On Obfuscation with Random Oracles

Ran Canetti[*] Yael Tauman Kalai[†] Omer Paneth[‡]

January 20, 2015

Abstract

Assuming trapdoor permutations, we show that there exist function families that cannot be VBB-obfuscated even if both the obfuscator and the obfuscated program have access to a random oracle. Specifically, these families are the robust unobfuscatable families of [Bitansky-Paneth, STOC 13].

Our result stands in contrast to the general VBB obfuscation algorithms in more structured idealized models where the oracle preserves certain algebraic homomorphisms [Canetti-Vaikuntanathan, ePrint 13; Brakerski-Rothblum, TCC 14; Barak et al., Eurocrypt 14].

[*]Boston University and Tel Aviv University. Email: `canetti@bu.edu`. Supported by the Check Point Institute for Information Security, an ISF grant 20006317, an NSF EAGER grant, and an NSF Algorithmic foundations grant 1218461.

[†]Microsoft Research

[‡]Boston University. Email: `omer@bu.edu`. Supported by the Simons award for graduate students in theoretical computer science and an NSF Algorithmic foundations grant 1218461.

1 Introduction

Program obfuscators, namely efficient compilers that transform an arbitrary program into one that has the same functionality but is otherwise "impenetrable", are an intriguing concept. The widely applicable interpretation of "impenetrable," called virtual black-box (VBB) [BGI$^+$01], requires that the obfuscated version of a program helps learn any predicate of the program no more than does oracle access to the program's input-output functionality.

While a number of program families of interest are known to be VBB obfuscatable (under some strong hardness assumptions), e.g. [Can97, Wee05, BCKP14], no general-purpose VBB-obfuscators of all programs can exist. Indeed [BGI$^+$01] show that, assuming one way functions, there exist *unobfuscatable functions*. These are functions that have a succinct description that cannot be effectively learned when having only oracle access to the function. At the same time, however, this succinct description can be extracted from *any* program that computes the function. Clearly, no program that computes such a function can possibly be VBB-obfuscated.

The construction of [BGI$^+$01] makes crucial use of the fact that programs can be represented as strings and in particular can be executed with their own specification as input. In contrast, in some abstract models where programs do not necessarily have succinct representations as strings VBB obfuscation is in fact obtainable. One example is "hardware assisted" obfuscation, where some part of the computation is modeled as a black-box representing impenetrable secure hardware [GIS$^+$10, BCG$^+$11].

Another example is motivated by the recent candidate construction of obfuscation for all circuits of Garg et. al. [GGH$^+$13b], that is based on an algebraic primitive called graded encodings [GGH13a]. The works of [BR14, BGK$^+$14] prove that close variants of the proposed candidate are VBB secure in a model where the graded encodings are implemented by an ideal oracle. [CV13] study a different construction based on ideal pseudofree groups. Here, idealized models serve as an intermediate steps on the way to full-fledged obfuscation, namely as a model for developing potentially viable obfuscation algorithms and for understanding their security properties, as well as the computational assumptions on which their security might be based.

This raises natural questions: What are the simplest and minimally-structured abstract models that allow for general-purpose VBB obfuscation? For instance, do general-purpose VBB obfuscators exist in the random-oracle model? Do they exist in the generic group model [Sho97, BS84]? In fact, is there *any* non-trivial abstract model of computation where general-purpose VBB obfuscation is impossible?

Answers to the above question may shed light on what algebraic structure (if any) is inherent for secure obfuscation — even in the plain model, and even when attempting to obtain only weaker notions of obfuscation such as indistinguishability obfuscation.

We note that Barak et al. show that their impossiblity holds even when all entities, namely the program to be obfuscated, the obfuscator and the obfuscated program have access to a random oracle. [1] Goldwaser and Rothblum [GR14] extend this to show that even the considerably weaker notion of *indistinguishability obfuscation* is unobtainable in general in this setting. However, these results do not answer the above questions. Specifically, they leave open the possibility of obfuscating fully specified programs that do not access the random oracle. Indeed, Lynn et al. ask whether general purpose obfuscation is possible in that setting [LPS04].

1.1 This Work

We consider obfuscation in the setting of Lynn et al. [LPS04], where both the obfuscator and the obfuscated program have access to a random oracle, and where the obfuscator is only required to operate on fully specified programs that do not have access to the random oracle. Furthermore, we give the adversary access to the same oracle. Here we show:

[1] In fact, [BGI$^+$01] prove that their negative result holds in the more general settings of *bounded relativization*.

