Obfuscation for Evasive Functions

Boaz Barak[1], Nir Bitansky[2] ⋆, Ran Canetti[2,3] ⋆⋆,
Yael Tauman Kalai[1], Omer Paneth[3] ⋆ ⋆ ⋆, and Amit Sahai[4] †

[1] Microsoft Research
[2] Tel Aviv University
[3] Boston University
[4] UCLA

Abstract. An *evasive circuit family* is a collection of circuits $\mathcal{C}$ such that for every input x, a random circuit from $\mathcal{C}$ outputs 0 on x with overwhelming probability. We provide a combination of definitional, constructive, and impossibility results regarding obfuscation for evasive functions:

1. The (average case variants of the) notions of *virtual black box* obfuscation (Barak et al, CRYPTO '01) and *virtual gray box* obfuscation (Bitansky and Canetti, CRYPTO '10) coincide for evasive function families. We also define the notion of *input-hiding* obfuscation for evasive function families, stipulating that for a random $C \in \mathcal{C}$ it is hard to find, given $\mathcal{O}(C)$, a value outside the preimage of 0. Interestingly, this natural definition, also motivated by applications, is likely not implied by the seemingly stronger notion of average-case virtual black-box obfuscation.
2. If there exist average-case virtual gray box obfuscators for all evasive function families, then there exist (quantitatively weaker) average-case virtual gray obfuscators for *all* function families.
3. There does not exist a *worst-case* virtual black box obfuscator even for evasive circuits, nor is there an average-case virtual gray box obfuscator for evasive *Turing machine* families.
4. Let $\mathcal{C}$ be an evasive circuit family consisting of functions that test if a low-degree polynomial (represented by an efficient arithmetic circuit) evaluates to zero modulo some large prime p. Then under a natural analog of the discrete logarithm assumption in a group supporting multilinear maps, there exists an input-hiding obfuscator for $\mathcal{C}$. Under a new *perfectly-hiding multilinear encoding* assumption, there is an average-case virtual black box obfuscator for the family $\mathcal{C}$.

⋆ Supported by an IBM Ph.D. Fellowship and the Check Point Institute for Information Security.

⋆⋆ Supported by the Check Point Institute for Information Security, an NSF EAGER grant, and an NSF Algorithmic foundations grant 1218461.

⋆ ⋆ ⋆ Work done while the author was an intern at Microsoft Research New England. Supported by the Simons award for graduate students in theoretical computer science and an NSF Algorithmic foundations grant 1218461.

† Work done in part while visiting Microsoft Research, New England. Research supported in part from a DARPA/ONR PROCEED award, NSF grants 1228984, 1136174, 1118096, and 1065276, a Xerox Faculty Research Award, a Google Faculty Research Award, an equipment grant from Intel, and an Okawa Foundation Research Grant. This material is based upon work supported by the Defense Advanced Research Projects Agency through the U.S. Office of Naval Research under Contract N00014-11-1-0389. The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense, the National Science Foundation, or the U.S. Government.

1 Introduction

The study of *Secure Software Obfuscation* — or, methods to transform a program (say described as a Boolean circuit) into a form that is executable but otherwise completely unintelligible — is a central research direction in cryptography. In this work, we study obfuscation of evasive functions— an *evasive function family* is a collection $\mathcal{C}_n$ of Boolean circuits mapping some domain $\mathcal{D}$ to $\{0, 1\}$ such that for every $x \in \mathcal{D}$ the probability over $C \leftarrow \mathcal{C}_n$ that $C(x) = 1$ is negligible. Focusing on evasive functions leads us to new notions of obfuscation, as well as new insights into general-purpose obfuscation.

Why Study Obfuscation of Evasive Functions? To motivate the study of the obfuscation of evasive functions, let us consider the following scenario taken from [13]: Suppose that a large software publisher discovers a new vulnerability in their software that causes the software to behave in undesirable ways on certain (rare) inputs. The publisher then designs a software patch P that tests the input x to see if it falls into the set S of bad inputs, and if so outputs 1 to indicate that the input x should be ignored. If $x \notin S$, the patch outputs 0 to indicate that the software can behave normally. If the publisher releases the patch P "in the clear", an adversary could potentially study the code of P and learn bad inputs $x \in S$ that give rise to the original vulnerability. Since it can take months before a majority of computers install a new patch, this would give the attacker plenty of time to exploit this vulnerability on unpatched systems *even when the set S of bad inputs was evasive to begin with.* If instead, the publisher could obfuscate the patch P before releasing it, then intuitively an attacker would gain no advantage in finding an input $x \in S$ from studying the obfuscated patch. Indeed, assuming that without seeing P the attacker has negligible chance of finding an input $x \in S$, it makes sense to model P as an evasive function.

Aside from the motivating scenario above, evasive functions are natural to study in the context of software obfuscation because they are a natural generalization of the special cases for which obfuscation was shown to exist in the literature such as point functions [8], hyperplanes [10], and conjunctions [6].[5] Indeed, as we shall see, the study of obfuscation of evasive functions turns out to be quite interesting from a theoretical perspective, and sheds light on general obfuscation as well.

What notions of obfuscation makes sense for evasive functions? As the software patch problem illustrates, a very natural property that we would like is *input hiding*: Given an obfuscation of a random circuit $C \leftarrow \mathcal{C}_n$, it should be hard for an adversary to find an input x such that $C(x) = 1$. It also makes sense to consider (average case versions of) strong notions of obfuscation, such as "virtual black box" (VBB) obfuscation introduced by Barak, Goldreich, Impagliazzo, Rudich, Sahai, Vadhan and Yang [3], which, roughly speaking, states that any predicate of the original circuit C computed from its obfuscation could be computed with almost the same probability by an efficient simulator having only oracle (i.e., black box) access to the function (see Section 2 for a formal definition). One can also consider the notion of "virtual gray box" (VGB)

[5] Conjunctions are not necessarily evasive, however the interesting case for obfuscation is large conjunctions which are evasive.

introduced by Bitansky and Canetti [4], who relaxed the VBB definition to allow the simulator to run in unbounded time (though still with only a polynomial number of queries to its oracle). Another definition proposed in [3], with a recent construction given by [15], is of "indistinguishability obfuscation" (IO). The actual meaning of IO is rather subtle, and we discuss it in Section 1.2.

1.1 Our results

We provide a combination of definitional, constructive, and impossibility results regarding obfuscation for evasive functions. First, we formalize the notion of input-hiding obfuscation for evasive functions (as sketched above). We give evidence that this notion of input-hiding obfuscation is actually *incomparable* to the standard definition of VBB obfuscation. While it is not surprising that input-hiding obfuscation does not imply VBB obfuscation, it is perhaps more surprising that VBB obfuscation does not imply input-hiding obfuscation (under certain computational assumptions). Intuitively, this is because VBB obfuscation requires only that *predicates* of the circuit being obfuscated are simulatable, whereas input hiding requires that no complete input string x that causes $C(x) = 1$ can be found.

Second, we formalize a notion of *perfect circuit-hiding* obfuscation, which asks that for *every* predicate of the circuit $C \leftarrow \mathcal{C}_n$, the probability that the adversary can guess the predicate (or its complement) given an obfuscation of C is negligibly close to the expected value of the predicate over $C \leftarrow \mathcal{C}_n$. We then show that for any evasive circuit family $\mathcal{C}$, this simple notion of obfuscation is equivalent to both average-case VBB obfuscation and average-case VGB obfuscation. Thus, in particular, we have:

Theorem 1 (Informal) *For every evasive function collection $\mathcal{C}$ and obfuscator $\mathcal{O}$, it holds that $\mathcal{O}$ is an average-case VBB obfuscator for $\mathcal{C}$ if and only if it is an average-case VGB obfuscator for $\mathcal{C}$.*

We also show that evasive functions are at the heart of the VGB definition in the sense that if it is possible to achieve VGB obfuscators for all evasive circuit families then it is possible to achieve a slightly weaker variant of VGB obfuscation for *all* circuits:

Theorem 2 (Informal) *If there exists an average-case VGB obfuscator for every evasive circuit family then there exists a "weak" average-case VGB obfuscator for every (not necessarily evasive) circuit family.*

The notion of "weak" VGB obfuscation allows the simulator to make a slightly super-polynomial number of queries to its oracle. It also allows the obfuscating algorithm to be inefficient. The latter relaxation is not needed if we assume the existence of indistinguishability obfuscators for all circuits, as conjectured in [15].

We then proceed to give new *constructions* of obfuscators for specific natural families of evasive functions. We focus on functions that test if a bounded (polynomial) degree multivariate polynomial, given by an arithmetic circuit, evaluates to zero modulo some large prime p. We provide two constructions that build upon the approximate multilinear map framework developed by Garg, Gentry, and Halevi [14] and continued by Coron, Lepoint, and Tibouchi [12]. We first construct an input-hiding obfuscator whose security is based on a variant of the discrete logarithm assumption in the multilinear setting. We then construct a perfect circuit-hiding obfuscator based on a new assumption, called *perfectly-hiding multilinear encoding*. Very roughly, we assume that given encodings of k and $r \cdot k$, for a random r and k, the value of any predicate of k cannot be efficiently learned.

Theorem 3 (Informal) *Let $\mathcal{C}$ be the evasive function family that tests if a bounded (polynomial) degree multivariate polynomial, given by an arithmetic circuit, evaluates to zero modulo some large prime p. Then:* ***(i)*** *Assuming the existence of a group supporting a multilinear map in which the discrete logarithm problem is hard, there exists an input-hiding obfuscator for $\mathcal{C}$, and* ***(ii)*** *Under the perfectly-hiding multilinear encoding assumption, there exists an average-case VBB obfuscator for all log-depth circuits in $\mathcal{C}$.*

These constructions can be combined to obtain a single obfuscator for testing if an input is in the zero-set of a bounded-degree polynomial, that simultaneously achieves input-hiding and average-case VBB obfuscation. We give an informal overview of our construction in Section 3.

Finally, we complement our constructive results by giving two impossibility results regarding the obfuscation of evasive functions. First, we show impossibility with respect to evasive Turing Machines:

Theorem 4 (Informal) *There exists a class of evasive Turing Machines $\mathcal{M}$ such that no input-hiding obfuscator exists with respect to $\mathcal{M}$ and no average-case VGB obfuscator exists with respect to $\mathcal{M}$.*

We also show that there exist classes of evasive *circuits* for which VBB obfuscation is not possible. However, here we only rule out *worst case* obfuscation:

Theorem 5 (Informal) *There exists a class of evasive circuits $\mathcal{C}$ such that no worst-case VBB obfuscator exists with respect to $\mathcal{C}$.*

1.2 Alternative approaches to obfuscation

We briefly mention two other approaches to general program obfuscation. One is to use the notion of *indistinguishability obfuscation* (IO) [3]. Indeed, in a recent breakthrough result, Garg, Gentry, Halevi, Raykova, Sahai and Waters [15] propose a candidate general obfuscation mechanism and conjecture that it is IO for all circuits. Improved variants of this construction have been proposed in [7, 2]. Roughly speaking, IO implies that an obfuscation of a circuit C hides "as much as possible" about the circuit C in the sense that if $\mathcal{O}$ is an IO obfuscator, and there exists some other obfuscator $\mathcal{O}'$ (with not too long output) that (for instance) achieves input hiding or VBB obfuscation for C, then $\mathcal{O}(C)$ will achieve the same security as well [16]. However, while IO obfuscators have found uses for many cryptographic applications [15, 18, 17], its hard to quantify what security is obtained by directly obfuscating a function with an IO obfuscator since for most classes of functions we do not know what "as much as possible" is. In particular, IO obfuscators are not known to imply properties such as input hiding or VBB/VGB for general circuits. For insance, the "totally unobfuscatable functions" of [3] (which are functions that are hard to learn but whose canonical representation can be recovered from any circuit) can be trivially IO-obfuscated, but this clearly does not give any meaningful security guarantees. Furthermore, we do not know whether IO obfuscators give guarantees such as input hiding or VBB on any families of evasive functions beyond those families that we already know how to VBB-obfuscate. Thus our work here can be seen as complementary to the question of constructing indistinguishability obfuscators. Furthermore, the hardness assumptions made in this work are much simpler than those implied by the IO security of any of the above candidates.

