On Obfuscating Point Functions

Hoeteck Wee*
Computer Science Division
University of California, Berkeley

ABSTRACT

We investigate the possibility of obfuscating point functions in the framework of Barak et al. from Crypto '01. A point function is a Boolean function that assumes the value 1 at exactly one point. Our main results are as follows:

1. We provide a simple construction of efficient obfuscators for point functions for a slightly relaxed notion of obfuscation, for which obfuscating general circuits is nonetheless impossible. Our construction relies on the existence of a very strong one-way permutation, and yields the first non-trivial obfuscator under general assumptions in the standard model. We also obtain obfuscators for point functions with multi-bit output and for prefix matching.

2. Our assumption is that there is a one-way permutation wherein any polynomial-sized circuit inverts the permutation on at most a polynomial number of inputs. We show that a similar assumption is in fact necessary, and that our assumption holds relative to a random permutation oracle.

3. Finally, we establish two impossibility results which indicate that the limitations on our construction, namely simulating only adversaries with single-bit output and using nonuniform advice in our simulator, are in some sense inherent.

Previous work gave negative results for the general class of circuits (Barak et al., Crypto '01) and positive results in the random oracle model (Lynn et al., Eurocrypt '04) or under non-standard number-theoretic assumptions (Canetti, Crypto '97). This work represents the first effort to bridge the gap between the two for a natural class of functionalities.

*hoeteck@cs.berkeley.edu. Work supported by US-Israel BSF Grant 2002246.

STOC'05, May 22-24, 2005, Hunt Valley, Maryland, USA.

Categories and Subject Descriptors: E.0 [**Data**]: General.

General Terms: Theory, Security.

Keywords: Obfuscation

1. INTRODUCTION

1.1 Background

A fundamental problem in computer science is understanding the extent to which we can exploit access to the source code of a computer program, beyond simply executing the program (described by the code) on different inputs and observing its input/output behavior. As noted in [1], the hardness of the Halting Problem and Satisfiability seems to indicate that source code access yields few significant capabilities, if any at all, apart from black-box access, that is, the ability to run the program or circuit on inputs of one's choice.

A related problem is whether we can always (efficiently) rewrite programs in such a way as to eliminate any advantage to seeing the source code beyond gaining black-box access to the program. For this notion to be meaningful, we require that the rewritten program has the same functionality as the original one. We refer to such an algorithm for rewriting programs as an *obfuscator*, and the rewritten code as the *obfuscated* program. A program obfuscator, if one exists, has numerous potential applications in cryptography, such as software protection and digital water-marking, construction of homomorphic encryption schemes and realizing cryptographic protocols proven secure in the random oracle model [1].

A theoretical study of obfuscation was initiated in the seminal work of Barak et al. in [1]. The security requirement of an obfuscated program therein is that it behaves like a "virtual black box", namely anything one could efficiently compute from the obfuscated program, one should be able to efficiently compute given just oracle access to the program; this is formalized using the simulation paradigm. The main result of [1] is that it is impossible to achieve this notion of obfuscation. In particular, there exists an *unobfuscatable* family of functions – each function in the family has an associated attribute such that having source code access to programs computing these functions yields a significant advantage in computing this attribute over just oracle access to the programs. This leaves two main directions to pursue on the subject of obfuscation: exploring weaker but nonetheless meaningful notions of obfuscation, and constructing obfuscators for restricted, yet still non-

trivial and interesting, class of programs. We focus on the latter in this paper.

The class of programs we consider is that which computes equality. Such a program may be specified using a circuit I_x, containing a string x, which outputs 1 if its input matches x exactly, and 0 otherwise. This program computes a *point function*, namely a Boolean function that assumes the value 1 at exactly one point. We choose to focus on the class of point functions due to its simplicity, its relation to password-hiding algorithms commonly used in practice[1], and the abundance of related work [3, 5, 15] that we are able to build on. The simplest construction in previous work is that of Lynn et al. [15] for obfuscating point functions with a random oracle $\mathcal{R}$: the obfuscated program stores $\rho = \mathcal{R}(x)$ and on input y, outputs 1 iff $\mathcal{R}(y) = \rho$. Their analysis exploits the property that the evaluation of $\mathcal{R}$ at x is random and independent of its value at any other point.

1.2 Contributions and perspective

We initiate a systematic study of obfuscating point functions in the framework of [1].

1.2.1 Positive results for obfuscation

We show that a sufficient condition for obfuscating point functions is the existence of permutations such that for the specific task of inverting the permutation on a random input, one cannot do much better than treating the permutation as a black-box. The property we need of these permutations may be formalized as follows:

ASSUMPTION 1.1. *There exists a polynomial-time computable permutation $\pi : \{0,1\}^n \to \{0,1\}^n$ and a constant c such that for every polynomial $s = s(n)$ and every nonuniform PPT A of size s, for all sufficiently large n,*

$$\Pr_{x \in U_n}[A(\pi(x)) = x] \leq s^c/2^n$$

Starting from such a permutation π, we construct obfuscators for point functions by instantiating the random oracle in the construction of [15] with the following hash function:

$$h(x; \tau_1, \ldots, \tau_{3n}) = (\tau_1, \ldots, \tau_{3n}, \langle x, \tau_1 \rangle, \langle \pi(x), \tau_2 \rangle, \ldots, \langle \pi^{3n-1}(x), \tau_{3n} \rangle)$$

The simulator for our obfuscator works like the one in [3]: it has as nonuniform advice a set L which specifies the point functions for which the hash value reveals "too much" information to the adversary. On oracle access to a point function, the simulator checks if it corresponds to a function in L, and if so, it can simulate the adversary on the hash value obtained by a random obfuscation of that function. Otherwise, it simulates the adversary on a random string (which may not even correspond to a valid hash). In our final construction, we derandomize the above hash function using random walk on expanders.

