))$ for $r \leftarrow \{0,1\}^*$.*

Definition 35 (Valid partial oracles). *We say that a partial oracle O_1 is valid if for some $O_2 \in \mathrm{Supp}(\mathbf{O})$: $O_1 \subseteq O_2$.*

Definition 36 (Oracle consistency/sampling notation). *We say a partial oracle O_1 is consistent with a set of query/response pairs S if $O_1 \cup \mathsf{S}$ is valid.*

For a partial oracle O_1 and randomness r we say that (O_1, r) agrees with a public key pk if (1) $G^{O_1}(r) = (pk, *)$ and (2) all the queries in $\mathrm{Query}(G^{O_1}(r))$ are defined in O_1. We say that (O_1, r) minimally agrees with pk if (1) (O_1, r) agrees with pk and (2) O_1 is defined only on the queries that occur during the execution and nothing more: namely, $O_1(qu)$ is defined iff $qu \in \mathrm{Query}(G^{O_1}(r))$.

We let $\mathrm{Partial}(pk, \mathsf{S})$ denote the set of all (O_1, r) where (1) (O_1, r) minimally agrees with pk and (2) O_1 agrees with S. We sometimes abuse notation and write $(O_1, sk) \leftarrow \mathrm{Partial}(pk, \mathsf{S})$ to mean the following sampling: $(O_1, r) \leftarrow \mathrm{Partial}(pk, \mathsf{S})$ and $(pk, sk) = G^{O_1}(r)$.

Definition 37 (Composed oracle). *Given a partial oracle O_p and full oracle O (an oracle that is defined on all points in its domain) we define $O_p \Diamond O$ to be the* composed oracle *that uses O_p to reply if the corresponding query is defined there, and uses O otherwise. Note that $O_p \Diamond O$ is not necessarily in $\mathrm{Supp}(O)$.*

Compilation procedure.

Construction 38 *The scheme $\ddot{\mathcal{E}} = (\ddot{G}, \ddot{E}, \ddot{D})$ is parameterized over two functions $\varepsilon = \varepsilon(\kappa)$ and $t = t(\kappa)$, which we will instantiate later.*

- $\ddot{G}(1^\kappa)$: *Do the following steps:*
  1. *Set $\mathsf{OrigG} = \varnothing$. Run $(pk, sk) \leftarrow G^O(1^\kappa)$, and add all query-answer pairs that are encountered during this execution to OrigG.*
  2. *Set $\mathsf{LearnG} = \varnothing$. While there exists a query $qu \notin \mathsf{LearnG}$ such that*

$$\Pr_{O' \leftarrow \mathbf{O}}[qu \in \mathrm{Query}(G^{O'}(1^\kappa)) \mid pk, \mathsf{LearnG}] \geq \varepsilon,$$

  then choose the lexicographically first such qu and add $(qu \xrightarrow[T]{} O'(qu))$ to LearnG. Note that $T \in \{f, \mathsf{gc}, \mathsf{gi}\}$.
  3. *Output $\ddot{pk} = (pk, \mathsf{LearnG})$ and $\ddot{sk} = sk$. (By Assumption 33, $\ddot{sk}$ contains OrigG.)*
- $\ddot{E}(\ddot{pk}, b)$: *Given $\ddot{pk} = (pk, \mathsf{LearnG})$ and $b \in \{0, 1\}$ do the following:*
  1. *Set $\mathsf{OrigE} = \varnothing$. Run $c \leftarrow E^O(pk, b)$ and add all the query-answer pairs that are observed during this execution to OrigE.*
  2. ***Generating helper set** H **for** $\ddot{D}$:* *Sample $t' \leftarrow [1, t]$. Set $\mathsf{S} = \mathsf{OrigE} \cup \mathsf{LearnG}$. For $i \in [1, t']$, do the following:*
     (a) ***Offline phase:*** *Sample $(\widehat{O}_i, \widehat{sk_i}) \leftarrow \mathrm{Partial}(pk, S)$.*
     (b) ***Semi-online phase:*** *Run $D^{\widehat{O}_i \Diamond O[\mathsf{S}]}(\widehat{sk_i}, c)$ and add all query/response pairs made to the oracle O to the set S. Let $\widehat{\mathsf{OrigD}_i}$ be the set of all query-answer pairs made by this execution.*

After all iterations, set $\mathsf{H} := \mathtt{ConstHelp}(\widehat{\mathsf{OrigD}}, \mathsf{S})$*, where we define* $\widehat{\mathsf{OrigD}} = \widehat{\mathsf{OrigD}}_1 \cup \cdots \cup \widehat{\mathsf{OrigD}}_{t'}$.