Theorem 1.1 (Main Theorem, informal)**.** *Assume trapdoor permutations exist. Then there exist function families that cannot be VBB obfuscated, even in a model where the obfuscator and the obfuscated function have access to a random oracle.*

Our impossibility extends to the case where the obfuscator and obfuscated program have access to an *invertible random permutation* rather than a random function. That is, the oracle represents a random permutation, and can be asked both to evaluate the function and to invert it. It also extends to the case of approximate obfuscation, where the obfuscated program is only required to agree with the original program on a significant fraction of the inputs.

1.2 Techniques.

The starting point of our proof is the existence of *robust unobfuscatable functions (RUFs)* which are a strengthening of the *unobfuscatable functions* of [BGI$^+$01]. Essentially, RUFs have a succinct description that cannot be effectively learned having only oracle access to the function. At the same time, this description can be extracted from any program that *approximates* the function, namely agrees with the function on some large fraction of the inputs, say 90%. Bitansky and Paneth [BP13] construct RUFs from any trapdoor permutation.

Our proof now proceeds by transforming any obfuscator in the RO model into an obfuscator in the plain model, namely one where the RO is not used. The transformation loses in correctness: the resulting plain-model obfuscator generates a program that computes the function correctly only on some fraction of the inputs. Still, impossibility is demonstrated by applying the transformation to an obfuscator for a family of RUFs.

We describe in more detail the transformation from obfuscation in the RO model to obfuscation in the plain model. Let $\mathcal{O}^{\mathsf{R}}$ be an obfuscator in the RO model. Our goal is to transform $\mathcal{O}^{\mathsf{R}}$ into an obfuscator $\mathcal{O}$ in the plain model. We start by describing a simple warm-up. Let $\mathcal{O}$ be the following plain-model obfuscator: given a description of a program C, the obfuscator $\mathcal{O}$ emulates an execution of the RO obfuscator $\mathcal{O}^{\mathsf{R}}(C)$, answering every oracle query of $\mathcal{O}^{\mathsf{R}}$ randomly and independently (repeated queries are answered consistently), and obtains a RO obfuscation $\tilde{C}^{\mathsf{R}}$ of C. Let $\mathcal{R}_C$ be the set of RO query-answer pairs that occurred during the emulation of $\mathcal{O}^{\mathsf{R}}(C)$. The obfuscator $\mathcal{O}$ then outputs an obfuscated program $\tilde{C}$ that has hard-coded to it the description of the RO obfuscation $\tilde{C}^{\mathsf{R}}$ and the set $\mathcal{R}_C$. Given an input x, the obfuscation $\tilde{C}$ emulates the RO obfuscation $\tilde{C}^{\mathsf{R}}(x)$. $\tilde{C}$ answers any RO query made by $\tilde{C}^{\mathsf{R}}$ as follows: if the query appears in the set $\mathcal{R}_C$ it is answered consistently with $\mathcal{R}_C$, otherwise, a random answer is given.[2]

The correctness of $\mathcal{O}$ follows directly from the correctness of $\mathcal{O}^{\mathsf{R}}$ in the RO model since, when $\tilde{C}$ emulates the program $\tilde{C}^{\mathsf{R}}$, all the RO queries made by $\tilde{C}^{\mathsf{R}}$ are answered randomly and consistently with the answers given to the obfuscator $\mathcal{O}^{\mathsf{R}}(C)$ that generated $\tilde{C}^{\mathsf{R}}$. However, even if $\mathcal{O}^{\mathsf{R}}$ is a VBB obfuscator in the RO model, the obfuscator $\mathcal{O}$ may be completely insecure, since the obfuscation $\tilde{C}$ includes the set $\mathcal{R}_C$ in the clear. This may reveal information about the program C.

In our actual transformation, the obfuscation $\tilde{C}$ will include a different set of RO query-answer pairs $\mathcal{R}_X$ that on the one hand, will give no information about the program C, but on the other hand, will result in a obfuscation that is only approximately correct.