Another approach is to consider obfuscation mechanisms in idealized models where basic group operations are modeled as calls to abstract oracles. Works along this line, using different levels of abstraction, include [15, 11, 7, 2]. It should be stressed however that, as far as we know, proofs of security in any of these idealized models bear no immediate relevance to the security of obfuscation schemes in the plain model.

1.3 Organization of the paper

In Section 2 we formally define evasive function families and the various notions of obfuscation that apply to them, and show equivalence between several of these notions. Section 3 contains our constructions for obfuscating zero-testing functions for low degree polynomials. In Section 4 we show that obtaining virtual gray box obfuscation for evasive functions implies a weaker variant of VGB for *all* functions. Section 5 contains our impossibility results for worst-case VBB obfuscation of evasive circuits and average-case VGB obfuscation of evasive Turing machines.

2 Evasive Circuit Obfuscation

Let $\mathcal{C} = \{\mathcal{C}_n\}_{n\in\mathbb{N}}$ be a collection of circuits such that every $C \in \mathcal{C}_n$ maps n input bits to a single output bit, and has size $\mathrm{poly}(n)$. We say that $\mathcal{C}$ is *evasive* if on every input, a random circuit in the collection outputs 0 with overwhelming probability:[6]

Definition 2.1 (Evasive Circuit Collection). *A collection of circuits $\mathcal{C}$ is evasive if there exist a negligible function μ such that for every $n \in \mathbb{N}$ and every $x \in \{0,1\}^n$:*

$$\Pr_{C\leftarrow\mathcal{C}_n}[C(x) = 1] \leq \mu(n) \enspace .$$

An equivalent definition that will be more useful to us states that given oracle access to a random circuit in the collection it is hard to find an input that maps to 1.

Definition 2.2 (Evasive Circuit Collection - Alternative Definition). *A collection of circuits $\mathcal{C}$ is evasive if for every (possibly inefficient) $\mathcal{A}$ and for every polynomial q there*

[6] To avoid confusion, we note that the notion here is unrelated (and quite different) from the notion of evasive relations in [9].

exist a negligible function μ such that and for every $n \in \mathbb{N}$:

$$\Pr_{C \leftarrow \mathcal{C}_n}[C(\mathcal{A}^{C[q(n)]}(1^n)) = 1] \leq \mu(n) \ .$$

Where $C[q(n)]$ denotes an oracle to the circuit C which allows at most $q(n)$ queries.

2.1 Definitions of Evasive Circuit Obfuscation

We start by recalling the syntax and functionality requirement for circuit obfuscation as defined in [3].

Definition 2.3 (Functionality of Circuit Obfuscation). *An obfuscator $\mathcal{O}$ for a circuit collection $\mathcal{C}$ is a PPT algorithm such that for all $C \in \mathcal{C}$, $\mathcal{O}(C)$ outputs a description of a circuit that computes the same function as C.*

We suggest two new security notions for obfuscation of evasive collections: perfect circuit-hiding and input-hiding. Both notions are average-case notions, that is, they only guarantee security when the obfuscated circuit is chosen at random from the collection.

The notion of input hiding asserts that given an obfuscation of a random circuit in the collection, it remains hard to find input that evaluates to 1.

Definition 2.4 (Input-Hiding Obfuscation). *An obfuscator $\mathcal{O}$ for a collection of circuits $\mathcal{C}$ is input-hiding if for every PPT adversary $\mathcal{A}$ there exist a negligible function μ such that for every $n \in \mathbb{N}$ and for every auxiliary input $z \in \{0,1\}^{\mathrm{poly}(n)}$ to $\mathcal{A}$:*

$$\Pr_{C \leftarrow \mathcal{C}_n}[C(\mathcal{A}(z, \mathcal{O}(C))) = 1] \leq \mu(n) \ ,$$

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

Our second security notion of perfect circuit-hiding asserts that an obfuscation of a random circuit in the collection does not reveal any partial information about original circuit. We show that for evasive collections, this notion is equivalent to existing definitions of average-case obfuscation such as average-case virtual black-box (VBB) [3], average-case virtual gray-box (VGB) [4], and average-case oracle indistinguishability [8].

Definition 2.5 (Perfect Circuit-Hiding Obfuscation). *Let $\mathcal{C}$ be a collection of circuits. An obfuscator $\mathcal{O}$ for a circuit collection $\mathcal{C}$ is perfect circuit-hiding if for every PPT adversary $\mathcal{A}$ there exist a negligible function μ such that for every balanced predicate $\mathcal{P}$, every $n \in \mathbb{N}$ and every auxiliary input $z \in \{0,1\}^{\mathrm{poly}(n)}$ to $\mathcal{A}$:*

$$\Pr_{C \leftarrow \mathcal{C}_n}[\mathcal{A}(z, \mathcal{O}(C)) = \mathcal{P}(C)] \leq \frac{1}{2} + \mu(n) \ ,$$

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

Remark 2.1 (On Definition 2.5). The restriction to the case of balanced predicates is made for simplicity of exposition only. We note that the proof of Theorem 2.1 implies that Definition 2.5 is equivalent to a more general definition that considers all predicates.

Remark 2.2 (On extending the definitions for non-evasive functions). The definitions of input-hiding and perfect circuit-hiding obfuscation are tailored for evasive collections and clearly cannot be achieved for all collections of circuits. For the case of input-hiding, this follows directly from Definition 2.2 of evasive collections. For the case of perfect circuit-hiding, consider the non-evasive collection $\mathcal{C}$ such that for every $C \in \mathcal{C}$, $C(0^n)$ outputs the first bit of C. Clearly, no obfuscator can preserve functionality while hiding the first bit of the circuit.

2.2 On the Relations Between the Definitions

An input-hiding obfuscation is not necessarily perfect circuit-hiding since an input-hiding obfuscation might always include, for example, the first bit of the circuit in the output. In the other direction we do not believe that every perfect circuit-hiding obfuscation is also input hiding. Intuitively, the reason is that the perfect circuit-hiding obfuscation only prevents the adversary from learning a predicate of the circuit. Note that there may be many inputs on which the circuit evaluates to 1, and therefore, even if the obfuscation allows the adversary to learn some input that evaluates to 1, it is not clear how to use such an input to learn a predicate of the circuit.

In the full version of this paper [1] we give an example of an obfuscation for some evasive collection that is perfect circuit-hiding but not input-hiding. The example is based on a worst case obfuscation assumption for hyperplanes [10]. Nonetheless, we prove that for evasive collections where every circuit only evaluates to 1 on a *polynomial* number of inputs, every perfect circuit-hiding obfuscation is also input-hiding.

2.3 Perfect Circuit-Hiding Obfuscation is Equivalent to Existing Notions

We start by recalling the average-case versions of existing security definitions of obfuscation.

Definition 2.6 (Average-Case Virtual Black-Box (VBB) from [3]). *An obfuscator $\mathcal{O}$ for a collection of circuits $\mathcal{C}$ is average-case VBB if for every PPT adversary $\mathcal{A}$ there exists a PPT simulator* Sim *and a negligible function μ such that for every predicate $\mathcal{P}$, every $n \in \mathbb{N}$ and every auxiliary input $z \in \{0,1\}^{\mathrm{poly}(n)}$ to $\mathcal{A}$:*

$$\left| \Pr_{C \leftarrow \mathcal{C}_n}[\mathcal{A}(z, \mathcal{O}(C)) = \mathcal{P}(C)] - \Pr_{C \leftarrow \mathcal{C}_n}[\mathsf{Sim}^C(z, 1^n) = \mathcal{P}(C)] \right| \leq \mu(n) \ ,$$

where the probability is also over the randomness of $\mathcal{O}$ and Sim.

The notion of VGB relaxes VBB by considering a computationally unbounded simulator, however, the number of queries the simulator makes to its oracle is bounded.

Definition 2.7 (Average-Case VGB Obfuscation from [4]). *An obfuscator $\mathcal{O}$ for a collection of circuits $\mathcal{C}$ is average-case VGB if for every PPT adversary $\mathcal{A}$ there exists a negligible function μ, a polynomial q and a (possibly inefficient) simulator* Sim *such* *that for every predicate $\mathcal{P}$, every $n \in \mathbb{N}$ and every auxiliary input $z \in \{0,1\}^{\mathrm{poly}(n)}$ to $\mathcal{A}$:*

$$\left| \Pr_{C \leftarrow \mathcal{C}_n}[\mathcal{A}(z, \mathcal{O}(C)) = \mathcal{P}(C)] - \Pr_{C \leftarrow \mathcal{C}_n}[\mathsf{Sim}^{C[q(n)]}(z, 1^n) = \mathcal{P}(C)] \right| \leq \mu(n) \ ,$$

where $C[q(n)]$ denotes an oracle to the circuit C which allows at most $q(n)$ queries and where the probability is also over the randomness of $\mathcal{O}$ and Sim.

The notion of oracle indistinguishability was originally formulated in the context of point functions, and here we give a variation of it for general collections. Similarly to our new notions, this definition is meaningful for evasive collections, but not for arbitrary collections.

Definition 2.8 (Average-Case Oracle-Indistinguishability Obfuscation from [8]). *An obfuscator $\mathcal{O}$ for a collection of circuits $\mathcal{C}$ is average-case oracle indistinguishable if for every PPT adversary $\mathcal{A}$ that outputs one bit, the following ensembles are computationally indistinguishable:*

– $\{(C, \mathcal{A}(z, \mathcal{O}(C))) \mid C \leftarrow \mathcal{C}_n\}_{n \in \mathbb{N}, z \in \{0,1\}^{\mathrm{poly}(n)}}$,
– $\{(C, \mathcal{A}(z, \mathcal{O}(C'))) \mid C, C' \leftarrow \mathcal{C}_n\}_{n \in \mathbb{N}, z \in \{0,1\}^{\mathrm{poly}(n)}}$.

The following theorem showing that, for evasive collections, the above notions are all equivalent to perfect circuit-hiding is proven in the full version of this paper [1]. We note that, for general circuits, average-case VBB and average-case VGB may not be equivalent (follows from [4, Proposition 4.1]). The equivalence of average-case VBB and average-case oracle-indistinguishability was proven for point functions by Canetti [8]. We generalize the claim for all evasive functions.