3. *Output* $\ddot{c} = (c, \mathsf{H})$.

– $\ddot{D}(\ddot{pk}, \ddot{sk}, \ddot{c})$: *Given* $\ddot{pk} = (pk, \mathsf{LearnG})$, $\ddot{sk}$, *and* $\ddot{c} = (c, \mathsf{H})$, *output the value of* $\widetilde{b} \leftarrow D^{O[\mathsf{H} \cup \mathsf{LearnG}]}(\ddot{sk}, c)$.

Query complexity of $\ddot{\mathcal{E}}$. The following lemma follows from the description of the compilation procedure of Construction 38.

Lemma 39. *Let q be as in Definition 23. Assuming $\varepsilon = \frac{1}{\mathrm{poly}(\kappa)}$ and $t = \mathrm{poly}(\kappa)$, all the algorithms of $\ddot{\mathcal{E}}^O$ make $q^{O(1)}$ queries. Concretely, the algorithm $\ddot{E}$ makes at most $\nu := 4tq^2$ queries.*

We note that by taking $\varepsilon = \frac{1}{\mathrm{poly}(\kappa)}$ the learning process of $\ddot{G}$ (i.e., for sampling LearnG) could be done by making a polynomial number of queries [6].

We give the correctness and security statements regarding the compiled scheme $\ddot{\mathcal{E}} = (\ddot{G}, \ddot{E}, \ddot{D})$. See the full version for the proofs.

Lemma 40 (Correctness of $\ddot{\mathcal{E}}$). *Suppose the original PKE scheme (G, E, D) is $(\frac{1}{2} + \delta)$-correct in the ideal model. The compiled PKE scheme $(\ddot{G}, \ddot{E}, \ddot{D})$ has at least $(\frac{1}{2} + \delta - \eta)$ correctness in the ideal model, where*

$$\eta = \frac{1}{2^{\kappa/5}} + \frac{2q}{t} + 3\varepsilon\nu t.$$

That is,

$$\Pr[\ddot{D}^O(sk, \ddot{c}) \neq b] \leq \tfrac{1}{2} - \delta + \eta \tag{7}$$

where the probability is taken over $O \leftarrow \mathbf{O}$, $(\ddot{pk}, sk) \leftarrow \ddot{G}^O(1^\kappa)$, $b \leftarrow \{0,1\}$ and $\ddot{c} = (c, \mathsf{H}) \leftarrow \ddot{E}^O(\ddot{pk}, b)$. Here t and ε are the underlying parameters of the compilation procedure, and ν is defined in Lemma 39. In particular, for any constant $c > 0$ by taking $t = 2q^{c+2}$ and $\varepsilon = \frac{1}{q^{3c+8}}$, the compiled scheme $(\ddot{G}, \ddot{E}, \ddot{D})$ has at least $(\frac{1}{2} + \delta - 1/\kappa^c)$ correctness in the ideal model.

Lemma 41 (Security of $\ddot{\mathcal{E}}$). *Let p be an arbitrary polynomial which satisfies*

$$8tq^2\varepsilon + \tfrac{1}{2^{\kappa/2-1}} \leq \tfrac{1}{p}.$$

There exists a polynomial-query algorithm $\mathtt{SecRed}$ *that satisfies the following. For any adversary $\mathcal{A}$ that breaks the semantic security of $(\ddot{G}^O, \ddot{E}^O, \ddot{D}^O)$ in the ideal model O with probability at least γ, the adversary $\mathtt{SecRed}^{\mathcal{A},O}$ breaks the semantic security of (G^O, E^O, D^O) with probability at least $\gamma - \beta$ where*

$$\beta = t \cdot (\tfrac{1}{p-1} + 4tq^2\varepsilon + \tfrac{1}{q^{2c+4}} + \tfrac{1}{2^{\kappa/2-1}}).$$

In particular, for any constant c by taking $t = 2q^{c+2}$ and $\varepsilon = \frac{1}{q^{3c+8}}$ we have the following: For any $\mathcal{A}$ breaking the semantic security of $(\ddot{G}^O, \ddot{E}^O, \ddot{D}^O)$ in the ideal model with probability at least γ, the (polynomial-query) adversary $\mathtt{SecRed}^{\mathcal{A},O}$ breaks the semantic security of (G^O, E^O, D^O) with probability at least $\gamma - \frac{22}{\kappa^{2+c}}$.