The actual plain-model obfuscator $\mathcal{O}$ starts by emulating the random oracle obfuscator $\mathcal{O}^{\mathsf{R}}(C)$ and obtains the RO obfuscation $\tilde{C}^{\mathsf{R}}$ and the set $\mathcal{R}_C$ as before. Next, $\mathcal{O}$ "tests" the RO obfuscation $\tilde{C}^{\mathsf{R}}$ to learn which oracle queries are often made by $\tilde{C}^{\mathsf{R}}$ when executed on a random input. Specifically, $\mathcal{O}$ samples random inputs $x_1, \ldots, x_\ell$ used to test the program $\tilde{C}^{\mathsf{R}}$. The set $\mathcal{R}_X$ is initially empty. For every $i \in [\ell]$, $\mathcal{O}$ emulates the RO obfuscation $\tilde{C}^{\mathsf{R}}(x_i)$ and answers any RO query made by $\tilde{C}^{\mathsf{R}}$ as follows: if

[2]This results in a *randomized* obfuscated program. In the full construction we make the obfuscated program deterministic by including in the description of the obfuscated program a list of random oracle answers that are reused in every evaluation.

the query appears in the set $\mathcal{R}_C$ or in the set $\mathcal{R}_X$ it is answered consistently, otherwise, a random answer is given. In both cases, the query-answer pair is added to the set $\mathcal{R}_X$. Note that the final set $\mathcal{R}_X$ may not contain all the queries in $\mathcal{R}_C$ and it may also contain queries outside $\mathcal{R}_C$.

Finally, the obfuscator $\mathcal{O}$ outputs an obfuscated program $\tilde{C}$ that has hard-coded to it the description of the RO obfuscation $\tilde{C}^{\mathsf{R}}$ and the set $\mathcal{R}_X$. As before, the obfuscation $\tilde{C}$ on an input x emulates the RO obfuscation $\tilde{C}^{\mathsf{R}}(x)$ and answers any RO query made by $\tilde{C}^{\mathsf{R}}$ as follows: if the query appears in the set $\mathcal{R}_X$ it is answered consistently with $\mathcal{R}_X$, otherwise, a random answer is given.

We argue that the new set $\mathcal{R}_X$ gives no information about the program C: Consider the following alternative way to sample the set $\mathcal{R}_X$. Let $\mathcal{R}$ be a random function that is consistent with the query-answer pairs in $\mathcal{R}_C$. Now execute the RO obfuscation $\tilde{C}^{\mathsf{R}}$ on random inputs $x_1, \dots, x_\ell$ and given oracle access to $\mathcal{R}$. The set $\mathcal{R}_X$ simply contains all the query-answer pairs that occur in these executions. Intuitively, since $\mathcal{R}_X$ can be sampled given the RO obfuscation $\tilde{C}^{\mathsf{R}}$ and oracle access to $\mathcal{R}$, it follows from the VBB security of the RO obfuscator $\mathcal{O}^{\mathsf{R}}$ that $\mathcal{R}_X$ reveals no information about the program C.

To argue that $\tilde{C}$ is approximately correct, consider an evaluation of $\tilde{C}$ on a random input x. $\tilde{C}$ emulates the RO obfuscation $\tilde{C}^{\mathsf{R}}(x)$ and answers any RO query made by $\tilde{C}^{\mathsf{R}}$ randomly and consistently with the set $\mathcal{R}_X$. As discussed in the warm-up, if all of the queries made by $\tilde{C}^{\mathsf{R}}$ were answered consistently with the set $\mathcal{R}_C$, perfect correctness would have followed from the correctness of $\mathcal{O}^{\mathsf{R}}$ in the RO model. However, the emulation of $\tilde{C}^{\mathsf{R}}(x)$ may make a query that is in the set $\mathcal{R}_C$ but not in the set $\mathcal{R}_X$. Such a query will be answered randomly in a way that may not be consistent with the answer in $\mathcal{R}_C$ and correctness may be lost. We can therefore bound the probability that $\tilde{C}(x)$ disagrees with $C(x)$ by the probability that $\tilde{C}^{\mathsf{R}}(x)$ makes a query $q \in \mathcal{R}_C \setminus \mathcal{R}_X$. Such a query q must not have been asked by any of the test executions of $\tilde{C}^{\mathsf{R}}$ on the random inputs $x_1, \dots, x_\ell$, otherwise it would have been added to the set $\mathcal{R}_X$. The probability that a query in $\mathcal{R}_C$ is asked by $\tilde{C}(x)$ but is not asked by $\tilde{C}(x_i)$ for any $i \in [\ell]$ is inversely proportional to ℓ. Therefore, by making ℓ large enough, we can make the correctness error sufficiently small (recall that any constant correctness error that is bounded away from 1 is sufficient for the negative result of [BP13] to hold).