Theorem 2.1. *Let $\mathcal{O}$ be an obfuscator for an evasive collection $\mathcal{C}$. The following statements are equivalent:*

1. *$\mathcal{O}$ is perfect circuit-hiding (Definition 2.5).*
2. *$\mathcal{O}$ is average-case VBB (Definition 2.6).*
3. *$\mathcal{O}$ is average-case VGB (Definition 2.7).*
4. *$\mathcal{O}$ is average-case oracle-indistinguishability (Definition 2.8).*

Remark 2.3 (On evasive obfuscation with a universal simulator). It follows from the proof of Theorem 2.1 that every obfuscator $\mathcal{O}$ for an evasive collection $\mathcal{C}$ that is average-case VBB-secure (or average-case VGB-secure) can be simulated as follows: given an adversary $\mathcal{A}$, the simulator Sim simply executes $\mathcal{A}$ on a random obfuscation $\mathcal{O}(C')$ of a circuit C' sampled uniformly from the collection $\mathcal{C}$. The simplicity of the above simulator can be demonstrated as follows:

– The simulator is *universal*, that is, the same algorithm Sim can simulate every obfuscator $\mathcal{O}$ and family $\mathcal{C}$ given only black box access to $\mathcal{O}$ and the ability to sample uniformly from $\mathcal{C}$.
– The simulator only makes *black-box* use of the adversary $\mathcal{A}$. This is in contrast to the case of worst-case VBB-security where non-black simulators are required ([8, 20]).
– The simulator does not make any calls to its oracle. This is in contrast to the case of non-evasive function where, for example, the possibility of obfuscating learnable functions can only be demonstrated by a simulator that queries it oracle.

3 Obfuscating Root Sets of Low Degree Polynomials

In this section, we present constructions of input-hiding obfuscation and of perfect circuit-hiding obfuscation for a subclass of evasive collections. We will be able to obfuscate collections that can be expressed as the zero-set of some low-degree polynomial. More concretely, we say that a collection $\mathcal{C}$ is of *low arithmetic degree* if for every n, there is a $\theta(n)$-bit prime p, and a polynomial size low degree arithmetic circuit $U(k, x)$ over $\mathbb{Z}_p$ where $k \in \mathbb{Z}_p^\ell, x \in \mathbb{Z}_p^m$ such that $\mathcal{C}_n = \{C_k\}_{k \in \mathbb{Z}_p^\ell}$ and $C_k(x) = 1$ iff $U(k, x) = 0$.

Note that the Schwartz-Zippel Lemma, together with the fact that $U(k, x)$ is of low degree implies that for every $x \in \mathbb{Z}_p^m$ either

$$\Pr_{k \leftarrow \mathbb{Z}_p^\ell}[C_k(x) = 0] = \mathrm{negl}(n) \quad \text{or,} \quad \Pr_{k \leftarrow \mathbb{Z}_p^\ell}[C_k(x) = 0] = 1 \ .$$

Thus, there exists a single function h such that the collection $\{C_k - h\}_{k \in \mathbb{Z}_p^\ell}$ is evasive, where $h(x) = 1$ iff:

$$\Pr_{k \leftarrow \mathbb{Z}_p^\ell}[C_k(x) = 0] = 1 \ .$$

In other words, a collection of low arithmetic degree is either evasive or it is a "translation" of an evasive collection by a fixed, efficiently computable (in RP) function that is independent of the key. Therefore, we can restrict ourselves WLOG to *evasive* collection of low arithmetic degree.

Both constructions will be based on *graded encoding* as introduced by Garg, Gentry, and Halevi [14]. The high-level idea behind both constructions is as follows. The obfuscation of a circuit C_k will contain some encoding of the elements of k. Using this encoding, an evaluator can homomorphically evaluate the low-degree polynomial $U(k, x)$. Then, the evaluator tests whether the output is an encoding of 0 and learns the value of $C_k(x)$.

The two constructions will encode the key k in two different ways, and their security will be based on two different hardness assumptions. The security of the input-hiding construction will be based on a discrete-logarithm-style assumption on the encoding. The obfuscation will support evasive collections of low arithmetic degree defined by a polynomial size circuit $U(k, x)$ of total degree $\mathrm{poly}(n)$. This class of circuits is equivalent to polynomial size arithmetic circuits of depth $\mathcal{O}(\log^2(n))$ and total degree $\mathrm{poly}(n)$ [19]. The security of the perfect circuit-hiding construction will be based on a new assumption called *the perfectly-hiding multilinear encoding assumption*. The obfuscation will support evasive collections defined by a polynomial size circuit $U(k, x)$ of depth $\mathcal{O}(\log(n))$. We also discuss a stronger variant of this assumption, which like in the input-hiding construction, supports arbitrary arithmetic circuits with total polynomial degree.

3.1 Graded Encoding

We start by defining a variant of the symmetric graded encoding scheme from ([14]), used in our construction, and by specifying hardness assumptions used.

Definition 3.1. *A d-graded encoding system consists of a ring R and a collection of disjoint sets of encodings $\left\{S_i^{(\alpha)} \mid \alpha \in R, 0 \leq i \leq d\right\}$. The scheme supports the following efficient procedures:*

- *Instance Generation: given security parameter n and the parameter d,* $\mathsf{Gen}(1^n, 1^d)$ *outputs public parameters* pp.
- *Encoding: given the public parameters* pp *and $\alpha \in R$,* $\mathsf{Enc}(\mathsf{pp}, \alpha)$ *outputs $u \in S_1^{(\alpha)}$.*
- *Addition and Negation: given the public parameters* pp *and two encodings $u_1 \in S_i^{(\alpha_1)}, u_2 \in S_i^{(\alpha_2)}$,* $\mathsf{Add}(\mathsf{pp}, u_1, u_2)$ *outputs an encoding in $S_i^{(\alpha_1+\alpha_2)}$, and* $\mathsf{Neg}(\mathsf{pp}, u_1)$ *outputs an encoding in $S_i^{(-\alpha_1)}$.*
- *Multiplication: given the public parameters* pp *and two encodings $u_1 \in S_{i_1}^{(\alpha_1)}$, $u_2 \in S_{i_2}^{(\alpha_2)}$ such that $i_1 + i_2 \leq d$,* $\mathsf{Mul}(\mathsf{pp}, u_1, u_2)$ *outputs an encoding in $S_{i_1+i_2}^{(\alpha_1 \cdot \alpha_2)}$.*
- *Zero Test: given the public parameters* pp *and an encodings u,* $\mathsf{Zero}(\mathsf{pp}, u)$ *outputs 1 if $u \in S_d^{(0)}$ and 0 otherwise.*

The main difference between this formulation and the formulation in [14] is that we assume that there is an efficient procedure for generating level 1 encoding of every ring element. In [14], it is only possible to obliviously generate an encoding of a random element in R, without knowing the underlying encoded element. While we currently do not know how how to use the construction of [14] to instantiate our scheme, we can get a candidate construction with public encoding based on the construction of [12], by publishing encodings of all powers of 2 as part of the public parameters. The known candidate constructions involve noisy encodings. Since in our construction we use at most $O(d)$ operations, we may set the noise parameter to be small enough so that it can be ignored. Therefore, from hereon, we omit the management of noise from the interfaces of the graded encoding.

Our first hardness assumption (used in the construction of input-hiding obfuscation) is that the encoding function is one-way. That is, given the output of $\mathsf{Enc}(\mathsf{pp}, \alpha)$ for randomly generated parameters pp and a random ring element $\alpha \in_R R$, it is hard to find α.

Our second hardness assumption (used in the construction of perfect circuit-hiding obfuscation) is a new assumption called *perfectly-hiding multilinear encoding*. The assumption states that given level 1 encodings of r and $r \cdot k$ for random $r, k \in R$, the value of any one bit predicate of k cannot be efficiently learned. The perfectly-hiding multilinear encoding assumption can be shown to hold in an ideal model where the encoding is generic (such as the generic ideal models described in [7, 2]).

Assumption 3.1 (perfectly-hiding multilinear encoding). *For every PPT adversary $\mathcal{A}$ that outputs one bit, the following ensembles are computationally indistinguishable:*

- $\left\{\mathsf{pp}, k, \mathcal{A}(\mathsf{Enc}(\mathsf{pp}, r), \mathsf{Enc}(\mathsf{pp}, r \cdot k)) : k, r \leftarrow R, \mathsf{pp} \leftarrow \mathsf{Gen}(1^n, 1^d)\right\}$,
- $\left\{\mathsf{pp}, k, \mathcal{A}(\mathsf{Enc}(\mathsf{pp}, r), \mathsf{Enc}(\mathsf{pp}, r \cdot k')) : k, k', r \leftarrow R, \mathsf{pp} \leftarrow \mathsf{Gen}(1^n, 1^d)\right\}$.

We note that both of the above assumptions *do not hold* for the candidate construction of [14] (assuming there is an efficient encoding procedure for every ring element). However, they are possibly satisfied by other constructions. In particular, to our knowledge there are no known attacks violating this assumption for the candidate construction of [12].

3.2 Input-hiding Obfuscation

Let $\mathcal{C}_n$ be an evasive collection defined by the arithmetic circuit $U(k,x)$, $k \in \mathbb{Z}_p^\ell, x \in \mathbb{Z}_p^m$ of degree $d = \mathrm{poly}(n)$. The obfuscator will make use of a d-symmetric graded encoding scheme over the ring $\mathbb{Z}_p$. For every $k \in \mathbb{Z}_p^\ell$ the obfuscation $\mathcal{O}(C_k)$ generates parameters $\mathsf{pp} \leftarrow \mathsf{Gen}(1^n, 1^d)$ for the graded encoding, and outputs a circuit that has the public parameters pp and the encodings $\{\mathsf{Enc}(\mathsf{pp}, k_i)\}$ for $i \in [\ell]$ hardwired into it. The circuit $\mathcal{O}(C_k)$, on input $x \in \mathbb{Z}_p^m$, first generates encodings $\{\mathsf{Enc}(\mathsf{pp}, x_i)\}$ for $i \in [m]$, and uses the evaluation functions of the graded encoding system to compute an encoding $u \in S_d^{(U(k,x))}$. $\mathcal{O}(C_k)$ then uses the zero test to check whether $u \in S_d^{(0)}$. If so it outputs 1 and otherwise it outputs 0.

More concretely, the encoding $u \in S_d^{(U(k,x))}$ is obtained by computing the encoded value for every wire of the arithmetic circuit $U(k,x)$. For every gate in the circuit connecting the wires w_1 and w_2 to w_3, given encodings

$$u_{w_1} \in S_{d_1}^{(\alpha_1)}, u_{w_2} \in S_{d_2}^{(\alpha_2)} \quad ,$$

for the values on wires w_1 and w_2, an encoding u^{w_2} of the value on wire w_3 is computed as follows. If the gate is an addition gate we first obtain encodings

$$u'_{w_1} \in S_{\max(d_1,d_2)}^{(\alpha_1)}, u'_{w_2} \in S_{\max(d_1,d_2)}^{(\alpha_2)} \quad ,$$

by multiplying either u_{w_1} or u_{w_2} by the appropriate encoding of 1. The encodings u'_{w_1}, u'_{w_2} are added to obtain the encoding $u_{w_3} \in S_{\max(d_1,d_2)}^{(\alpha_1+\alpha_2)}$. If the gate is a multiplication gate, we multiply the encodings u_{w_1}, u_{w_2} to obtain the encoding $u_{w_3} \in S_{d_1+d_2}^{(\alpha_1 \cdot \alpha_2)}$. Note that the degree of the encoding computed for every is at most the degree of the polynomial computed by the wire and therefore does not exceed d.