1.2.2 The one-way permutation assumption

We show that there is an oracle relative to which Assumption 1.1 holds; in fact, with overwhelming probability over a random permutation π, the assumption holds relative to π. This is a stronger statement than the assertion that Assumption 1.1 holds in the random oracle model. While the hardness of inverting a random permutation is hardly surprising, the proof requires a careful analysis that improves on previous results on the hardness of inverting a random permutation [8]. Combined with our construction, this means that we can obfuscate point functions in the *non-programmable* random permutation oracle model [16], wherein virtual black-box property stipulates that the adversary's view be simulated with respect to the same oracle. On the other hand, the obfuscator in [15] (along with other positive results therein) only achieves the virtual black-box property in the programmable random oracle model. Our assumption is also qualitatively different from assuming a random oracle in that we specify a task with respect to which the permutation π "behaves" like a random oracle. This is an important distinction as there are schemes and tasks which can be securely realized in the random oracle model but are provably impossible if the random oracle is instantiated with any efficiently computable function ensemble [4, 16].

That said, we stress that we are not trying to argue that the assumption is "reasonable". After all, in the framework of "work", that is, running time of the adversary divided by its success probability, standard cryptographic assumptions assert hardness $n^{\omega(1)}$ for work, whereas we require a lower bound of $2^{n-O(\log n)}$. This bound is fairly close to the trivial upper bound: every efficiently computable function can be inverted with work $2^{n+O(\log n)}$. However, we are able to show that a qualitatively similar assumption is *necessary* for obfuscating point functions. We show that if there is an algorithm for solving CIRCUIT-SAT (circuit satisfiability, namely to determine if a given circuit has a satisfying assignment) that does noticeably better than exhaustive search[2], then an obfuscator for point functions does *not* exist. This narrows the gap between the computational assumptions that are necessary and those that are sufficient for non-trivial obfuscation to be possible.

1.2.3 Limitations of our simulator construction

Dependency on ϵ. We achieve a notion of virtual black-box which is somewhat weaker than the one in [1], and is equivalent to the one in [3]:

> (weak virtual black-box property) For any nonuniform PPT A and any function $\epsilon(n) = 1/n^{O(1)}$, there exists a nonuniform PPT S_A such that for all sufficiently large n and for all $x \in \{0,1\}^n$:
>
> $$\left| \Pr\left[A(\mathcal{O}(I_x)) = 1\right] - \Pr\left[S_A^{I_x}(1^n) = 1\right] \right| \leq \epsilon(n)$$

The difference from [1] is that we allow the size of the simulator to have an inverse polynomial dependency on the distinguishing probability. A similar relaxation has been used in the definition of zero-knowledge (e.g. [7]). We argue that this is sufficient to capture an intuitive and appealing notion of obfuscation: an adversary can

[1]The virtual black-box property guarantees that an adversary upon seeing the obfuscated password cannot do much better than running a dictionary attack [17].

[2]That we do not know any such algorithm indicates that for the purpose of finding a satisfying circuit for a given circuit (which can be solved in linear time given an oracle to the decision problem), one may not be able to do much better than treating the circuit as a black-box.

approximate whatever he learns from seeing an obfuscated circuit within any inverse polynomial accuracy at a price of a polynomial blow-up in its running time and advice. We stress that the impossibility result of [1] for obfuscating general circuits extends to this weaker definition.

Using nonuniform advice. Our simulator is inherently nonuniform, even for a uniform adversary. We hard-wire into the simulator nonuniform advice about the adversary and we do not know how to efficiently compute this advice given the description of the circuit A. We are able to show, however, that such advice exists and has a succinct description if Assumption 1.1 holds. More generally, we do not know an efficient transformation from adversaries to simulators, although we are able to bound the size of the simulator by a fixed polynomial in the size of the adversary (and the distinguishing probability). We feel that the use of nonuniformity is not a major short-coming in our simulator, since we can interpret our obfuscator as providing the guarantee that "anything an adversary can efficiently compute given the obfuscated circuit, he could also efficiently compute from observing the input/output behavior of the circuit, when given a small amount of help". (Refer to [9, Sec 4.3.3] for a discussion on how this notion is problematic in the context of zero-knowledge protocols.) We also prove that we cannot obfuscate point functions with simulators using black-box access to the adversary and a limited number of queries to the point function oracle which is independent of the size of the adversary.

Predicate adversaries. Our simulator only applies to a predicate adversary that computes a $\{0,1\}$-valued function of its input; that is, an adversary that is trying to decide some property of the original circuit. Note that simulating such an adversary already captures semantic security. As pointed out in [1], we would like (for positive results) for the simulator to satisfy a stronger requirement, namely to simulate the view of the adversary. We show that this is impossible for point functions. In fact, we show a stronger negative result: every family of circuits that can be obfuscated against general adversaries is efficiently and exactly learnable using membership queries. Point functions, in particular, are not learnable. Moreover, our argument relativizes, thereby ruling out obfuscating point functions against general adversaries even in the non-programmable random oracle model.

1.3 Additional related work

Obfuscation with number-theoretic assumptions. The problem of obfuscating point functions was first studied by Canetti [3], who presented several equivalent definitions, one of which is the same as the one used in this paper. In addition, he showed that the definition can be realized based on a strong variant of the Decisional Diffie-Hellman problem, namely that the problem remains hard if one of the inputs comes from any distribution of super-logarithmic min-entropy (instead of the uniform distribution). We note that this assumption has a flavor of pseudorandomness, whereas we start with a hardness assumption from which we derive pseudorandomness. The paper also introduced the hashing paradigm for obfuscating point functions used here and in [15].