Connection to [IR89]. Our proof follows the same outline as the proof of Impagliazzo and Rudich [IR89] separating key-agreement protocols from one-way functions (as well as many subsequent works). In essence, Impagliazzo and Rudich rule out existence of key-agreement protocols secure gainst unbounded adversaries in the RO model. They do so in two steps: first they transform any key-agreement protocol in the RO model into a key-agreement protocol in the plain model. Next they rely on the impossibility for information-theoretically secure key-agreement. We follow the same two steps: first we transform any general (possibly approximate) obfuscator in the RO model to a general approximate obfuscator in the plain model. Next we rely on the impossibility of the latter. Note that in our case the impossibility in the plain model is stronger in the sense that it rules out existence of a primitive that provides only computational security.

2 Impossibility of Obfuscation in the RO Model

In this section we prove an impossibility result for general purpose obfuscation in the RO model. We start by defining approximate obfuscation and state the known impossibility result for obfuscation with approximate correctness.

2.1 Approximate Obfuscation.

We define approximate obfuscation, both in the RO model and in the plain model.

Let $\mathcal{F} = \{F_k\}_{k \in \{0,1\}^*}$ be a family of functions such that F_k has a domain $D_{|k|}$.

Definition 2.1 (Approximate Obfuscation). *For a function $\epsilon : \mathbf{N} \to [0,1]$, a PPT algorithm $\mathcal{O}$ is a secure ϵ-approximate obfuscator for $\mathcal{F}$ if it satisfies the following requirements:*

- *Approximate Functionality: for all $n \in \mathbb{N}, k \in \{0,1\}^n$:*

$$\Pr_{x \leftarrow D_n}[\mathcal{O}(k)(x) \neq F_k(x)] \leq \epsilon(n) \ ,$$

  where the probability is also over the coins of the obfuscator $\mathcal{O}$.

- *Virtual Black-Box: for every poly-size adversary $\mathcal{A}$ there exists a poly-size simulator $\mathcal{S}$ and a negligible function μ such that for every $k \in \{0,1\}^*$:*

$$\left|\Pr[\mathcal{A}(\mathcal{O}(k)) = 1] - \Pr[\mathcal{S}^{F_k}(1^{|k|}) = 1]\right| \leq \mu(|k|) \ ,$$

  where the probabilities are over the coins of the obfuscator $\mathcal{O}$, the adversary $\mathcal{A}$ and the simulator $\mathcal{S}$.

Definition 2.2 (Approximate Obfuscation in the RO model). *For a function $\epsilon : \mathbf{N} \to [0,1]$, a PPT algorithm $\mathcal{O}$ is a secure ϵ-approximate obfuscator for $\mathcal{F}$ in the RO model if it satisfies the following requirements:*

- *Approximate Functionality: for all $n \in \mathbb{N}, k \in \{0,1\}^n$:*

$$\Pr_{x \leftarrow D_n}[\mathcal{O}^{\mathcal{R}}(k)(x) \neq F_k(x)] \leq \epsilon(n) \ ,$$

  where $\mathcal{R} : \{0,1\}^* \to \{0,1\}^*$ is a random function, and the probability is also over $\mathcal{R}$ and the coins of the obfuscator $\mathcal{O}$.

- *Virtual Black-Box: for every poly-size adversary $\mathcal{A}$ there exist a poly-size simulator $\mathcal{S}$ and a negligible function μ such that for every $k \in \{0,1\}^*$:*

$$\left|\Pr[\mathcal{A}^{\mathcal{R}}(\mathcal{O}^{\mathcal{R}}(k)) = 1] - \Pr[\mathcal{S}^{F_k}(1^{|k|}) = 1]\right| \leq \mu(|k|) \ ,$$

  where the probabilities are over $\mathcal{R}$, the coins of the obfuscator $\mathcal{O}$, the adversary $\mathcal{A}$, and the simulator $\mathcal{S}$.

Next we formally state the known impossibility results for approximate obfuscation in the plain model. The following is a direct corollary of [BP13, Theorem 3.1, Theorem 4.1, Lemma 4.1].

Corollary 2.1 ([BP13]). *Assuming trapdoor permutations, there exists a family of functions $\mathcal{F}$ such that an $\left(\frac{1}{2} - \epsilon\right)$-approximate obfuscator for $\mathcal{F}$ does not exist for every noticeable function ϵ.*

Remark 2.1 (More on the impossibility of approximate obfuscation). The work of [BP13] constructs a family of *error-robust* unobfuscatable functions. These are families $\{F_k\}_{k \in \{0,1\}^*}$ such that given oracle access to F_k for a random key k, the key remains completely hidden. However, given the code of any function that agrees with F_k on $\frac{1}{2} + \epsilon$ of the inputs, it is possible to fully recover the key k. This implies the following strong impossibility for approximate obfuscation: For any $\left(\frac{1}{2} - \epsilon\right)$-approximate obfuscator for $\{F_k\}$, with probability at least $\frac{\epsilon}{2}$ over the coins the the obfuscation, the obfuscated function agrees with the original function with probability at least $\frac{1+\epsilon}{2}$. Therefore, with noticeable probability over the coins the the obfuscation, it is always possible to reconstruct the entire key from the obfuscated program.