Theorem 3.2. *$\mathcal{O}$ is an input-hiding obfuscator for $\mathcal{C}$, assuming* Enc *is one-way.*

Proof. We need to prove that $\mathcal{O}$ satisfies both the functionality requirement and the security requirement. The functionality requirement follows immediately from the guarantees of the graded encoding scheme. Thus, we focus on proving the security requirement. To this end, fix any PPT adversary $\mathcal{A}$, and suppose for the sake of contradiction that for infinitely many values of n,

$$\Pr_{k \leftarrow \mathbb{Z}_p^\ell}[C_k(\mathcal{A}(\mathcal{O}(C_k))) = 1] \geq \frac{1}{\mathrm{poly}(n)} \quad . \tag{1}$$

Next we prove that there exists a PPT adversary $\mathcal{A}'$ that brakes the one-wayness of the encoding. The adversary $\mathcal{A}'$ will make use of the the helper procedure $\mathsf{Simplify}$ described in the following claim:

Claim 3.3. *There exists an efficient procedure* Simplify *that, given a multivariate non-trivial polynomial $P : \mathbb{Z}_p^\ell \to \mathbb{Z}_p$ of total degree d, represented as an arithmetic circuit, outputs a set of multivariate polynomials $\{P_j\}_{j\in[\ell]}$ (represented as arithmetic circuits) such that:*

1. *P_j is a multivariate non-trivial polynomial of total degree d.*
2. *For every $r \in \mathbb{Z}_p^\ell$ such that $P(\boldsymbol{r}) = 0$ there exist $j \in [\ell]$ such that $P(r) = 0$ but the univariate polynomial $Q(x) = P(r_1, \ldots, r_{j-1}, x, r_{j+1} \ldots, r_\ell)$ is non-trivial.*

We note that a very similar claim was proven by Bitansky *et al.* [5, Proposition 5.15]. We reprove it here for completeness

Proof (Claim 3.3). Given an arithmetic circuit computing a multivariate polynomial $P : \mathbb{Z}_p^\ell \to \mathbb{Z}_p$ of total degree d such that $P \not\equiv 0$, the procedure Simplify is as follows:

1. Set $P_1 = P$. repeat the following for $j = 1$ to ℓ.
2. Decompose P_j as follows:

$$P_j(k_j, \ldots, k_\ell) = \sum_{i=1}^{d} k_j^i \cdot P_{j,i}(k_{j+1}, \ldots, k_\ell).$$

3. Set P_{j+1} to be the non-zero polynomial $P_{j,i}$ with the minimal i.

Note that decomposing an arithmetic circuit into homogeneous components can be done efficiently. It is left to show that for every $r \in \mathbb{Z}_p^\ell$ if $P(r) = 0$ then there exists $j \in [\ell]$ such that

$$Q(x) = P_j(x, r_{j+1}, \ldots, r_\ell) \not\equiv 0 \ ,$$

and

$$P_j(r_j, r_{j+1}, \ldots, r_\ell) = 0.$$

The proof is by induction on j. The base case is when $j = 1$, for which it holds that:

$$P_1(x_1, \ldots, x_\ell \not\equiv 0$$
$$P_1(r_1, \ldots, r_\ell) = 0 \ .$$

For any $1 \le j < \ell$, suppose that:

$$P_j(x_j, \ldots, x_\ell) \not\equiv 0$$
$$P_j(r_j, \ldots, r_\ell) = 0$$
$$P_j(x, r_{j+1}, \ldots, r_\ell) \equiv 0 \ ;$$

then, by the construction of P_{j+1} from P_j,

$$P_{j+1}(x_{j+1}, \ldots, x_\ell) \not\equiv 0$$
$$P_{j+1}(r_{j+1}, \ldots, r_\ell) = 0 \ .$$

If this inductive process reaches P_ℓ, then it holds that:

$$P_\ell(x_\ell) \not\equiv 0$$
$$P_\ell(r_\ell) = 0 \ ,$$

which already satisfies the claim. □

The adversary $\mathcal{A}'$. $\mathcal{A}'$ is given the public parameters pp, and an encoding u of a random element $r \in \mathbb{Z}_p$. $\mathcal{A}'$ is defined as follows:

1. $\mathcal{A}'$ samples a random index $i \in [\ell]$ and a random element $k_j \leftarrow \mathbb{Z}_p$ for every $j \in [\ell] \setminus \{i\}$.
2. $\mathcal{A}'$ generates a random obfuscation $\mathcal{O}(C_k)$ from the encodings $\{\mathsf{Enc}(k_j)\}_{j\in[\ell]\setminus\{i\}}$ and using his input u as the encoding of k_i.
3. $\mathcal{A}'$ executes $\mathcal{A}(\mathcal{O}(C_k))$ and obtains an input x. If $\mathcal{O}(C_k)(x) \neq 1$, $\mathcal{A}'$ aborts.
4. Otherwise, $\mathcal{A}'$ executes the helper procedure $\mathsf{Simplify}$ on the polynomial $U(\cdot, x)$ with the values of x fixed and obtains the polynomials $\{P_j\}_{j\in[\ell]}$.
5. $\mathcal{A}'$ constructs the univariate polynomial $Q(x) = P_i(k_1, \dots, k_{i-1}, x, k_{i+1}, \dots, k_\ell)$ (the rest of the elements of k are known to $\mathcal{A}'$).
6. If $Q \equiv 0$, $\mathcal{A}'$ aborts, otherwise it outputs a random root of Q.

We show that $\mathcal{A}'$ outputs the correct value of r with noticeable probability. By our assumption on $\mathcal{A}$, $\mathcal{O}(C_k)(x) \neq 1$ with some noticeable probability ϵ. In this case, it follows from the correctness of $\mathcal{O}$ that $U(k, x) = 0$. Let j be the index guaranteed by Claim 3.3. Since the distribution of $k_1, \dots, k_\ell$ is independent from the choice of i, it follows that conditioned on the event $U(k, x) = 0$, $i = j$ with probability $1/\ell$. In this case by Claim 3.3 it holds that $P_i(k) = 0$ but the univariate polynomial P defined above is not identically zero, which means that r is one of the at most d roots of P. Therefore, $\mathcal{A}'$ will output the correct root with probability at least $\epsilon/(\ell d)$. □

3.3 Perfect Circuit-Hiding Obfuscation

Let $\mathcal{C}_n$ be an evasive collection defined by the arithmetic circuit $U(k, x)$, $k \in \mathbb{Z}_p^\ell, x \in \mathbb{Z}_p^m$ of depth degree $h = O(\log(n))$. The obfuscator will make use of a d-symmetric graded encoding scheme for $d = 2^h$ over the ring $\mathbb{Z}_p$. For every $k \in \mathbb{Z}_p^\ell$, the obfuscation $\mathcal{O}(C_k)$ generates parameters $\mathsf{pp} \leftarrow \mathsf{Gen}(1^n, 1^d)$ for the graded encoding, samples random elements $r_1, \dots r_\ell \in \mathbb{Z}_p$, and outputs a circuit that has the public parameters pp hardwired into it, and for every for $i \in [\ell]$, has the encodings $\mathsf{Enc}(\mathsf{pp}, r_i)$ and $\mathsf{Enc}(\mathsf{pp}, r_i \cdot k_i)$ hardwired into it.

The circuit $\mathcal{O}(C_k)$, on input $x \in \mathbb{Z}_p^m$, does the following: For $i \in [m]$, it generates the encodings $\mathsf{Enc}(\mathsf{pp}, r'_i)$ and $\mathsf{Enc}(\mathsf{pp}, r'_i \cdot x_i)$ where $r'_i = 1$ (Note that when encoding the input we do not need r'_i to be random for security. We only use r'_i the make the encoding of the input x and the key k have the same format). Using the evaluation functions of the graded encoding system, $\mathcal{O}(C_k)$ then computes a pair of encodings $u_0 \in S_d^{(\tilde{r})}$ and $u_1 \in S_d^{(\tilde{r}\cdot U(k,x))}$ for some $\tilde{r} \neq 0$ that is a function of $r_1, \dots, r_\ell$. Finally, it uses the zero test to check whether $u_1 \in S_d^{(0)}$. If so it outputs 1 and otherwise it outputs 0.

We next elaborate on how this encoded output is computed. The circuit $\mathcal{O}(C_k)$ evaluates the arithmetic circuit $U(k, x)$ gate by gate, as follows: Fix any gate in the circuit connecting the wires w_1 and w_2 to w_3. Suppose that for wires w_1 and w_2 we have the pairs of encodings

$$(u_0^{w_1}, u_1^{w_1}) \in S_{d_1}^{(\tilde{r}_1)} \times S_{d_1}^{(\tilde{r}_1\cdot\alpha_1)}, \quad (u_0^{w_2}, u_1^{w_2}) \in S_{d_2}^{(\tilde{r}_2)} \times S_{d_2}^{(\tilde{r}_2\cdot\alpha_2)} \ ,$$

where $\tilde{r}_1, \tilde{r}_2 \neq 0$ (supposedly the value on the wire w_1 is α_1 and the value on wire w_2 is α_2). If the gate is a multiplication gate, one can compute an encoding for wire w_3, by simply computing the pair of encodings:

$$(u_0^{w_3}, u_1^{w_3}) \in S_{d_1+d_2}^{(\tilde{r}_1 \cdot \tilde{r}_2)} \times S_{d_1+d_2}^{(\tilde{r}_1 \cdot \tilde{r}_2 \cdot \alpha_1 \cdot \alpha_2)} .$$

If the gate is an addition gate we compute $u_0^{w_3}$ in the same way. We also compute the encodings:

$$u_1^{w_{3,1}}, u_1^{w_{3,2}} \in S_{d_1+d_2}^{(\tilde{r}_1 \cdot \tilde{r}_2 \cdot \alpha_1)} \times S_{d_1+d_2}^{(\tilde{r}_1 \cdot \tilde{r}_2 \cdot \alpha_2)} ,$$

which can then be added to get:

$$u_1^{w_3} \in S_{d_1+d_2}^{(\tilde{r}_1 \cdot \tilde{r}_2 \cdot (\alpha_1 + \alpha_2))} .$$

Note that in any case $\tilde{r}_1 \cdot \tilde{r}_2 \neq 0$. Also note that in the evaluation of every level of the circuit U the maximal degree of the encodings at most doubles and therefore it never exceeds $d = 2^h$.

Theorem 3.4. *$\mathcal{O}$ is a perfect circuit-hiding obfuscator for $\mathcal{C}$, assuming the encoding satisfies the perfectly-hiding multilinear encoding assumption.*

Proof. By Theorem 2.1, it suffices to prove that $\mathcal{O}$ is an average-case oracle indistinguishability obfuscator for $\mathcal{C}$. Namely, it suffices to prove that for every PPT adversary $\mathcal{A}$ that outputs a single bit,

$$\left\{(C_k, \mathcal{A}(\mathcal{O}(C_k))) \mid k \leftarrow \mathbb{Z}_p^\ell\right\} \approx_c \left\{(C_k, \mathcal{A}(\mathcal{O}(C_{k'}))) \mid k, k' \leftarrow \mathbb{Z}_p^\ell\right\} .$$

Suppose for the sake of contradiction there exists a PPT adversary $\mathcal{A}$ (that outputs a single bit), a distinguisher D, and a non-negligible function ϵ, such that

$$\left| \Pr_{k \leftarrow \mathbb{Z}_p^\ell}[D(k, \mathcal{A}(\mathcal{O}(C_k))) = 1] - \Pr_{k,k' \leftarrow \mathbb{Z}_p^\ell}[D(k, \mathcal{A}(\mathcal{O}(C_{k'}))) = 1] \right| \geq \epsilon(n) ,$$

Recall that the obfuscated circuit $\mathcal{O}(C_k)$ consists of the public parameters pp and from the encodings:

$$\{\mathsf{Enc}(\mathsf{pp}, r_i), \mathsf{Enc}(\mathsf{pp}, r_i \cdot k_i)\}_{i \in [\ell]} ,$$

where $r_1, \ldots, r_\ell \leftarrow \mathbb{Z}_p^*$. Since sampling encoding corresponding to random input wires is efficient, the equation above, together with a standard hybrid argument, implies that there exists $i \in [\ell]$, a PPT adversary $\mathcal{A}'$ (that outputs a single bit), a distinguisher D', and a non-negligible function ϵ', such that

$$\left| \begin{array}{l} \Pr_{k_i \leftarrow \mathbb{Z}_p}[D'(\mathsf{pp}, k_i, \mathcal{A}'(\mathsf{Enc}(\mathsf{pp}, r_i), \mathsf{Enc}(\mathsf{pp}, r_i \cdot k_i))) = 1] - \\ \Pr_{k_i, k_i' \leftarrow \mathbb{Z}_p}[D'(\mathsf{pp}, k_i, \mathcal{A}'(\mathsf{Enc}(\mathsf{pp}, r_i), \mathsf{Enc}(\mathsf{pp}, r_i \cdot k_i'))) = 1] \end{array} \right| \geq \epsilon'(n) ,$$

contradicting the perfectly-hiding multilinear encoding assumption. □

Remark 3.1 (On unifying the constructions). Under a stronger variant of the perfectly-hiding multilinear encoding assumption we can directly prove that the input hiding obfuscation construction presented in Section 3.2 is also perfect circuit-hiding. Intuitively, the stronger variant assumes that the the function Enc given a random input k already hides every predicate of k (without adding any additional randomization).