Obfuscation with high min-entropy. Canetti et al. [5] and Dodis et al. [6] show how to achieve a weaker notion of virtual black-box (based on a definition in [3]) assuming just standard collision-resistant hash functions. Instead of requiring that the distinguishing probability be small for every $x \in \{0,1\}^n$, they require that the distinguishing probability be small averaging over x sampled from a distribution of sufficiently high min-entropy. One way of formalizing this is as follows, where $\delta \in (0,1)$ is a parameter:

> for any nonuniform PPT A, there exists a nonuniform oracle PPT S_A and a negligible function $\epsilon(n)$ such that for all sufficiently large n and for all distributions X over $\{0,1\}^n$ with min-entropy at least n^δ:
>
> $$\mathrm{E}_{x \in X} \left| \Pr\left[A(\mathcal{O}(I_x)) = 1\right] - \Pr\left[S_A^{I_x}(1^n) = 1\right] \right| \leq \epsilon(n)$$

The idea in [5, 6] is to hash x down to a string of length roughly n^δ using a strong extractor with a collision-resistant property. Such an extractor can be constructed by composing pair-wise independent hash functions with collision-resistant hash functions. The adversary's view of the hash value, for a random x chosen from a distribution of high min-entropy, can then be simulated using a random string. In particular, the simulator achieves security in an information-theoretic sense without making any query to the point function oracle and simulates general adversaries using just black-box access (to the adversary) and no additional nonuniformity. However, the construction only achieves *computational* functionality, that is, given a circuit produced by their construction, no efficient algorithm can find an input on which this circuit differs from the desired functionality. The work of Dodis et al. [6] also constructs obfuscators for proximity queries in this framework.

There is qualitative difference between the setting with high min-entropy considered in [5, 6] and that considered in this paper. Consider a predicate adversary A that checks whether the point function corresponds to an x whose first $n - 3\log n$ bits are 0, by checking whether the obfuscated circuit evaluates to 1 on n^3 inputs. To achieve virtual black-box in our setting, it is essential that the simulator queries the point function oracle on $\Omega(n^3)$ inputs, whereas in the relaxed setting with high min-entropy, the output of A can be trivially simulated because it is 0 except with negligible probability.

Obfuscation with random oracles. The obfuscator for point functions in [15] achieves the virtual black-box property in the *programmable* random oracle model, which is formalized as follows:

> for any nonuniform PPT A, there exists a nonuniform oracle PPT S_A and a negligible function $\epsilon(n)$ such that for all sufficiently large n and for all $x \in \{0,1\}^n$:
>
> $$\left| \Pr_{\mathcal{R}}\left[A^{\mathcal{R}}(\mathcal{O}^{\mathcal{R}}(I_x)) = 1\right] - \Pr\left[S_A^{I_x}(1^n) = 1\right] \right| \leq \epsilon(n)$$

In fact, their simulators produce views that are identically distributed to the views for general adversaries (over random oracles $\mathcal{R}$), bypassing all of the limitations described in Sec 1.2.3. Starting with their obfuscator for point functions, Lynn et al. [15] derived obfuscators for fairly complex access

control functionalities via composition. Unfortunately, the composition techniques presented therein do not apply to our construction.

2. PRELIMINARIES

2.1 Notation and definitions

We model efficient adversary strategies using nonuniform probabilistic polynomial-time machines (PPT), which are essentially the same as families of probabilistic polynomial-sized circuits. We say that a nonuniform PPT A has size s if the running time and the length of the nonuniform advice for A is bounded by s.

We write U_n to denote the uniform distribution over $\{0,1\}^n$, and $\mathrm{neg}(n)$ to denote a function of the form $n^{-\omega(1)}$. In the context of describing probability distributions, we write $x \in U_n$ to denote choosing x at random from U_n; we also use $x \in L$ to denote choosing an element x from the set L uniformly at random. For a probabilistic function f, we use $f(x;R)$ to denote the output of f on input x and internal coin tosses R, and we say that f is *public-coin* [14] if it publishes its internal coin tosses as part of its output.

Definition 1. A *point function* $I_x : \{0,1\}^n \to \{0,1\}$, is specified by a string $x \in \{0,1\}^n$ and on input $y \in \{0,1\}^n$, outputs 1 if $y = x$, and 0 otherwise. We would also use I_x to denote a circuit that hardwires the value x and computes I_x. The *family of point functions* is given by the collection $\{I_x\}_{x\in\{0,1\}^n, n\in\mathbb{N}}$.

2.2 Obfuscation

Definition 2. [1, 3] A probabilistic polynomial-time algorithm $\mathcal{O}$ is an *obfuscator for the family of circuits* $\mathcal{C} = \cup_n \mathcal{C}_n$ (where $\mathcal{C}_n$ is the subset of circuits in $\mathcal{C}$ that take inputs of length n) if the following three conditions hold:

- (approximate functionality) There exists a negligible function α such that for all n, for all $C \in \mathcal{C}_n$, with probability $1-\alpha(n)$ over the internal coin tosses of the obfuscator, $\mathcal{O}(C)$ describes a circuit that computes the same function as C.
- (polynomial slowdown) There is a polynomial p such that for every circuit $C \in \mathcal{C}$, $|\mathcal{O}(C)| \le p(|C|)$.
- (weak virtual black-box property) There is a polynomial q such that for any nonuniform PPT A of size s and any function $\epsilon(n) = 1/n^{O(1)}$, there exists a nonuniform PPT S_A of size $q(s, 1/\epsilon)$ such that for all sufficiently large n and for all circuits $C \in \mathcal{C}_n$:

$$\left| \Pr\left[A(\mathcal{O}(C)) = 1\right] - \Pr\left[S_A^C(1^{|C|}) = 1\right] \right| \le \epsilon(n)$$

The weak virtual black-box property is a relaxation of the virtual black-box property in [1] (in allowing the size of S_A to depend in ϵ), and in the case of point functions coincides with the definition of oracle simulatability in [3]. Obfuscation for general circuits is impossible even under this definition [1]. In addition, as pointed out in [1], this definition is equivalent to the one that requires, for every predicate P (not necessarily efficiently computable), the probability that $A_n(\mathcal{O}(C)) = P(C)$ be at most the probability that $S_n^C(1^{|C|}) = P(C)$ plus ϵ.