2.2 The Impossibility

We start by describing a transformation from any (possibly approximate) obfuscation in the RO model to an approximate obfuscation in the plain model. The approximation error of the resulting obfuscation will be slightly larger then that of the original obfuscation.

Theorem 2.1. *If a family of functions $\mathcal{F}$ has a secure ϵ-approximate obfuscator in the RO model then it has a secure $(\epsilon + \delta)$-approximate obfuscator in the plain model for every noticeable function δ.*

Then, we combine the transformation in Theorem 2.1 with the known impossibility result for approximate obfuscation (Corollary 2.1) to derive the following impossibility for obfuscation in the RO model:

Corollary 2.2. *Assuming trapdoor permutations, there exists a family of functions $\mathcal{F}$ such that an $\left(\frac{1}{2} - \epsilon\right)$-approximate obfuscator for $\mathcal{F}$ in the RO model does not exist for every noticeable function ϵ.*

Next we prove Theorem 2.1. See Section 1.2 for a high-level overview of the proof.

Proof. Let $\mathcal{O}$ be a secure ϵ-approximate obfuscator for $\mathcal{F}$ in the RO model, making at most $\ell = \ell(|k|)$ queries to the oracle. We construct a secure $(\epsilon + \delta)$-approximate obfuscator $\mathcal{O}'$ for $\mathcal{F}$ in the plain model.

The obfuscator $\mathcal{O}'$:

1. On input k, emulate $\mathcal{O}(k)$ as follows. Run $\mathcal{O}$ on input k, answer every oracle query made by $\mathcal{O}(k)$ randomly (assume w.l.o.g that $\mathcal{O}$ never makes the same query twice), and obtain an obfuscated oracle circuit C. Set $\mathcal{R}_k$ to be all the queries made by $\mathcal{O}(k)$ and their answers.
2. Set $\mathcal{R}_C$ to be the empty set.
3. For $i = 1$ to $\left\lceil \frac{|C| \cdot \ell}{\delta} \right\rceil$:
   (a) Sample $x_i \leftarrow D_{|k|}$.
   (b) Execute $C(x_i)$. For every oracle query made by $C(x_i)$, if it is in $\mathcal{R}_C \cup \mathcal{R}_k$ then answer consistently, otherwise answer randomly (assume w.l.o.g that C never makes the same query twice). Add all new pairs of queries made by $C(x_i)$ and their answers to $\mathcal{R}_C$.
4. Sample $|C|$ random oracle answers $r_1, \ldots, r_{|C|}$.
5. Output the description of a circuit C' as follows:
   (a) The circuit C' has the description of C, the set $\mathcal{R}_C$ and the answers $\{r_i\}$ hardcoded into it.
   (b) On input x, C' emulates $C(x)$. Let q_i be the i-th oracle query made by $C(x)$. If q_i is in $\mathcal{R}_C$, C' answers consistently, otherwise, C' answers with r_i.
   (c) C' outputs the same as $C(x)$.

Next we show that $\mathcal{O}'$ is a secure $(\epsilon{+}\delta)$-approximate obfuscator. That is, $\mathcal{O}'$ satisfies the approximate functionality and the virtual black-box requirements.

Approximate functionality. Fix a key $k \in \{0,1\}^n$, let $\epsilon = \epsilon(n), \delta = \delta(n)$, and let x be a random input sampled from D_n. By the approximate functionality of $\mathcal{O}$, the circuit C produced by $\mathcal{O}(k)$ satisfies:

$$\Pr_x[C^{\mathcal{R}}(x) \neq F_k(x)] \leq \epsilon \ . \tag{1}$$

Let C' be the obfuscated circuit generated by the plain-model obfuscator $\mathcal{O}'(k)$. Recall that $C'(x)$ emulates the execution of $C(x)$ and the answers the oracle queries made by C. Queries that are in