Assumption 3.5 (strong perfectly-hiding multilinear encoding). *For every PPT adversary $\mathcal{A}$ that outputs one bit, the following ensembles are computationally indistinguishable:*

– $\left\{\mathsf{pp}, k, \mathcal{A}(\mathsf{Enc}(\mathsf{pp}, k)) : k, \leftarrow R, \mathsf{pp} \leftarrow \mathsf{Gen}(1^n, 1^d)\right\}$,
– $\left\{\mathsf{pp}, k, \mathcal{A}(\mathsf{Enc}(\mathsf{pp}, k')) : k, k', \leftarrow R, \mathsf{pp} \leftarrow \mathsf{Gen}(1^n, 1^d)\right\}$.

Note that this strong perfectly-hiding multilinear encoding assumption cannot hold for a deterministic encoding function (unlike with the perfectly-hiding multilinear encoding assumption). Using the stronger assumption above, we can get perfect circuit-hiding obfuscation for a larger class of functions. Specifically, $U(k, x)$ can be any arithmetic circuit computing a polynomial of degree $\mathrm{poly}(n)$, removing the logarithmic depth restriction.

4 Evasive Function and Virtual Grey Box Obfuscation

We show that average-case VGB obfuscation for all evasive functions implies a slightly weaker form of average-case VGB obfuscation for *all* function.

We start by giving a slightly more general definition for VGB obfuscation that considers also computationally unbounded obfuscators and allows for the query complexity of the simulator to be super-polynomial. Note that when the obfuscator is unbounded, we need to explicitly require that it has a polynomial slowdown, that is, the that the obfuscated circuit is not too large.

Definition 4.1 (Weak Average-Case VGB Obfuscation). *Let $\mathcal{C} = \{\mathcal{C}_n\}_{n\in\mathbb{N}}$ be a collection of circuits such that every $C \in \mathcal{C}_n$ is a circuit of size $\mathrm{poly}(n)$ that takes n bits as input. A (possibly inefficient) algorithm $\mathcal{O}$ is a weak average-case VGB obfuscator for $\mathcal{C}$ if it satisfies the following requirements:*

– *Functionality: for all $C \in \mathcal{C}$, $\mathcal{O}(C)$ outputs a description of a circuit that computes the same function as C.*
– *Polynomial Slowdown: There exist a polynomial p such that for every $C \in \mathcal{C}$, $|\mathcal{O}(C)| < p(|C|)$.*
– *Security: For every super-polynomial function $q = q(n)$ and for every PPT adversary $\mathcal{A}$ there exist a (possibly inefficient) simulator Sim and a negligible function μ such that for every predicate $\mathcal{P}$, every $n \in \mathbb{N}$ and every auxiliary input $z \in \{0,1\}^{\mathrm{poly}(n)}$ to $\mathcal{A}$:*

$$\left| \Pr_{C \leftarrow \mathcal{C}_n}[\mathcal{A}(z, \mathcal{O}(C)) = \mathcal{P}(C)] - \Pr_{C \leftarrow \mathcal{C}_n}[\mathsf{Sim}^{C[q(n)]}(z, 1^n) = \mathcal{P}(C)] \right| \leq \mu(n) \ .$$

Where $C[q(n)]$ denotes an oracle to the circuit C which allows at most $q(n)$ queries.

Remark 4.1 (On obfuscation with inefficient obfuscator). The notion of obfuscation with computationally unbounded obfuscators was first considered in [3]. To demonstrate the meaningfulness of this notion we note that assuming the existence of *indistinguishability obfuscation* for a collection $\mathcal{C}$ with an *effficent* obfuscator, the existence of a (weak) average-case VGB Obfuscation for $\mathcal{C}$ with a computationally unbounded obfuscator already implies the existence of a (weak) average-case VGB Obfuscation for $\mathcal{C}$ with an *effficent* obfuscator.

Theorem 4.1. *If there exists an average-case VGB obfuscator for every collection of evasive circuits, then there exists a weak average-case VGB obfuscator for every collection of circuits.*

Proof overview of Theorem 4.1. Let $\mathcal{C}$ be a (non-evasive) collection of circuit that we want to VGB obfuscate. We start by showing a computationally unbounded leaning algorithm $\mathcal{L}$ that given oracle access to a circuit $C \in \mathcal{C}$ tries to learn the circuit C. Clearly, If $\mathcal{L}$ can make unbounded number of queries it can learn C exactly. However, if the number of queries $\mathcal{L}$ makes to C is bounded by some super-polynomial function $q(n)$, we show that $\mathcal{L}$ can still learn a circuit $C' \in \mathcal{C}$ that is "close" to C. That is, C and C' only disagree on some negligible fraction of the inputs.

The learning algorithm $\mathcal{L}$ will repeatedly query C on the input that gives maximal information about the circuit C, taking into account the information gathered from all the previous oracle answers. $\mathcal{L}$ stops when it learns C or when there is no query that will give "enough" information about C. In this case, we denote by $K(C)$ the set of all circuits in $\mathcal{C}_n$ that are consistent with all the previous oracle answers. We show that all the circuits in $K(C)$ are close to C, and $\mathcal{L}$ will just output a random circuit $C' \in K(C)$.

The high-level idea behind the construction of the weak average-case VGB obfuscator $\mathcal{O}$ for $\mathcal{C}$ is that given black box access to a random circuit C a weak VGB simulator, that is, an unbounded adversary that can make at most $q(n)$ oracle queries to C, is able to run the learning algorithm $\mathcal{L}$ and learn the set $K(C)$. Therefore, a secure obfuscation $\mathcal{O}(C)$ of C does not need to hide the set $K(C)$. In particular, $\mathcal{O}(C)$ may contain a random circuit $C' \leftarrow K(C)$ in the clear. To satisfy the functionality requirement, the obfuscation cannot contain only C'. Additionally, $\mathcal{O}(C)$ will contain the circuit C_{diff}, where $C_{\mathsf{diff}} = C \oplus C'$ is a circuit that outputs 1 on every input on which C and C' differ. Now, to evaluate C on an input x an evaluator can compute $C(x) = C'(x) \oplus C_{\mathsf{diff}}(x)$. Clearly, $\mathcal{O}(C)$ cannot contain C_{diff} in the clear, since C_{diff} depends on C. instead, $\mathcal{O}(C)$ will obfuscate C_{diff} using the VGB obfuscator for evasive collections. Since C' is a random function in $K(C)$ it only differs from C on a negligible fraction of the inputs, and therefore C_{diff} outputs 1 only on a negligible fraction of the inputs.

Unfortunately, this high-level idea does not quite work. The problem is that since $\mathcal{O}(C)$ contains the circuit C' in the clear, we cannot argue that C_{diff} is taken at random from an evasive collection. In particular, given C it may be easy to find an input where C_{diff} outputs 1. For example, if C outputs 1 only on a single input x, and C' outputs 1 only on a single input x', then C_{diff} will output 1 on both inputs x and x'. Now, given the circuit C' it is easy find the input x' such that $C_{\mathsf{diff}}(x') = 1$ and therefore we do not know how to securely obfuscate C_{diff}.

To fix this problem we do not choose C' to be a random circuit in the set $K(C)$, but instead C' will be a circuit computing the majority function on many random circuits taken from the set $K(C)$. Now we can show that even given the circuit C' it is hard to find an input where C and C' differ, and therefore the obfuscation of C_{diff} is secure.

Proof (Theorem 4.1). Let $\mathcal{O}$ be an average-case VGB secure obfuscator for every collection of evasive circuits. Let $\mathcal{C} = \{\mathcal{C}_n\}_{n\in\mathbb{N}}$ be a (not necessarily evasive) collection of circuits such that every $C \in \mathcal{C}_n$ is a circuit of size $p(n)$, for some polynomial p, that takes n bits as input. Let $q(n)$ be any super-polynomial function. We construct a weak average-case VGB obfuscator $\mathcal{O}'$ for $\mathcal{C}$ where the simulator makes at most $q(n)$ queries to its oracle. $\mathcal{O}'$ will make use of the following learning algorithm $\mathcal{L}$ as a subroutine. The algorithm $\mathcal{L}$ is an inefficient algorithm that has oracle access to C, it queries its oracle at most $q'(n)$ times for $q'(n) \triangleq \frac{q(n)}{(n+1)}$, and outputs a circuit that is "close" to C.

Loosely speaking, algorithm $\mathcal{L}$ starts by setting K to be the set of all circuits in $\mathcal{C}_n$. Then, in each iteration $\mathcal{L}$ reduces the size of K so that, at the end, it contains only circuits that are "close" to C. The number of iterations is at most $q'(n)$ and in each iteration $\mathcal{L}$ makes a single oracle call. However, the computation done by $\mathcal{L}$ in each iteration is inefficient.

Formally, the algorithm $\mathcal{L}$ is defined as follows:

1. Set $K \leftarrow \mathcal{C}_n$.
2. For every $b \in \{0,1\}$ and every $x \in \{0,1\}^n$, compute:
$$p_x^b = \Pr_{C' \leftarrow K}[C'(x) = b], \quad p_x = \min(p_x^0, p_x^1) \ .$$
3. Set $x^* = \arg\max_x p_x$.
4. If $p_{x^*} < \frac{p(n)}{q'(n)}$, then return a random $C' \leftarrow K$.
5. Else, query the oracle on x^* and set:
$$K \leftarrow K \cap \{C' | C(x^*) = C'(x^*)\} \ .$$
6. If K contains a single element C, return C.
7. Else, goto Step 2.

We later argue that the output of $\mathcal{L}^C$ is a circuit that is close to C, that is, the circuits only disagree on a negligible fraction of the inputs. Next, we describe a weak average-case VGB obfuscator $\mathcal{O}'$ for $\mathcal{C}$. In the description of $\mathcal{O}'$, we use the following notation: we denote by $C_1 \oplus C_2$ the circuit that is composed from the circuits C_1 and C_2 where the output wires of these circuits are connected by a XOR gate. Similarly, we we denote by $\mathsf{Majority}_{i\in[n]} C_i$ the circuit that is composed from the circuits $C_1, \dots, C_n$ where the output wires of these circuits are connected by a $\mathsf{Majority}$ gate.