References

1. Benny Applebaum. Garbled circuits as randomized encodings of functions: a primer. In *Tutorials on the Foundations of Cryptography*, pages 1–44. Springer, 2017. 2
2. Gilad Asharov and Gil Segev. Limits on the power of indistinguishability obfuscation and functional encryption. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pages 191–209. IEEE, 2015. 3, 5, 9
3. Gilad Asharov and Gil Segev. On constructing one-way permutations from indistinguishability obfuscation. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A: 13th Theory of Cryptography Conference, Part II*, volume 9563 of *Lecture Notes in Computer Science*, pages 512–541, Tel Aviv, Israel, January 10–13, 2016. Springer, Heidelberg, Germany. 3
4. Paul Baecher, Christina Brzuska, and Marc Fischlin. Notions of black-box reductions, revisited. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 296–315. Springer, 2013. 2, 12
5. Boaz Barak. The complexity of public-key cryptography. In *Tutorials on the Foundations of Cryptography*, pages 45–77. Springer, 2017. 2
6. Boaz Barak and Mohammad Mahmoody-Ghidary. Lower bounds on signatures from symmetric primitives. In *48th Annual Symposium on Foundations of Computer Science*, pages 680–688, Providence, RI, USA, October 20–23, 2007. IEEE Computer Society Press. 27
7. Boaz Barak and Mohammad Mahmoody-Ghidary. Merkle puzzles are optimal - an $O(n^2)$-query attack on any key exchange from a random oracle. In Shai Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 374–390, Santa Barbara, CA, USA, August 16–20, 2009. Springer, Heidelberg, Germany. 16
8. Donald Beaver. Correlated pseudorandomness and the complexity of private computations. In *28th Annual ACM Symposium on Theory of Computing*, pages 479–488, Philadephia, PA, USA, May 22–24, 1996. ACM Press. 3
9. Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In Ting Yu, George Danezis, and Virgil D. Gligor, editors, *ACM CCS 12: 19th Conference on Computer and Communications Security*, pages 784–796, Raleigh, NC, USA, October 16–18, 2012. ACM Press. 2, 3, 4
10. Nir Bitansky, Akshay Degwekar, and Vinod Vaikuntanathan. Structure vs. hardness through the obfuscation lens. In *Annual International Cryptology Conference*, pages 696–723. Springer, 2017. 3
11. Dan Boneh, Periklis A. Papakonstantinou, Charles Rackoff, Yevgeniy Vahlis, and Brent Waters. On the impossibility of basing identity based encryption on trapdoor permutations. In *49th Annual Symposium on Foundations of Computer Science*, pages 283–292, Philadelphia, PA, USA, October 25–28, 2008. IEEE Computer Society Press. 2
12. Zvika Brakerski, Jonathan Katz, Gil Segev, and Arkady Yerukhimovich. Limits on the power of zero-knowledge proofs in cryptographic constructions. In *Theory of Cryptography Conference*, pages 559–578. Springer, 2011. 3, 4, 5
13. Ran Canetti, Yael Tauman Kalai, and Omer Paneth. On obfuscation with random oracles. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015: 12th Theory of Cryptography Conference, Part II*, volume 9015 of *Lecture Notes in Computer Science*, pages 456–467, Warsaw, Poland, March 23–25, 2015. Springer, Heidelberg, Germany. 6