$\mathcal{R}_C$ are answered consistently with $\mathcal{R}$, and queries outside $\mathcal{R}_C$ are answered from the set of random answers $\{r_i\}$. Since every distinct query made by $C(x)$ is answered randomly and independently, we can consider an identical experiment where C' answers all of C's queries using a random oracle $\mathcal{R}'$ which agrees with $\mathcal{R}$ on all the queries in $\mathcal{R}_C$. Additionally, all the answers of $\mathcal{R}$ and $\mathcal{R}'$ outside the set $\mathcal{R}_k \cup \mathcal{R}_C$ are random independent of C. Let $G(x)$ be the event that the execution of $C^{\mathcal{R}'}(x)$ does not query $\mathcal{R}'$ on any query in the set $\mathcal{R}_k \setminus \mathcal{R}_C$. We have that conditioned on $G(x)$, the output of $C^{\mathcal{R}'}(x)$ and of $C^{\mathcal{R}}(x)$ are identically distributed, and specifically:

$$\Pr_x[(C^{\mathcal{R}'}(x) \neq F_k(x)) \wedge G(x)] = \Pr_x[(C^{\mathcal{R}}(x) \neq F_k(x)) \wedge G(x)] \leq \epsilon \ .$$

Therefore, we can bound the probability of error on x by bounding the probability of the event $\neg G(x)$ as follows:

$$\Pr_x[C^{\mathcal{R}'}(x) \neq F_k(x)] \leq \Pr_x[(C^{\mathcal{R}'}(x) \neq F_k(x)) \wedge G(x)] + \Pr_x[\neg G(x)] \leq \epsilon + \Pr_x[\neg G(x)] \ .$$

Thus, to prove approximate functionality it suffices to prove the following claim, bounding the probability of the event $\neg G(x)$.

Claim 2.1.

$$\Pr_x[\neg G(x)] \leq \delta.$$

Proof of Claim 2.1. We start by giving a high-level overview of the proof. For a random input x, the execution of $C(x)$ makes at most $|C|$ oracle queries. To bound the probability of the event $\neg G(x)$ we bound the probability that the i'th query of $C^{\mathcal{R}'}(x)$ is the first query to fall in the set $\mathcal{R}_k \setminus \mathcal{R}_C$, for every $i \in |C|$. To this end, we argue that the for every query $q \in \mathcal{R}_k$, the probability that the i'th query of $C^{\mathcal{R}'}(x)$ is indeed q, but q was never queried during the "testing phase" of $\mathcal{O}'$ (Step 3) is small. (if q is queried queried during testing phase then $q \in \mathcal{R}_C$.)

Recall that in the testing phase of $\mathcal{O}'$ we execute $C^{\mathcal{R}}$ on many random inputs. Since we are only bounding the probability that the i'th query of $C^{\mathcal{R}'}(x)$ is the *first* query to fall outside the set $\mathcal{R}_k \setminus \mathcal{R}_C$, we can condition on the event that all previous queries do not fall in the set $\mathcal{R}_k \setminus \mathcal{R}_C$. Conditioned on this event, by the definition of the oracles $\mathcal{R}$ and $\mathcal{R}'$, the i-th query of $C^{\mathcal{R}'}$ and of $C^{\mathcal{R}}$ are identically distributed. Therefore, the probability that the i'th query of $C^{\mathcal{R}'}(x)$ is q, but q was never queried in any of the test executions is bounded by the inverse of the number of test executions. Since the number of different queries $q \in \mathcal{R}_k$ is bounded by ℓ we get the required bound on probability that the i'th query of $C^{\mathcal{R}'}(x)$ falls in the set $\mathcal{R}_k \setminus \mathcal{R}_C$, and therefore also on the probability of the event $\neg G(x)$.

We continue with the formal proof of the claim. Let:

$$I = \left\lceil \frac{|C| \cdot \ell}{\delta} \right\rceil \ ,$$

be the number if iterations of the loop in Step 3 of $\mathcal{O}'$. Let q_j be the j-th query $C(x)$ makes. Let $q_{i,j}$ be the j-th query made by the emulation of C on the random input x_i in the i-th iteration of the loop in Step 3. For every $j \in [\ell]$, let $G_j(x)$ the event that $q_j \notin \mathcal{R}_k \setminus \{q_{i,j}\}_{i \in [I]}$. Note that

$$G_j(x) \Rightarrow q_j \notin \mathcal{R}_k \setminus \mathcal{R}_C \ ,$$

and therefore,

$$\forall_j G_j(x) \Rightarrow G(x) \ .$$

Thus we can bound the probability of the event $\neg G(x)$ as follows:

$$\begin{aligned}\Pr_x[\neg G(x)] &\leq \Pr_x[\neg G_1(x) \vee \cdots \vee \neg G_{|C|}(x)] \\ &= \sum_{j \in |C|} \Pr_x\left[G_1(x) \wedge \cdots \wedge G_{j-1}(x) \wedge \neg G_j(x)\right] \ .\end{aligned}$$