Note about notation. For any two circuits C and C', we use $C \equiv C'$ to denote that C and C' are functionally equivalent. That is, the circuits compute the same function, but may be very different as "formal" circuits. We use the notation $C = C'$ when C and C' are not only functionally equivalent, but are also equal as "formal" circuits.

The obfuscator. The obfuscator $\mathcal{O}'$ on input $C \in \mathcal{C}$ operates as follows:

1. For $i \in [n]$, set $C_i \leftarrow \mathcal{L}^C$ where all executions of $\mathcal{L}$ use independent randomness.
2. Construct the circuit $C_{\mathsf{maj}} = \mathsf{Majority}_{i\in[n]} C_i$.
3. Construct the circuit $C_{\mathsf{diff}} = C_{\mathsf{maj}} \oplus C$.
4. Construct and output the circuit $C_{\mathsf{out}} = C_{\mathsf{maj}} \oplus \mathcal{O}(C_{\mathsf{diff}})$.

The correctness and polynomial slowdown properties of $\mathcal{O}'$ follow from those of $\mathcal{O}$, and from the fact that the circuits in $\mathcal{C}_n$ are of (approximately) the same size.

To show that $\mathcal{O}'$ is a weak average-case VGB secure, we demonstrate an inefficient simulator Sim. For every PPT adversary $\mathcal{A}$, and for every auxiliary input $z \in \{0,1\}^{\mathrm{poly}(n)}$ to $\mathcal{A}$, Sim is given z, and oracle access to $C[q(n)]$. Sim acts as follows:

1. For $i \in [n]$, set $C_i \leftarrow \mathcal{L}^{C[q'(n)]}$, where independent randomness is used in different executions of $\mathcal{L}$.
2. Construct the circuit $C'_{\mathsf{maj}} = \mathsf{Majority}_{i\in[n]}\, C_i$.
3. Sample $C' \leftarrow \mathcal{L}^{C[q'(n)]}$ and construct the circuit $C'_{\mathsf{diff}} = C'_{\mathsf{maj}} \oplus C'$.
4. Construct the circuit $C_{\mathsf{sim}} = C'_{\mathsf{maj}} \oplus \mathcal{O}(C'_{\mathsf{diff}})$.
5. Execute $\mathcal{A}(z, C_{\mathsf{sim}})$ and output the result.

Sim invokes the learning algorithm, $\mathcal{L}$, $n+1$ times, and each invocation may include $q'(n) = \frac{q(n)}{(n+1)}$ queries to the oracle. Thus, in total Sim makes at most $q(n)$ oracle calls.

Another note about notation. In the rest of the proof, C denotes a random circuit in $\mathcal{C}_n$ (unless specified otherwise we assume C is distributed uniformly among all circuits in $\mathcal{C}_n$). The random variables C_{maj}, C_{diff} and C_{out} represent the value of the corresponding local variable in a random execution of the obfuscator $\mathcal{O}'$ on input C (this random variable is both over the random choice of C and of the coins used by $\mathcal{O}'$). Similarly, the random variables C'_{maj}, C'_{diff} and C_{sim} represent the value of the corresponding local variable in a random execution of $\mathsf{Sim}^{C[q(n)]}$ (the value of these random variables does not depend on the auxiliary input z passed to Sim). For simplicity of notation, in the rest of the proof, we omit the auxiliary input z from the parameter list of $\mathcal{A}$ and of Sim.

The simulator Sim is valid if for every predicate $\mathcal{P}$:

$$\left| \Pr_{C\leftarrow\mathcal{C}_n}[\mathcal{A}(\mathcal{O}'(C)) = \mathcal{P}(C)] - \Pr_{C\leftarrow\mathcal{C}_n}[\mathsf{Sim}^{C[q(n)]}(1^n) = \mathcal{P}(C)] \right| \le \mathrm{negl}(n) \ .$$

That is, if:

$$\left| \begin{array}{l} \Pr_{C\leftarrow\mathcal{C}_n}[\mathcal{A}((C_{\mathsf{maj}}, \mathcal{O}(C_{\mathsf{diff}}))) = \mathcal{P}(C)] \\ -\Pr_{C\leftarrow\mathcal{C}_n}[\mathcal{A}((C'_{\mathsf{maj}}, \mathcal{O}(C'_{\mathsf{diff}}))) = \mathcal{P}(C)] \end{array} \right| \le \mathrm{negl}(n) \ . \tag{2}$$

Before proving Equation (2), we introduce the following notation. For every circuit $C \in \mathcal{C}_n$, let $K(C)$ be the value of the set K when $\mathcal{L}^C$ terminates. Note that once C is fixed $K(C)$ is fully determined; indeed, up to step Step 4, where $\mathcal{L}$ outputs a random circuit from $K(C)$, $\mathcal{L}$ is deterministic. Define $\mathsf{GOOD}(C, C_{\mathsf{maj}})$ to be the event that:

$$C_{\mathsf{maj}} \equiv \mathop{\mathsf{Majority}}_{C'\in K(C)} C'$$

The following three lemmas imply Equation (2), and thus conclude the proof.

Lemma 4.1. *For every circuit $C \in \mathcal{C}_n$, the variables C_{maj} and C'_{maj} are identically distributed.*

Lemma 4.2.

$$\Pr_{C\leftarrow\mathcal{C}_n}[\mathsf{GOOD}(C, C_{\mathsf{maj}})] \ge 1 - \mathrm{negl}(n) \ .$$

Lemma 4.3. *For every C^*_{maj} in the support of C_{maj}, i.e., such that:*

$$\Pr_{C \leftarrow \mathcal{C}_n}[C_{\mathsf{maj}} = C^*_{\mathsf{maj}}] > 0 \ \ ,$$

it holds that:

$$\left| \begin{array}{l} \Pr_{C \leftarrow \mathcal{C}_n}\left[\mathcal{A}((C_{\mathsf{maj}}, \mathcal{O}(C_{\mathsf{diff}}))) = \mathcal{P}(C) \middle| \begin{array}{c} C_{\mathsf{maj}} = C^*_{\mathsf{maj}} \\ \mathsf{GOOD}(C, C^*_{\mathsf{maj}}) \end{array} \right] \\ - \Pr_{C \leftarrow \mathcal{C}_n}\left[\mathcal{A}((C'_{\mathsf{maj}}, \mathcal{O}(C'_{\mathsf{diff}}))) = \mathcal{P}(C) \middle| \begin{array}{c} C'_{\mathsf{maj}} = C^*_{\mathsf{maj}} \\ \mathsf{GOOD}(C, C^*_{\mathsf{maj}}) \end{array} \right] \end{array} \right| \leq \operatorname{negl}(n) \ \ .$$

Proof (Lemma 4.1). C'_{maj} is computed by Sim in the same way that C_{maj} is computed by $\mathcal{O}'$ except that that Sim limits the learning algorithm $\mathcal{L}$ to make at most $q'(n)$ oracle queries. It is therefore sufficient to show that $\mathcal{L}$ makes at most $q'(n)$ queries. By the choice of x^*, in every execution of $\mathcal{L}$, at Step 5, the size of the set K reduces by a factor of at least $(1 - p^*_x) \geq 1 - \frac{p(n)}{q'(n)}$. Since $|\mathcal{C}_n| \leq 2^{p(n)}$, after $q'(n)$ queries K must contain a single element and will thus $\mathcal{L}$ terminate. □

Proof (Lemma 4.2). Fix C, and denote by $\tilde{C}_{\mathsf{maj}}$ the circuit:

$$\tilde{C}_{\mathsf{maj}} \equiv \underset{C' \in K(C)}{\mathsf{Majority}} \, C' \ \ .$$

If $K(C)$ contains a single element (corresponding to Step 6 of $\mathcal{L}$), then GOOD must occur. Else, the stopping condition in Step 4 of $\mathcal{L}$ guarantees that for every $x \in \{0,1\}^n$, and letting $p^b_x = \Pr_{C'' \leftarrow K(C)}[C''(x) = b]$, it holds that $\min(p^0_x, p^1_x) < \frac{p(n)}{q'(n)}$. That is, almost all circuits in $K(C)$ agree with the majority value $\tilde{C}_{\mathsf{maj}}(x)$. Formally, for every $x \in \{0,1\}^n$,

$$\Pr_{C'' \leftarrow K(C)}[C''(x) \neq \tilde{C}_{\mathsf{maj}}(x)] < \frac{p(n)}{q'(n)} \ \ .$$

The event GOOD does not occur if and only if there exist $x \in \{0,1\}^n$ such that at least $n/2$ of the circuits $C_1, \ldots, C_n$ disagree with $\tilde{C}_{\mathsf{maj}}$ on x. Since every C_i is sampled independently from $K(C)$ and since q' is super-polynomial we have that:

$$\Pr_{C \leftarrow \mathcal{C}_n}[\neg \mathsf{GOOD}] \leq 2^n \cdot \binom{n}{\frac{n}{2}} \cdot \left(\frac{p(n)}{q'(n)} \right)^{\frac{n}{2}} < \left(\frac{16 p(n)}{q'(n)} \right)^{\frac{n}{2}} = \operatorname{negl}(n) \ \ .$$

□

Proof (Lemma 4.3). Fix C^*_{maj} such that:

$$\Pr_{C \leftarrow \mathcal{C}_n}[C_{\mathsf{maj}} = C^*_{\mathsf{maj}}] > 0 \ \ .$$

If $C_{\mathsf{maj}} = C^*_{\mathsf{maj}}$, the circuits have the exact same structure and therefore, C^*_{maj} is of the form $C^*_{\mathsf{maj}} = \mathsf{Majority}_{i \in [n]} \, C^*_i$ for some circuits $C^*_1, \ldots C^*_n \in \mathcal{C}_n$. The next claim will be useful for proving the lemma.

Claim 4.2.

1. *For every $C^* \in \mathcal{C}_n$ and every $C, C' \in K(C^*)$, the random variable C_{maj} in the execution of $\mathcal{O}(C)$ and the random variable C_{maj} in the execution of $\mathcal{O}(C')$ are identically distributed.*
2. *For every $C_1^*, \ldots, C_n^* \in \mathcal{C}_n$, for $C^*_{\mathsf{maj}} = \mathsf{Majority}_{i \in [n]} C_i^*$, and for every $C \notin K(C_1^*)$, C^*_{maj} is outside the support of C_{maj} (defined by the execution of $\mathcal{O}'(C)$).*

To prove Claim 4.2, we will use yet another simpler claim:

Claim 4.3. *For every $C^* \in \mathcal{C}_n$ and every $C \in K(C^*)$ it holds that $K(C) = K(C^*)$.*

Proof. Fix any $C^* \in \mathcal{C}_n$ and any $C \in K(C^*)$. Consider the set of oracle queries made by $\mathcal{L}^C$ and by $\mathcal{L}^{C^*}$ and their answers. If the two query-answer sets are equal then $K(C) = K(C^*)$. Else, both $\mathcal{L}^C$ and $\mathcal{L}^{C^*}$ make some query x^* such that $C(x^*) \neq C^*(x^*)$. However, this contradicts the fact that $C \in K(C^*)$, which means that C agrees with C^* on all the queries performed by $\mathcal{L}^{C^*}$ in the formation of $K(C^*)$.

□

Proof (Claim 4.2). For Part 1, fix any $C^* \in \mathcal{C}_n$ and any $C, C' \in K(C^*)$. Claim 4.3 implies that $K(C) = K(C') = K(C^*)$. Since the output of $\mathcal{L}^C$ is just a random element in $K(C)$ if follows that the output of $\mathcal{L}^C$ and the output of $\mathcal{L}^{C'}$ are identically distributed, and therefore the random variable C_{maj} in the execution of $\mathcal{O}'(C)$ and the random variable C_{maj} in the execution of $\mathcal{O}'(C')$ are also identically distributed.