3. CONSTRUCTING OBFUSCATORS

3.1 The basic construction

We begin by describing the hash function with which we will instantiate the random oracle in the construction of [15].

CONSTRUCTION 3.1. *Let $\pi : \{0,1\}^n \to \{0,1\}^n$ be a permutation. We define a (public-coin) probabilistic function $h : \{0,1\}^n \times \{0,1\}^{3n^2} \to \{0,1\}^{3n^2+3n}$ as follows:*

$$h(x;\tau_1,\ldots,\tau_{3n}) = (\tau_1,\ldots,\tau_{3n}, \langle x,\tau_1\rangle, \langle \pi(x),\tau_2\rangle, \ldots, \langle \pi^{3n-1}(x),\tau_{3n}\rangle)$$

Remark 1. This construction (and the analysis that is to follow) closely resembles the pseudorandom generator of [2, 18, 11]. We emphasize that we do not append $\pi^{3n}(x)$ to the output of our hash function and that we use independent random strings for computing the dot product. The same construction was used in [5] as a building block for constructing pseudorandom function ensembles with certain collision-free properties.

To motivate the above construction and the analysis, we briefly review why instantiating the random oracle with several standard cryptographic primitives fails to yield an obfuscator. A one-way permutation would not work as it may reveal the first bit of x. Neither would using a pseudorandom generator with seed x as the output is only indistinguishable from uniform for a random x. Committing to the string x also fails for our purpose, as a commitment scheme guarantees that commitments to the strings 0^n and 1^n be indistinguishable, whereas functionality for an obfuscator requires that one be able to distinguish between the two.

THEOREM 3.2. *Under Assumption 1.1, there exists a (public-coin) obfuscator for the family of point functions.*

PROOF. We instantiate the random oracle in the construction of [15] with the hash function h from Construction 3.1 applied to the permutation π given by Assumption 1.1. That is, on input I_x and randomness R, the obfuscator $\mathcal{O}$ computes $(R,\rho) = h(x;R)$ and hard-wires (R,ρ) into the obfuscated circuit. On input y, the obfuscated circuit outputs 1 iff $h(y;R) = (R,\rho)$. Clearly, $\mathcal{O}$ satisfies polynomial slow-down and public-coin.

APPROXIMATE FUNCTIONALITY: Fix $x \neq y \in \{0,1\}^n$. Then,

$$\begin{aligned} & \Pr_{\tau_1,\ldots,\tau_{3n}}[h(x;\tau_1,\ldots,\tau_{3n}) = h(y;\tau_1,\ldots,\tau_{3n})] \\ = & \Pr_{\tau_1,\ldots,\tau_{3n}}[\forall\, j = 1,\ldots,3n : \langle \pi^{j-1}(x),\tau_j\rangle = \langle \pi^{j-1}(y),\tau_j\rangle] \\ = & \frac{1}{2^{3n}} \end{aligned}$$

because π is a permutation, so $\pi^{j-1}(x) \neq \pi^{j-1}(y)$ for all $j = 1, 2, \ldots, 3n$. Taking a union bound over all $x, y \in \{0,1\}^n$ shows that h is collision-free, from which approximate functionality follows[3].

[3]In fact, we achieve a stronger notion of approximate functionality, namely that there exists a negligible function α such that for all n, with probability $1-\alpha(n)$ over the internal coin tosses of the obfuscator, for all $C \in \mathcal{C}_n$ and for all $x \in \{0,1\}^n$, $(\mathcal{O}(C))(x) = C(x)$.

VIRTUAL BLACK-BOX: Let A be a predicate nonuniform PPT of size s. The only auxiliary information stored in the obfuscated circuit is the string $h(x;R)$, so it suffices to simulate the output of A on input $h(x;R)$ instead of $\mathcal{O}(I_x;R)$. We define

$$L_n = \{x \in \{0,1\}^n : \left|\Pr[A(h(x;U_{3n^2})) = 1] - \Pr[A(U_{3n^2+3n}) = 1]\right| \geq \epsilon\}$$

The set L_n specifies the inputs for which the hash value reveals "too much" information to the adversary, in that the adversary's output cannot be approximated by $A(U_{3n^2+3n})$.

CLAIM 3.3. *For all sufficiently large n,*

$$|L_n| \leq \text{poly}(n, s, 1/\epsilon)$$

Note that the virtual black-box property follows readily from Claim 3.3 as follows: consider the nonuniform oracle PPT S_A that has L_n hardwired into it. On input 1^n and oracle access to I_x, S_A queries I_x on each element of L_n. If $x \in L_n$, S_A outputs a random sample from the distribution $A(h(x;U_{3n^2}))$. Otherwise (that is, $x \notin L_n$), S_A outputs a random sample from the distribution $A(U_{3n^2+3n})$.

To prove Claim 3.3, we show how to construct using A a circuit of size $\text{poly}(n,1/\epsilon)s$ that inverts π on $\epsilon|L_n|/16n$ inputs. We may assume (by replacing A with its negation if necessary) that there exists a subset L'_n of L_n of size at least $|L_n|/2$ such that for all $x \in L'_n$:

$$\Pr[A(h(x;U_{3n^2})) = 1] - \Pr[A(U_{3n^2+3n}) = 1] \geq \epsilon$$

Averaging over $x \in L'_n$, we have

$$\Pr_{x \in L'_n}[A(h(x;U_{3n^2})) = 1] - \Pr[A(U_{3n^2+3n}) = 1] \geq \epsilon$$