14. Dana Dachman-Soled. Towards non-black-box separations of public key encryption and one way function. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B: 14th Theory of Cryptography Conference, Part II*, volume 9986 of *Lecture Notes in Computer Science*, pages 169–191, Beijing, China, October 31 – November 3, 2016. Springer, Heidelberg, Germany. 5
15. Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976. 1
16. Nico Döttling and Sanjam Garg. Identity-based encryption from the Diffie-Hellman assumption. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part I*, volume 10401 of *Lecture Notes in Computer Science*, pages 537–569, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Heidelberg, Germany. 2, 4
17. Sanjam Garg, Steve Lu, Rafail Ostrovsky, and Alessandra Scafuro. Garbled RAM from one-way functions. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th Annual ACM Symposium on Theory of Computing*, pages 449–458, Portland, OR, USA, June 14–17, 2015. ACM Press. 3
18. Sanjam Garg, Mohammad Mahmoody, and Ameer Mohammed. Lower bounds on obfuscation from all-or-nothing encryption primitives. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part I*, volume 10401 of *Lecture Notes in Computer Science*, pages 661–695, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Heidelberg, Germany. 5, 6
19. Sanjam Garg, Mohammad Mahmoody, and Ameer Mohammed. When does functional encryption imply obfuscation? In *TCC 2017: 15th Theory of Cryptography Conference, Part I*, Lecture Notes in Computer Science, pages 82–115. Springer, Heidelberg, Germany, March 2017. 5
20. Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In Lance Fortnow and Salil P. Vadhan, editors, *43rd Annual ACM Symposium on Theory of Computing*, pages 99–108, San Jose, CA, USA, June 6–8, 2011. ACM Press. 5
21. Yael Gertner, Tal Malkin, and Omer Reingold. On the impossibility of basing trapdoor functions on trapdoor predicates. In *42nd Annual Symposium on Foundations of Computer Science*, pages 126–135, Las Vegas, NV, USA, October 14–17, 2001. IEEE Computer Society Press. 7
22. Chun-Yuan Hsiao and Leonid Reyzin. Finding collisions on a public road, or do secure hash functions need secret coins? In Matthew Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 92–105, Santa Barbara, CA, USA, August 15–19, 2004. Springer, Heidelberg, Germany. 7
23. Russell Impagliazzo and Steven Rudich. Limits on the provable consequences of one-way permutations. In *21st Annual ACM Symposium on Theory of Computing*, pages 44–61, Seattle, WA, USA, May 15–17, 1989. ACM Press. 2, 5, 6, 7, 8, 10, 12, 15, 16
24. Yuval Ishai and Eyal Kushilevitz. Randomizing polynomials: A new representation with applications to round-efficient secure computation. In *41st Annual Symposium on Foundations of Computer Science*, pages 294–304, Redondo Beach, CA, USA, November 12–14, 2000. IEEE Computer Society Press. 2
25. Yehuda Lindell and Benny Pinkas. A proof of security of Yao's protocol for two-party computation. *Journal of Cryptology*, 22(2):161–188, April 2009. 2
26. Steve Lu and Rafail Ostrovsky. How to garble RAM programs. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*,

volume 7881 of *Lecture Notes in Computer Science*, pages 719–734, Athens, Greece, May 26–30, 2013. Springer, Heidelberg, Germany. 3

27. Mohammad Mahmoody, Ameer Mohammed, and Soheil Nematihaji. On the impossibility of virtual black-box obfuscation in idealized models. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A: 13th Theory of Cryptography Conference, Part I*, volume 9562 of *Lecture Notes in Computer Science*, pages 18–48, Tel Aviv, Israel, January 10–13, 2016. Springer, Heidelberg, Germany. 6
28. Mohammad Mahmoody, Ameer Mohammed, Soheil Nematihaji, Rafael Pass, and Abhi Shelat. Lower bounds on assumptions behind indistinguishability obfuscation. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A: 13th Theory of Cryptography Conference, Part I*, volume 9562 of *Lecture Notes in Computer Science*, pages 49–66, Tel Aviv, Israel, January 10–13, 2016. Springer, Heidelberg, Germany. 6
29. Periklis A. Papakonstantinou, Charles W. Rackoff, and Yevgeniy Vahlis. How powerful are the DDH hard groups? Cryptology ePrint Archive, Report 2012/653, 2012. http://eprint.iacr.org/2012/653. 2, 4
30. Rafael Pass. Limits of provable security from standard assumptions. In Lance Fortnow and Salil P. Vadhan, editors, *43rd Annual ACM Symposium on Theory of Computing*, pages 109–118, San Jose, CA, USA, June 6–8, 2011. ACM Press. 5
31. Rafael Pass, Wei-Lung Dustin Tseng, and Muthuramakrishnan Venkitasubramaniam. Towards non-black-box lower bounds in cryptography. In Yuval Ishai, editor, *TCC 2011: 8th Theory of Cryptography Conference*, volume 6597 of *Lecture Notes in Computer Science*, pages 579–596, Providence, RI, USA, March 28–30, 2011. Springer, Heidelberg, Germany. 5
32. Omer Reingold, Luca Trevisan, and Salil P. Vadhan. Notions of reducibility between cryptographic primitives. In Moni Naor, editor, *TCC 2004: 1st Theory of Cryptography Conference*, volume 2951 of *Lecture Notes in Computer Science*, pages 1–20, Cambridge, MA, USA, February 19–21, 2004. Springer, Heidelberg, Germany. 2, 3, 12
33. Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signature and public-key cryptosystems. *Communications of the Association for Computing Machinery*, 21(2):120–126, 1978. 1
34. Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In David B. Shmoys, editor, *46th Annual ACM Symposium on Theory of Computing*, pages 475–484, New York, NY, USA, May 31 – June 3, 2014. ACM Press. 7
35. Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th Annual Symposium on Foundations of Computer Science*, pages 162–167, Toronto, Ontario, Canada, October 27–29, 1986. IEEE Computer Society Press. 2