For Part 2, fix $C_1^*, \ldots, C_n^* \in \mathcal{C}_n$, and $C \notin K(C_1^*)$, and let $C^*_{\mathsf{maj}} = \mathsf{Majority}_{i \in [n]} C_i^*$. If $C_{\mathsf{maj}} = C^*_{\mathsf{maj}}$ (that is, the circuits are formally identical) then it must be that $C_1^* \in K(C)$. Indeed, because the circuits $C_{\mathsf{maj}}, C^*_{\mathsf{maj}}$ are formally equal, C_1^* equals a circuit $C_1 \in K(C)$ where $C_{\mathsf{maj}} = \mathsf{Majority}_{i \in [n]} C_i$. This, together with Claim 4.3, implies that $K(C) = K(C_1^*)$. Since it is always true that $C \in K(C)$, this also implies that $C \in K(C_1^*)$, contradicting our assumption.

□

We are now ready to prove Lemma 4.3. Recall that $C^*_{\mathsf{maj}} = \mathsf{Majority}_{i \in [n]} C_i^*$. Denote the set $K(C_1^*)$ by K^*. By Claim 4.2 (Part 2), the lemma's statement is equivalent to the following, where we sample C from K^* instead of from $\mathcal{C}_n$:

$$\left| \begin{array}{l} \Pr_{C \leftarrow K^*} \left[\mathcal{A}((C_{\mathsf{maj}}, \mathcal{O}(C_{\mathsf{diff}}))) = \mathcal{P}(C) \middle| \begin{array}{c} C_{\mathsf{maj}} = C^*_{\mathsf{maj}} \\ \mathsf{GOOD}(C, C^*_{\mathsf{maj}}) \end{array} \right] \\ - \Pr_{C \leftarrow K^*} \left[\mathcal{A}((C'_{\mathsf{maj}}, \mathcal{O}(C'_{\mathsf{diff}}))) = \mathcal{P}(C) \middle| \begin{array}{c} C'_{\mathsf{maj}} = C^*_{\mathsf{maj}} \\ \mathsf{GOOD}(C, C^*_{\mathsf{maj}}) \end{array} \right] \end{array} \right| \leq \mathrm{negl}(n) \ . \quad (3)$$

Let the adversary $\mathcal{A}'$ be $\mathcal{A}$ with C^*_{maj} hard-coded to it. That is, $\mathcal{A}'(\mathcal{O}(C_{\mathsf{diff}}))$ outputs $\mathcal{A}((C^*_{\mathsf{maj}}, \mathcal{O}(C_{\mathsf{diff}})))$. Now we can rewrite Equation (3) as:

$$\left| \begin{array}{l} \Pr_{C \leftarrow K^*} \left[\mathcal{A}'(\mathcal{O}(C_{\mathsf{diff}})) = \mathcal{P}(C) \middle| \begin{array}{c} C_{\mathsf{maj}} = C^*_{\mathsf{maj}} \\ \mathsf{GOOD}(C, C^*_{\mathsf{maj}}) \end{array} \right] \\ - \Pr_{C \leftarrow K^*} \left[\mathcal{A}'(\mathcal{O}(C'_{\mathsf{diff}})) = \mathcal{P}(C) \middle| \begin{array}{c} C'_{\mathsf{maj}} = C^*_{\mathsf{maj}} \\ \mathsf{GOOD}(C, C^*_{\mathsf{maj}}) \end{array} \right] \end{array} \right| \leq \mathrm{negl}(n) \ . \quad (4)$$

Let $\mathcal{P}'$ be a predicate that has C^*_{maj} hardwired into it, and is defined as follows: On inputs of the form $C_{\mathsf{diff}} = C^*_{\mathsf{maj}} \oplus C$ where $\mathsf{GOOD}(C, C^*_{\mathsf{maj}})$ holds, the predicate $\mathcal{P}'(C_{\mathsf{diff}})$ outputs $\mathcal{P}(C)$. On all other inputs the output of $\mathcal{P}'$ is arbitrarily defined to be 0. Now we can rewrite Equation (4) as:

$$\left| \begin{array}{l} \Pr_{C \leftarrow K^*}\left[\mathcal{A}'(\mathcal{O}(C_{\mathsf{diff}})) = \mathcal{P}'(C_{\mathsf{diff}}) \middle| \begin{array}{c} C_{\mathsf{maj}} = C^*_{\mathsf{maj}} \\ \mathsf{GOOD}(C, C^*_{\mathsf{maj}}) \end{array} \right] \\ - \Pr_{C \leftarrow K^*}\left[\mathcal{A}'(\mathcal{O}(C'_{\mathsf{diff}})) = \mathcal{P}'(C_{\mathsf{diff}}) \middle| \begin{array}{c} C'_{\mathsf{maj}} = C^*_{\mathsf{maj}} \\ \mathsf{GOOD}(C, C^*_{\mathsf{maj}}) \end{array} \right] \end{array} \right| \leq \operatorname{negl}(n) \ . \quad (5)$$

Recall that $\mathcal{O}'$ sets $C_{\mathsf{diff}} = C_{\mathsf{maj}} \oplus C$. Let $\mathcal{C}_{\mathsf{diff}}$ be the collection:

$$\mathcal{C}_{\mathsf{diff}} = \left\{ C_{\mathsf{diff}} = C_{\mathsf{maj}} \oplus C \;\middle|\; \begin{array}{l} C \leftarrow K^* \\ C_{\mathsf{maj}} = C^*_{\mathsf{maj}} \\ \mathsf{GOOD}(C, C^*_{\mathsf{maj}}) \end{array} \right\}_{n \in \mathbb{N}, C^*_{\mathsf{maj}}} .$$

Additionally, recall that Sim sets $C'_{\mathsf{diff}} = C'_{\mathsf{maj}} \oplus C'$ where $C' \leftarrow \mathcal{L}^C$. Let $\mathcal{C}'_{\mathsf{diff}}$ be the collection:

$$\mathcal{C}'_{\mathsf{diff}} = \left\{ C'_{\mathsf{diff}} = C'_{\mathsf{maj}} \oplus C' \;\middle|\; \begin{array}{l} C \leftarrow K^* \\ C' \leftarrow \mathcal{L}^C \\ C'_{\mathsf{maj}} = C^*_{\mathsf{maj}} \\ \mathsf{GOOD}(C, C^*_{\mathsf{maj}}) \end{array} \right\}_{n \in \mathbb{N}, C^*_{\mathsf{maj}}} .$$

Now we can rewrite Equation (5) as:

$$\left| \begin{array}{l} \Pr_{C_{\mathsf{diff}} \leftarrow \mathcal{C}_{\mathsf{diff}}}[\mathcal{A}'(\mathcal{O}(C_{\mathsf{diff}})) = \mathcal{P}'(C_{\mathsf{diff}})] \\ - \Pr_{C'_{\mathsf{diff}} \leftarrow \mathcal{C}_{\mathsf{diff}}, C_{\mathsf{diff}} \leftarrow \mathcal{C}'_{\mathsf{diff}}}[\mathcal{A}'(\mathcal{O}(C'_{\mathsf{diff}})) = \mathcal{P}'(C_{\mathsf{diff}})] \end{array} \right| \leq \operatorname{negl}(n) \ . \quad (6)$$

By the proof of Claim 4.2, for any circuit $C \in K^*$, it holds that $K(C) = K^*$. Noting that C, C' defined in the collections $\mathcal{C}_{\mathsf{diff}}$ and $\mathcal{C}'_{\mathsf{diff}}$ are random circuits in K^*, and thus the two collections are identical. We can now rewrite Equation (6) as:

$$\left| \begin{array}{l} \Pr_{C_{\mathsf{diff}} \leftarrow \mathcal{C}_{\mathsf{diff}}}[\mathcal{A}'(\mathcal{O}(C_{\mathsf{diff}})) = \mathcal{P}'(C_{\mathsf{diff}})] \\ - \Pr_{C'_{\mathsf{diff}}, C_{\mathsf{diff}} \leftarrow \mathcal{C}_{\mathsf{diff}}}[\mathcal{A}'(\mathcal{O}(C'_{\mathsf{diff}})) = \mathcal{P}'(C_{\mathsf{diff}})] \end{array} \right| \leq \operatorname{negl}(n) \ . \quad (7)$$

$\square$

To prove equation Equation 7 and conclude the proof of the lemma, we show that the collection $\mathcal{C}_{\mathsf{diff}}$ is evasive. This, together with the fact that $\mathcal{O}$ is average-case VGB secure evasive collections and the proof of Theorem 2.1, imply that Equation 7 holds.

Claim 4.4. *The collection $\mathcal{C}_{\mathsf{diff}}$ is evasive.*

Proof. Let C be the random variable given in the definition of $\mathcal{C}_{\mathsf{diff}}$. If K^* contains a single element (corresponding to Step 6 of $\mathcal{L}$) then $C \equiv C_{\mathsf{maj}}$ and the collection $\mathcal{C}_{\mathsf{diff}}$ contains, in fact, only the all-zero function, and is therefore evasive. Assuming $|K^*| > 1$, the stopping condition in Step 4 of $\mathcal{L}$ guarantees that for every $x \in \{0,1\}^n$, letting $p_x^b = \Pr_{C'' \leftarrow K(C)}[C''(x) = b]$, it holds that for $\min(p_x^0, p_x^1) < \frac{p(n)}{q'(n)}$. This implies:

$$\Pr_{C'' \leftarrow K(C)}\left[C''(x) \neq \underset{\bar{C} \in K(C)}{\mathsf{Majority}}\, \bar{C}(x)\right] < \frac{p(n)}{q'(n)} \ .$$

By the proof of Claim 4.2, $K(C) = K^*$ for every $C \in K^*$, and therefore also

$$\Pr_{C'' \leftarrow K^*}[C''(x) \neq \underset{\bar{C} \in K^*}{\mathsf{Majority}}\, \bar{C}(x)] < \frac{p(n)}{q'(n)} \ .$$

Plugging-in the definitions of C_{diff} and of $\mathsf{GOOD}(C, C^*_{\mathsf{maj}})$, we get that for every $x \in \{0,1\}^n$,

$$\Pr_{C \leftarrow K^*}[C_{\mathsf{diff}}(x) = 1] = \Pr_{C \leftarrow K^*}[C(x) \neq C^*_{\mathsf{maj}}(x)] < \frac{p(n)}{q'(n)} = \frac{p(n)}{n^{\omega(1)}} = \mathrm{negl}(n),$$

which implies that $\mathcal{C}_{\mathsf{diff}}$ is evasive. □

□

5 Impossibility Results

Definitions 2.5 and 2.4 only consider circuit obfuscation with average-case security. In this section we give impossibility results for obfuscating evasive *Turing machines* and for obfuscating evasive circuits with *worst-case* security.

5.1 Impossibility of Turing Machine Obfuscation

Barak et. al. [3] show the impossibility general obfuscation of circuits and Turing machines. We show that the impossibility of Turing machines obfuscation can be extended to the case of evasive functions. Similarly to the result of [3], our negative result applies for VBB obfuscation as well as for weaker notions such as average-case obfuscation (see [3] for more details). In particular, we get an impossibility for the Turing machine versions of Definitions 2.5 and 2.4.

Let $\mathcal{M} = \{\mathcal{M}_n\}_{n \in \mathbb{N}}$ be a collection of Turing machines such that every $M \in \mathcal{M}_n$ has description of size $\mathrm{poly}(n)$ and outputs a bit. We say that $\mathcal{M}$ is *evasive* if given oracle access to a random machine in the collection it is hard to find an input that evaluates to 1.