Consider the following $3n+1$ hybrid distributions: where $x \in L'_n, \tau_1, \ldots, \tau_{3n} \in U_n$ and $b_1, \ldots, b_{3n} \in \{0,1\}$:

$$\begin{aligned}
D_0 &= \{\tau_1, \ldots, \tau_{3n}, \langle \pi^{3n-1}(x), \tau_{3n}\rangle, \ldots, \langle x, \tau_1\rangle\} \\
D_1 &= \{\tau_1, \ldots, \tau_{3n}, \langle \pi^{3n-1}(x), \tau_{3n}\rangle, \ldots, b_1\} \\
&\vdots \quad\quad \vdots \quad\quad \cdots \quad\quad \cdots \quad\quad \vdots \\
D_{3n} &= \{\tau_1, \ldots, \tau_{3n}, \quad b_{3n}, \quad \ldots, b_1\}
\end{aligned}$$

A standard hybrid argument yields a distinguisher for the distributions D_{j-1} and D_j for some $j \in \{1, 2, \ldots, 3n-1\}$ and a next-bit predictor P computing $\langle \pi^{j-1}(x), \tau_j\rangle$ with advantage $\epsilon/3n$. Upon hard-wiring the internal coin tosses of P and the choices of $b_1, \ldots, b_{j-1}$ and $\tau_{j'}$ for all $j' \neq j$, we obtain a deterministic PPT P' of size $O(s)$ satisfying:

$$\Pr_{x \in L'_n, \tau_j}[P'(\pi^j(x), \tau_j) = \langle \pi^{j-1}(x), \tau_j\rangle] \geq 1/2 + \epsilon/3n$$

By an averaging argument, it follows that for a $\epsilon/6n$ fraction of x in L'_n (and call such an x "good"):

$$\Pr_{\tau_j}[P'(\pi^j(x), \tau_j) = \langle \pi^{j-1}(x), \tau_j\rangle] \geq 1/2 + \epsilon/6n$$

We can then apply Goldreich-Levin [11] to recover "good" x's. Now, we have a probabilistic circuit of size $\text{poly}(n,1/\epsilon)s$ that recovers each "good" x with probability $3/4$. We can then hardwire the randomness into the circuit to obtain one of size $\text{poly}(n,1/\epsilon)s$ that inverts π on a $\epsilon/8n$ fraction of the values in $\pi^j(L'_n)$, that is, on at least $\epsilon|L_n|/16n$ different values. Assumption 1.1 yields $\epsilon|L_n|/16n \leq (\text{poly}(n,1/\epsilon)s)^c$, from which the upper bound on $|L_n|$ follows. □

3.2 An improved construction

The amount of randomness used by the obfuscator may be reduced from $3n^2$ to $O(n)$ via random walk on expanders. Let $\{G_n\}$ be a family of explicit d-regular expanders for some constant d with vertex set $\{0,1\}^n$ and normalized second largest eigenvalue bounded by $1/2$ [13]. A standard spectral argument (see for instance [9, Sec 2.6.2]) shows that these expanders satisfy a mixing property:

LEMMA 3.4. *Let M denote the normalized adjacency matrix of the expander G_n described above. Let P denote a projection onto any subset of $\{0,1\}^n$ of density $1/2$. Then, for any vector $\vec{z} \in \mathbb{R}^{2^n}$, $\|PM\vec{z}\| \leq \frac{\sqrt{3}}{2}\|\vec{z}\|$.*

We may then derandomize Construction 3.1 by sampling the strings $\tau_1, \ldots, \tau_{3n}$ for computing dot products using a random walk on G_n.

CONSTRUCTION 3.5. *Let $\pi : \{0,1\}^n \to \{0,1\}^n$ be a permutation and G_n be a d-regular graph. We define a probabilistic function $h : \{0,1\}^n \times \{0,1\}^{O(n)} \to \{0,1\}^{O(n)}$ as follows:*

$$h(x;\tau,\sigma_1,\ldots,\sigma_{10n}) = (\tau, \sigma_1, \ldots, \sigma_{10n}, \langle x, \tau\rangle, \langle \pi(x), G_n(\tau;\sigma_1)\rangle, \ldots, \langle \pi^{10n}(x), G_n(\tau;\sigma_1,\ldots,\sigma_{10n})\rangle)$$

where $\sigma_1, \ldots, \sigma_{10n} \in [d]$, and $G_n(\tau;\sigma_1,\ldots,\sigma_i)$ is the label on the vertex reached by a walk on G_n that starts at the vertex labelled τ and then following the sequences of i edges labelled $\sigma_1, \ldots, \sigma_i$.

Now, we describe how to modify the proof of Theorem 3.2 to obtain an obfuscator for point functions using this construction.

APPROXIMATE FUNCTIONALITY: This basically follows from the fact that random walks on expanders are rapidly mixing. To formalize this, fix $x \neq y \in \{0,1\}^n$. For $i = 1, 2, \ldots, 10n+1$, let P_i denote projection onto $\{r \in \{0,1\}^n \mid \langle \pi^{i-1}(x), r\rangle = \langle \pi^{i-1}(y), r\rangle\}$. Then,

$$\begin{aligned}
&\Pr_{\tau,\sigma_1,\ldots,\sigma_{10n}}[h(x;\tau,\sigma_1,\ldots,\sigma_{10n}) = h(y;\tau,\sigma_1,\ldots,\sigma_{10n})] \\
=\ & |P_{10n+1}MP_{10n}M\cdots P_2MP_1\vec{u}| \\
\leq\ & 2^{n/2}\|P_{10n+1}MP_{10n}M\cdots P_2MP_1\vec{u}\| \\
\leq\ & (\tfrac{\sqrt{3}}{2})^{10n}
\end{aligned}$$

where $\vec{u}$ is the vector corresponding to the uniform distribution over $\{0,1\}^n$. We can then take a union bound over all $x, y \in \{0,1\}^n$ as before.