It is therefore sufficient to show that for every $j \in [|C|]$,

$$\Pr_x[G_1(x) \wedge \cdots \wedge G_{j-1}(x) \wedge \neg G_j(x)] \leq \frac{\delta}{|C|} \ .$$

To this end, fix $j \in [|C|]$ and fix the oracles $\mathcal{R}$ and $\mathcal{R}'$. Let $\tilde{G}_{j-1}(x)$ denote the event:

$$G_1(x) \wedge \cdots \wedge G_{j-1}(x) \ .$$

Note that:

$$\Pr_x[\tilde{G}_{j-1}(x) \wedge \neg G_j(x)] \leq \Pr_x[\neg G_j(x) | \tilde{G}_{j-1}(x)]$$

and therefore, it suffices to prove that:

$$\Pr_x[\neg G_j(x) | \tilde{G}_{j-1}(x)] \leq \frac{\delta}{|C|} \ .$$

For every query q denote by

$$p_q \triangleq \Pr_x[q_j = q | \tilde{G}_{j-1}(x)] \ .$$

Since for every $i \in [I]$, x and x_i are both uniform in D_n and since the oracles $\mathcal{R}$ and $\mathcal{R}'$ only differ on queries in the set $\mathcal{R}_k \cap \mathcal{R}_C$ we have that conditioned on $\tilde{G}_{j-1}(x)$ the view of the two executions:

$$C^{\mathcal{R}'}(x) \quad \text{and} \quad C^{\mathcal{R}}(x_i)$$

up until the j-th query, are identically distributed. Therefore, for every $i \in [I]$:

$$p_q = \Pr_x[q_j = q | \tilde{G}_{j-1}(x)] = \Pr_x[q_{i,j} = q | \tilde{G}_{j-1}(x)] \ .$$

Thus, as desired,

$$\begin{aligned}&\Pr_x[\neg G_j(x) | \tilde{G}_{j-1}(x)] \leq \\ &\sum_{q \in \mathcal{R}_k} \Pr_x[(q_j = q) \wedge (\forall i, q_{i,j} \neq q) \,|\tilde{G}_{j-1}(x)] \leq \\ &\sum_{q \in \mathcal{R}_k} p_q (1 - p_q)^{\frac{|C| \cdot \ell}{\delta}} \leq \qquad (2) \\ &\sum_{q \in \mathcal{R}_k} \frac{\delta}{|C| \cdot \ell} \leq \frac{\delta}{|C|} \ ,\end{aligned}$$

where (2) follows from the fact that the expression $p_q(1 - p_q)^e$ is maximized by $p_q = \frac{1}{e+1}$. This completes the proof of Claim 2.1. □

Virtual Black-Box. Fix a key $k \in \{0, 1\}^n$ and let $\mathcal{A}'$ be an adversary that tries to learn some information from the obfuscation $\mathcal{O}'(k)$. We show how to use the code of $\mathcal{A}'$ to construct an adversary $\mathcal{A}$ that learns the same information from the obfuscation $\mathcal{O}(k)$ where both $\mathcal{A}$ and $\mathcal{O}$ have access to the same random oracle. That is, we will show that:

$$\Pr[\mathcal{A}^{\mathcal{R}}(\mathcal{O}^{\mathcal{R}}(k)) = 1] = \Pr[\mathcal{A}'(\mathcal{O}'(k)) = 1] \ , \tag{3}$$

where the probabilities are over $\mathcal{R}$, the coins of the obfuscators $\mathcal{O}$ and $\mathcal{O}'$, and the coins of the adversaries $\mathcal{A}$ and $\mathcal{A}'$. By the security of $\mathcal{O}$, there exist a simulator $\mathcal{S}$ and a negligible function μ such that:

$$\left|\Pr[\mathcal{A}^{\mathcal{R}}(\mathcal{O}^{\mathcal{R}}(k)) = 1] - \Pr[\mathcal{S}^{F_k}(1^n) = 1]\right| \leq \mu(n) \ . \tag{4}$$