Definition 5.1 (Evasive Turing Machine Collection). *A collection of Turing machines $\mathcal{M}$ is evasive if there exists a negligible function μ such that for every $n \in \mathbb{N}$ and every $x \in \{0,1\}^*$*

$$\Pr_{M \leftarrow \mathcal{M}_n}[M(x) = 1] \leq \mu(n) \ .$$

We start by recalling the syntax, functionality, and polynomial slowdown requirements for Turing machine obfuscation as defined in [3]. Then we give security definitions that are the Turing machine versions of Definitions 2.5 and 2.4.

Definition 5.2 (Turing Machine Obfuscation). *An obfuscator $\mathcal{O}$ for $\mathcal{M}$ is a PPT algorithm that satisfies the following requirements:*

– *Functionality: For every $n \in \mathbb{N}$ and every $M \in \mathcal{M}_n$, $\mathcal{O}(M)$ outputs a description of a Turing machine that computes the same function as M.*
– *Polynomial Slowdown: There exists a polynomial p such that for every $M \in \mathcal{M}$ and for every $x \in \{0,1\}^*$ if the running time of $M(x)$ is t, then the running time of $(\mathcal{O}(M))(x)$ is at most $p(t)$.*

Definition 5.3 (Perfect Turing-Machine-Hiding). *An obfuscator $\mathcal{O}$ for a collection of Turing machines $\mathcal{M}$ is perfect circuit-hiding if for every PPT adversary $\mathcal{A}$ there exist a PPT simulator* Sim *and a negligible function μ such that for every $n \in \mathbb{N}$ and every efficiently computable predicate $\mathcal{P}$:*

$$\left| \Pr_{M \leftarrow \mathcal{M}_n}[\mathcal{A}(\mathcal{O}(M)) = \mathcal{P}(M)] - \Pr_{M \leftarrow \mathcal{M}_n}[\mathsf{Sim}(1^n) = \mathcal{P}(M)] \right| \leq \mu(n) \ .$$

Definition 5.4 (Input Hiding). *A obfuscator $\mathcal{O}$ for a collection of Turing machines $\mathcal{M}$ is input hiding if there exists a negligible function μ such that for every $n \in \mathbb{N}$ and for every PPT adversary $\mathcal{A}$*

$$\Pr_{M \leftarrow \mathcal{M}_n}[M(\mathcal{A}(\mathcal{O}(M))) = 1] \leq \mu(n) \ .$$

The impossibility. The impossibility of [3] demonstrates a pair of functions $C_{\alpha,\beta}, D_{\alpha,\beta}$ such that given oracle access to these functions, it is impossible to learn the key (α, β). However, given any efficient implementation of $C_{\alpha,\beta}$ and $D_{\alpha,\beta}$ as a pair of Turing machines, it is possible to learn (α, β). The two functions are then combined into a single function that cannot be obfuscated. The idea is to "embed" the functions $C_{\alpha,\beta}$ and $D_{\alpha,\beta}$ of [3] inside an evasive Turing machine.

For a key $\alpha, \beta \in \{0,1\}^n$ define the machine $C_{\alpha,\beta}$ as follows:

$$C_{\alpha,\beta}(x; i) = \begin{cases} \beta_i & \text{if } x = \alpha \\ 0 & \text{otherwise} \end{cases}$$

The machine $D_{\alpha,\beta}$ takes as input a description of a machine C that is suppose to run in time $p(n)$ and checks whether C computes the same function as $C_{\alpha,\beta}$ on the inputs $\{(\alpha, i)\}_{i \in [n]}$. Namely,

$$D_{\alpha,\beta}(C) = \begin{cases} \beta_1 & \text{if } \forall i \in [n], C(\alpha, i) \text{ outputs } \beta_i \text{ within } p(n) \text{ steps} \\ 0 & \text{otherwise} \end{cases} \ .$$

The polynomial p is defined to be greater than the running time of $\mathcal{O}(C_{\alpha,\beta})$. Next we define a single machine $M_{\alpha,\beta}$ combining the machines $C_{\alpha,\beta}$ and $D_{\alpha,\beta}$, as follows:

$$M_{\alpha,\beta}(b, z) = \begin{cases} C_{\alpha,\beta}(z) & \text{if } b = 0 \\ D_{\alpha,\beta}(z) & \text{if } b = 1 \end{cases} \ .$$

It is straightforward to verify that $C_{\alpha,\beta}$ and $D_{\alpha,\beta}$ are evasive, and therefore $M_{\alpha,\beta}$ is also evasive. By construction, an adversary that is given $\mathcal{O}(M_{\alpha,\beta})$ can compute a description of machines M_C and M_D, computing $C_{\alpha,\beta}$ and $D_{\alpha,\beta}$ respectively, where the running time of M_C is at most p. The adversary can therefore execute $M_D(M_C)$ and obtain β_1 with probability 1. Note that a simulator (with no access to $M_{\alpha,\beta}$) can guess β_1 with probability at most $1/2$ and therefore $\mathcal{O}$ is not perfect circuit-hiding (Definition 5.3).

To show that $\mathcal{O}$ is not input hiding (Definition 5.4) we consider an adversary that produces the input $(1, M_C)$ to $M_{\alpha,\beta}$. Since $f_{\alpha,\beta}(1, M_C) = \beta_1$ and β is random, the adversary outputs a preimage of 1 with probability $1/2$.

5.2 Impossibility of Worst-Case Obfuscation

The impossibility of [3] for circuit obfuscation demonstrates a collection of circuits $\mathcal{C}_n = \{C_s\}_{s\in\{0,1\}^n}$ such that given oracle access to C_s for a random seed s it is impossible to learn s. However, given any circuit computing the same function as C_s, an adversary can learn s. In general we do not know how to "embed" $\mathcal{C}$ inside an evasive collection without loosing the above learnability property. However, such embedding is possible when the adversary has some partial knowledge about the seed of the circuit taken from the evasive collection. This type of attack can be used to rule out a worst-case security definition.

We recall the definition of worst-case VBB from [3]. We present an equivalent version of the definition that uses a predicate and resembles Definition 2.6 for average-case VBB. Note that a worst-case version of the input-hiding security definition (Definition 2.4) cannot hold against non-uniform adversaries.

Definition 5.5 (Worst-Case Virtual Black-Box (VBB) from [3]). *An obfuscator $\mathcal{O}$ for a collection of circuits $\mathcal{C}$ is perfect circuit-hiding in the worst-case if for every PPT adversary $\mathcal{A}$ there exists a PPT simulator* Sim *and a negligible function μ such that for every $n \in \mathbb{N}$, every $C \in \mathcal{C}_n$ and every predicate $\mathcal{P}$:*

$$\left|\Pr[\mathcal{A}(\mathcal{O}(C)) = \mathcal{P}(C)] - \Pr[\mathsf{Sim}^C(1^n) = \mathcal{P}(C)]\right| \le \mu(n) \ .$$

Let $\mathcal{C}_n = \{C_s\}_{s\in\{0,1\}^n}$ be the collection defined by [3]. For $\alpha, s \in \{0,1\}^n$ we define $C'_{\alpha,s}$ as follows:

$$C'_{\alpha,s}(x_1, x_2, i) = \begin{cases} [C_s(x_1)]_i & \text{if } x_2 = \alpha \\ 0 & \text{otherwise} \end{cases} .$$

First note that the collection $\mathcal{C}$ is evasive, since for every input (x_1, x_2, i) the probability over a random key (α, s) that $x_2 = \alpha$ is negligible. However, this circuit cannot be VBB obfuscated. There is an adversary that given an obfuscation of $C'_{\alpha,s}$ for $\alpha = 0^n$ and for a random s, can transform this obfuscation into a circuit computing the same function as C_s and thereby learn s. Conversely, every simulator that is given oracle access to $C'_{\alpha,s}$ for $\alpha = 0^n$ and for a random s, cannot learn more than what can be learned with oracle access to C_s, and in particular cannot learn s.

6 Acknowledgement

We thank Vijay Ganesh for suggesting to us the "software patch" problem.

References

[1] Barak, B., Bitansky, N., Canetti, R., Kalai, Y.T., Paneth, O., Sahai, A.: Obfuscation for evasive functions. Cryptology ePrint Archive, Report 2013/668 (2013), `http://eprint.iacr.org/`

[2] Barak, B., Garg, S., Kalai, Y.T., Paneth, O., Sahai, A.: Protecting obfuscation against algebraic attacks. Cryptology ePrint Archive, Report 2013/631 (2013), `http://eprint.iacr.org/`

[3] Barak, B., Goldreich, O., Impagliazzo, R., Rudich, S., Sahai, A., Vadhan, S.P., Yang, K.: On the (im)possibility of obfuscating programs. In: CRYPTO. pp. 1–18 (2001)

[4] Bitansky, N., Canetti, R.: On strong simulation and composable point obfuscation. In: CRYPTO. pp. 520–537 (2010)

[5] Bitansky, N., Chiesa, A., Ishai, Y., Ostrovsky, R., Paneth, O.: Succinct non-interactive arguments via linear interactive proofs. In: TCC. pp. 315–333 (2013)

[6] Brakerski, Z., Rothblum, G.N.: Obfuscating conjunctions. In: CRYPTO. pp. 416–434 (2013)

[7] Brakerski, Z., Rothblum, G.N.: Virtual black-box obfuscation for all circuits via generic graded encoding. Cryptology ePrint Archive, Report 2013/563 (2013), `http://eprint.iacr.org/`

[8] Canetti, R.: Towards realizing random oracles: Hash functions that hide all partial information. In: CRYPTO. pp. 455–469 (1997)

[9] Canetti, R., Goldreich, O., Halevi, S.: The random oracle methodology, revisited. J. ACM 51(4), 557–594 (2004)

[10] Canetti, R., Rothblum, G.N., Varia, M.: Obfuscation of hyperplane membership. In: TCC. pp. 72–89 (2010)

[11] Canetti, R., Vaikuntanathan, V.: Obfuscating branching programs using black-box pseudo-free groups. Cryptology ePrint Archive, Report 2013/500 (2013), `http://eprint.iacr.org/`

[12] Coron, J.S., Lepoint, T., Tibouchi, M.: Practical multilinear maps over the integers. In: CRYPTO (1). pp. 476–493 (2013)

[13] Ganesh, V., Carbin, M., Rinard, M.C.: Cryptographic path hardening: Hiding vulnerabilities in software through cryptography. CoRR abs/1202.0359 (2012)

[14] Garg, S., Gentry, C., Halevi, S.: Candidate multilinear maps from ideal lattices. In: EUROCRYPT. pp. 1–17 (2013)

[15] Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. In: FOCS (2013)

[16] Goldwasser, S., Rothblum, G.N.: On best-possible obfuscation. In: TCC. pp. 194–213 (2007)

[17] Hohenberger, S., Sahai, A., Waters, B.: Replacing a random oracle: Full domain hash from indistinguishability obfuscation. IACR Cryptology ePrint Archive 2013, 509 (2013)

[18] Sahai, A., Waters, B.: How to use indistinguishability obfuscation: Deniable encryption, and more. IACR Cryptology ePrint Archive 2013, 454 (2013)

[19] Valiant, L.G., Skyum, S., Berkowitz, S., Rackoff, C.: Fast parallel computation of polynomials using few processors. SIAM J. Comput. 12(4), 641–644 (1983)

[20] Wee, H.: On obfuscating point functions. IACR Cryptology ePrint Archive 2005, 1 (2005)