VIRTUAL BLACK-BOX: Let A be a PPT of size s and we define a set L_n as before. In addition, we may consider an analogous collection of $10n+2$ hybrid distributions and derive a distinguisher for the distributions D_{j-1} and D_j, for some $j \in \{1, 2, \ldots, 10n+2\}$. Since $G_n(U_n;\sigma_1,\ldots,\sigma_{j-1}) = U_n$ for any fixed $\sigma_1, \ldots, \sigma_{j-1}$, we may write D_{j-1} and D_j as:

$$\begin{aligned}
D_{j-1} &= \{\tau_j, \sigma_1, \ldots, \langle \pi^j(x), G_n(\tau_j;\sigma_j)\rangle, \langle \pi^{j-1}(x), \tau_j\rangle \ldots, b_1\} \\
D_j &= \{\tau_j, \sigma_1, \ldots, \langle \pi^j(x), G_n(\tau_j;\sigma_j)\rangle, \quad b_j, \quad \ldots, b_1\}
\end{aligned}$$

As in the previous analysis, the distinguisher yields a next-bit predictor, and upon hard-wiring its internal coin tosses

and the choices of $\sigma_1, \ldots, \sigma_{10n}, b_1, \ldots, b_{j-1}$, we obtain a deterministic PPT predictor P' of size $O(s)$ satisfying:

$$\Pr_{x \in L'_n, \tau_j}[P'(\pi^j(x), \tau_j) = \langle \pi^{j-1}(x), \tau_j \rangle] \geq 1/2 + \epsilon/(10n+1)$$

The rest of the proof is as before.

Remark 2. In our framework for obfuscation, we may describe both the inputs and outputs of our obfuscation algorithm using programs with auxiliary information instead of circuits. In this case, our obfuscator for point functions based on Construction 3.5 produces programs whose description is linear in that of the input, which is optimal up to constant factors.

Next, we note that we may relax the assumption in Theorem 3.2 to requiring a strongly one-way permutation ensemble, instead of a specific permutation. Unlike the setting of standard one-way permutations, this does not immediately imply Assumption 1.1:

ASSUMPTION 3.6. *There exists a collection of polynomial-time computable permutation ensemble $\mathcal{F} = \{\mathcal{F}_n\}_{n \in \mathbb{N}}$ where each $f \in \mathcal{F}_n$ is a permutation on $\{0,1\}^n$ and a constant c such that for every polynomial $s = s(n)$ and every nonuniform PPT A of size s, there exists a negligible function α, such that for all sufficiently large n, with probability $1 - \alpha(n)$ over $f \in \mathcal{F}_n$,*

$$\Pr_{x \in U_n}[A(f, f(x)) = x] \leq s^c/2^n$$

3.3 Extension to multi-bit and prefix matching

Definition 3. A *point function with $m(n)$-bit output* $I_{x,\beta} : \{0,1\}^n \to \{0,1\}^{m(n)}$, is specified by a string $x \in \{0,1\}^n$ and another string $\beta \in \{0,1\}^{m(n)}$ and on input $y \in \{0,1\}^n$, outputs β if $y = x$, and $0^{m(n)}$ otherwise. We would also use $I_{x,\beta}$ to denote a circuit that hardwires the values x and β and computes $I_{x,\beta}$. The *family of point functions with $m(n)$-bit output* is given by the collection $\{I_{x,\beta}\}_{x \in \{0,1\}^n, \beta \in \{0,1\}^{m(n)}, n \in \mathbb{N}}$.

THEOREM 3.7. *Under Assumption 1.1, for any $m(n) = O(\log n)$, there exists a (public-coin) obfuscator for the family of point functions with $m(n)$-bit output.*

PROOF (SKETCH). Again, we use the obfuscator for point functions with multi-bit output in [15], except we replace the random oracle with a generalization of the hash function from Construction 3.1, $h_{n,m} : \{0,1\}^n \times \{0,1\}^{(m+3n)n} \to \{0,1\}^{(m+3n)(n+1)}$ defined as follows:

$$h(x; \tau_1, \ldots, \tau_{3n+m}) = (\tau_1, \ldots, \tau_{3n+m}, \langle x, \tau_1 \rangle, \ldots, \langle \pi^{3n+m-1}, \tau_{3n+m} \rangle)$$

Specifically, on input I_x and randomness R, the obfuscator $\mathcal{O}$ computes $(R, \rho_x, \rho'_x) = h(x; R)$ where $|\rho_x| = 3n$ and $|\rho'_x| = m$, and hard-wires (R, ρ_x, z) into the obfuscated circuit, where $z = \rho'_x \oplus \beta$. On input y, the obfuscated circuit computes $(R, \rho_y, \rho'_y) = h(y; R)$ outputs $z \oplus \rho'_y$ if $\rho_x = \rho_y$ and 0^m otherwise. We may also derandomize $h_{n,m}$ using random walk on expanders to obtain a construction that uses only $O(m+n)$ random bits. □

Definition 4. A *$t(n)$-bit prefix function* $\text{pre}_x : \{0,1\}^n \to \{0,1\}$, is specified by a string $x \in \{0,1\}^{t(n)}$ and on input $y \in \{0,1\}^n$, outputs 1 if the first $t(n)$ bits of y matches x, and 0 otherwise. We would also use pre_x to denote a circuit (padded with 1^n) that hardwires the values x and computes pre_x. The *family of $t(n)$-bit prefix functions* is given by the collection $\{\text{pre}_x\}_{x \in \{0,1\}^{t(n)}, n \in \mathbb{N}}$.