It follows from Equations (3) and (4) that $\mathcal{S}$ is a good simulator for $\mathcal{A}'$ as well. It is left to show how to construct an adversary $\mathcal{A}$ that satisfies Equation (3). Loosely speaking, given an obfuscation $\mathcal{O}(k)$, $\mathcal{A}$ will use the same strategy of the obfuscator $\mathcal{O}'$ to transform the obfuscation $\mathcal{O}(k)$ into an obfuscation $\mathcal{O}'(k)$ and then execute $\mathcal{A}'$ on $\mathcal{O}'(k)$. $\mathcal{A}$ will use its random oracle to answer queries made by $\mathcal{O}(k)$. Formally, $\mathcal{A}$ is defined as follows:

1. Given an obfuscated input circuit C and given access to oracle $\mathcal{R}$, repeat the following for $i = 1$ to $\left\lceil \frac{|C|\cdot\ell}{\delta} \right\rceil$:
   (a) Sample $x_i \leftarrow D_n$.
   (b) Execute $C(x_i)$ and forward its oracle queries to $\mathcal{R}$.
2. Sample $|C|$ random oracle answers $r_1, \ldots, r_{|C|}$.
3. Set $\mathcal{R}_C$ to be the set of queries made by C in Step 1 and their answers. Construct a circuit C' from C, $\mathcal{R}_C$ and $\{r_i\}$ as in Step 5 of the obfuscator $\mathcal{O}'$.
4. Output the same as $\mathcal{A}'(C')$.

By construction, the circuit C' used by $\mathcal{A}$ in Step 4 is distributed identically to the output of $\mathcal{O}'(k)$ and therefore Equation (3) holds. $\square$

References

[BCG$^+$11] Nir Bitansky, Ran Canetti, Shafi Goldwasser, Shai Halevi, Yael Tauman Kalai, and Guy N. Rothblum. Program obfuscation with leaky hardware. In *ASIACRYPT*, pages 722–739, 2011.

[BCKP14] Nir Bitansky, Ran Canetti, Yael Tauman Kalai, and Omer Paneth. On virtual grey box obfuscation for general circuits. In *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part II*, pages 108–125, 2014.

[BGI$^+$01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In *CRYPTO*, pages 1–18, 2001.

[BGK$^+$14] Boaz Barak, Sanjam Garg, Yael Tauman Kalai, Omer Paneth, and Amit Sahai. Protecting obfuscation against algebraic attacks. In *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, pages 221–238, 2014.

[BP13] Nir Bitansky and Omer Paneth. On the impossibility of approximate obfuscation and applications to resettable cryptography. In *STOC*, pages 241–250, 2013.

[BR14] Zvika Brakerski and Guy N. Rothblum. Virtual black-box obfuscation for all circuits via generic graded encoding. In *Theory of Cryptography - 11th Theory of Cryptography Conference, TCC 2014, San Diego, CA, USA, February 24-26, 2014. Proceedings*, pages 1–25, 2014.

[BS84] László Babai and Endre Szemerédi. On the complexity of matrix group problems I. In *25th Annual Symposium on Foundations of Computer Science, West Palm Beach, Florida, USA, 24-26 October 1984*, pages 229–240, 1984.

[Can97] Ran Canetti. Towards realizing random oracles: Hash functions that hide all partial information. In *CRYPTO*, pages 455–469, 1997.

[CV13] Ran Canetti and Vinod Vaikuntanathan. Obfuscating branching programs using black-box pseudo-free groups. *IACR Cryptology ePrint Archive*, 2013:500, 2013.

[GGH13a] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In *EUROCRYPT*, pages 1–17, 2013.

[GGH$^+$13b] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *FOCS*, 2013.

[GIS$^+$10] Vipul Goyal, Yuval Ishai, Amit Sahai, Ramarathnam Venkatesan, and Akshay Wadia. Founding cryptography on tamper-proof hardware tokens. In *TCC*, pages 308–326, 2010.

[GR14] Shafi Goldwasser and Guy N. Rothblum. On best-possible obfuscation. *J. Cryptology*, 27(3):480–505, 2014.

[IR89] Russell Impagliazzo and Steven Rudich. Limits on the provable consequences of one-way permutations. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing, May 14-17, 1989, Seattle, Washigton, USA*, pages 44–61, 1989.

[LPS04] Ben Lynn, Manoj Prabhakaran, and Amit Sahai. Positive results and techniques for obfuscation. In *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, pages 20–39, 2004.

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In *Advances in Cryptology - EUROCRYPT '97, International Conference on the Theory and Application of Cryptographic Techniques, Konstanz, Germany, May 11-15, 1997, Proceeding*, pages 256–266, 1997.

[Wee05] Hoeteck Wee. On obfuscating point functions. *IACR Cryptology ePrint Archive*, 2005:1, 2005.