In order to obfuscate prefix functions, we require a stronger variant of Assumption 1.1:

ASSUMPTION 3.8. *There exists a polynomial-time computable permutation $\pi : \{0,1\}^n \to \{0,1\}^n$ and a constant c such that for every $s(n) = 2^{o(n)}$ and every nonuniform PPT A of size $s(n)$: for all sufficiently large n,*

$$\Pr_{x \in U_n}[A(\pi(x)) = x] \leq s^c/2^n$$

THEOREM 3.9. *Under Assumption 3.8, for any "nice" function $t(n)$, there exists a (public-coin) obfuscator for the family of $t(n)$-bit prefix functions.*

PROOF (SKETCH). If $t(n) = O(\log n)$, then the family of $t(n)$-bit prefix functions is efficiently and exactly learnable using membership queries and can therefore be trivially obfuscated as pointed out in [15] (see Prop 5.2). If $t(n) = \omega(\log n)$, the construction is the same as that for point functions, except we use Construction 3.5 (or Construction 3.1) with input length $t(n)$. Specifically, on input pre_x (with $x \in \{0,1\}^{t(n)}$) and randomness R, the obfuscator $\mathcal{O}$ computes $(R, \rho) = h(x; R)$ and hard-wires (R, ρ) into the obfuscated circuit. On input $y \in \{0,1\}^n$, the obfuscated circuit outputs 1 iff $h(y_0; R) = (R, \rho)$, where y_0 is the $t(n)$-bit prefix of y. Note that we require a stronger assumption because a nonuniform PPT adversary here runs in time super-polynomial in $m(n)$, which is the input length of the permutation π we use in this construction. If we introduce a security parameter into the definition of virtual black-box, then we could derive the analysis directly from Theorem 3.2. □

4. COMPUTATIONAL ASSUMPTIONS

4.1 On inverting a random permutation

We extend the argument of [8] to show that a random permutation is strongly one-way in the sense of Assumption 1.1. Clearly, our analysis is tight up to polynomial factors. To simplify the exposition, we include the input gates in determining the size of a circuit, so that any circuit has size at least that of its input. We use Π_n to denote the set of all permutations over $\{0,1\}^n$.

THEOREM 4.1. *For all sufficiently large n, with probability at least $1 - 2^{-n^3}$ over a random $\pi \in \Pi_n$: for all s with $n \leq s \leq 2^{n/5}$ and for all oracle circuits A of size s,*

$$\Pr_{x \in U_n}[A^\pi(\pi(x)) = x] \leq s^4/2^n$$

The key insight is the same as that in [8]: any permutation π for which there is an oracle circuit A such that A inverts π on "many" inputs has a "short" description (given A). Hence there cannot be too many such permutations.

CLAIM 4.2. *Let A be an oracle circuit that makes (at most) $s-1$ queries to a permutation $\pi : \{0,1\}^n \to \{0,1\}^n$, and for which $\Pr_y[A^\pi(y) = \pi^{-1}(y)] \geq s^4/2^n$. Then, π can be described using at most*

$$\log \binom{2^n}{s^3} + s^3 \log s + \log[2^n]_{s^4-s^3} + \log(2^n - s^4)!$$

bits given A (where $[n]_k$ denotes the quantity $n(n-1)\cdots(n-k+1)$).

PROOF. Let $N = 2^n$. WLOG, assume that A makes distinct queries to π, and always queries π on the value it is going to output. Consider the set I of s^4 points on which A inverts π, after making s queries to π. We define a set Y and another set W via the following process: initially W is $\{0,1\}^n$ whereas Y is empty, and all the elements of I are candidates for inclusion in Y. Take the lexicographically first element y from I, and place it in Y. Next, simulate the computation of $A^\pi(y)$ and let $x_1, \ldots, x_{q(y)}$ be the distinct queries made by A to π, with $q(y) \leq s$ and $\pi(x_{q(y)}) = y$. Let $y_1, \ldots, y_{q(y)}$ be the corresponding answers (that is, $y_i = \pi(x_i)$). We add the answers $y_1, \ldots, y_{q(y)-1}$ (in order) and the number $q(y)$ to our description of π, and remove the strings $x_1, \ldots, x_{q(y)}$ from I. In addition, we remove the strings $x_1, \ldots, x_{q(y)}$ from W. At any step of the construction, one element is added to Y and at most $s-1$ elements is removed from I. Since I initially contains s^4 elements, in the end, we have $|Y| \geq s^3$. In fact, we will stop the process when Y reaches exactly s^3 elements.

We claim that given the set Y, the values $y_1, \ldots, y_{q(y)-1}$ and the number $q(y)$ for each $y \in Y$, the values of π on W and the circuit A, it is possible to compute π everywhere. The values of π^{-1} on Y can be reconstructed sequentially for all $y \in Y$, taken in lexicographic order, as follows: Simulate the computation of $A^\pi(y)$. Our description allows us to answer the first $q(y)-1$ queries, and the $q(y)$th query is exactly $\pi^{-1}(y)$. Note that while reconstructing π^{-1} on Y, we have also reconstructed the set W and π on all of $\{0,1\}^n - W$.

Let $Q = \sum_{y \in Y} q(y)$, so $Q \leq s^3(s-1)$ and $|W| \leq 2^n - Q - s^3$. Describing Y requires $\log \binom{N}{s^3}$ bits. The descriptions $y_1, \ldots, y_{q(y)-1}$ and the number $q(y)$ for all $y \in Y$ require at most $s^3 \log s + \sum_{i=1}^{Q} \log(N-i+1) = s^3 \log s + \log[N]_Q$. Once we have constructed the set W, π on $\{0,1\}^n - W$ can be described using $\log(N-|W|)!$ bits. The total description requires at most $\log \binom{N}{s^3} + s^3 \log s + \log[N]_Q + \log(N-|W|)! \leq \log \binom{N}{s^3} + s^3 \log s + \log[2^n]_{s^4-s^3} + \log(N-s^4)!$ bits. □

Now that we have established Claim 4.2, we may complete the proof of Theorem 4.1 as follows.

PROOF (OF THEOREM 4.1). Fix an oracle circuit A of size s, where $s \leq 2^{n/5}$. It follows from the above claim that the fraction of permutations $\pi \in \Pi_n$ such that $\Pr_x[A^\pi(\pi(x)) = x] \geq s^4/2^n$ is at most

$$\binom{N}{s^3} \cdot s^{s^3} \cdot [N]_{s^4-s^3} \cdot (N-s^4)! \cdot \frac{1}{N!} = \frac{s^{s^3}}{s^3!} \cdot \frac{[N]_{s^4-s^3}}{[N-s^3]_{s^4-s^3}}$$

which is upper bounded by

$$\left(\frac{es}{s^3}\right)^{s^3} \cdot \exp\left(\frac{s^3}{N-s^3} + \ldots + \frac{s^3}{N-s^4}\right) < \left(\frac{e^2}{s^2}\right)^{s^3}$$

Since there are at most $2^{sn \log s}$ oracle circuits of size s, a union bound shows that the probability over a random choice of $\pi \in \Pi_n$ that there exists an oracle circuit A of size s, such that $n \leq s \leq 2^{n/5}$ and $\Pr_x[A^\pi(\pi(x)) = x] \geq s^4/2^n$, is at most

$$\sum_{s=n}^{2^{n/5}} s^{ns} \left(\frac{e^2}{s^2}\right)^{s^3} < 2^{n/5} \left(\frac{e^2}{n}\right)^{n^3} < 2^{-n^3} \quad \square$$

Remark 3. The improvement over [8] is in the length of the description in Claim 4.2: the analysis in [8] yields $2\log\binom{2^n}{s^3} + \log(2^n - s^3)! \approx 2^n n + s^3 n - \Theta(s^3 \log n)$ bits whereas our analysis yields approximately $2^n n - \Omega(s^3 \log s)$ bits. More generally, the analysis in [8] cannot provide a lower bound better than $2^{n/2}$ for "work" whereas our analysis yields a lower bound of $2^{n-\Theta(\log n)}$.

Consider a relativization of Definition 2 wherein all parties (the obfuscation algorithm, the adversary and the simulator) have access to a random permutation oracle, and the simulator must simulate the adversary's (single-bit) output with respect to the same oracle. This is a stronger requirement than that for the random oracle model, since the adversary and the simulator are chosen *after* the oracle is fixed. We refer to this as the non-programmable random permutation oracle model [16]:

> (virtual black-box property with a non-programmable random permutation oracle) With probability $1 - \mathrm{neg}(n)$ over $\pi \in \Pi_n$, for every nonuniform PPT A and every function $\epsilon(n) = 1/n^{O(1)}$, there exists a nonuniform oracle PPT S_A of size $\mathrm{poly}(n, 1/\epsilon)$ such that for all sufficiently large n, for all circuits $C \in \mathcal{C}_n$:
>
> $$\left| \Pr\left[A^\pi(\mathcal{O}^\pi(C)) = 1\right] - \Pr\left[S_A^{C,\pi}(1^{|C|}) = 1\right] \right| \leq \epsilon(n)$$

COROLLARY 4.3. *There exists a (public-coin) obfuscator for point functions in the non-programmable random permutation oracle model.*

4.2 Necessity of strong assumptions

In this section, we explore the necessity of Assumptions 1.1 and 3.6.

PROPOSITION 4.4. *Suppose that public-coin obfuscators exist. Then, there exists an efficiently computable function ensemble $\mathcal{F} = \{\mathcal{F}_n\}$ satisfying the following properties:*

- *(weakly collision-free) For all $x \neq y$, $\Pr_{f \in \mathcal{F}_n}[f(x) \neq f(y)] \geq 1 - \mathrm{neg}(n)$.*
- *(somewhat strongly one-way) For all polynomials $p(n)$, for all PPT A of size $p(n)$, there exists a constant $c > 0$ such that for all sufficiently large n, $|Q_n| < n^c$, where Q_n is the set:*

  $$\{x \in \{0,1\}^n \mid \Pr_{f \in \mathcal{F}_n}[A_n(f, f(x)) \in f^{-1}(f(x))] \geq 1/p(n)\}$$

Note that the main qualitative differences between the function ensemble herein and that stipulated in Assumption 3.6 are weakly collision-free functions vs permutations and a reversal of quantifiers between x and f in the specification of the one-wayness property.

PROOF. Let $\mathcal{O}$ be a public-coin obfuscator for point functions using $r(\cdot)$ random bits. We define[4]

$$\mathcal{F}_n = \{f \mid (f(x); f) = \mathcal{O}(I_x; R), R \in \{0,1\}^{r(n)}\}$$

that is, every $R \in \{0,1\}^{r(n)}$ is the index[5] of a function $f \in \mathcal{F}_n$, and $f(x)$ is the string $\mathcal{O}(I_x; R)$, excluding R (which is part of $\mathcal{O}(I_x; R)$ since $\mathcal{O}$ is public-coin). It is clear that functionality for $\mathcal{O}$ implies $\mathcal{F}$ is weakly collision-free.

Next, for each n, let x_n denote the lexicographic mid-point of Q_n, and let P_n denote the predicate "$\succ x_n$". Consider the predicate PPT B, which on input (f, y)

i. Compute $x' = A(f, y)$.

ii. Feed x' as input to the obfuscated circuit $\mathcal{O}(I_x) = (f(x); f)$. If the output is 1, output $P_n(x')$; otherwise, output a random bit.

It is easy to see that for all $x \in Q_n$:

$$\Pr_{f \in \mathcal{F}_n}[B(f, f(x)) = P_n(x)] \geq \tfrac{1}{2} + \tfrac{1}{2p(n)} - \mathrm{neg}(n)$$

where the $\mathrm{neg}(n)$ term captures the probability that $\mathcal{O}(I_x)$ fails to compute I_x. Let S be the simulator for B with distinguishing probability $1/4p(n)$ given by the virtual black-box property of $\mathcal{O}$. Therefore, for all sufficiently large n, for all $x \in Q_n$, we have:

$$\Big| \Pr_{f \in \mathcal{F}_n}\big[B(f, f(x)) = P_n(x)\big] - \Pr\big[S^{I_x}(1^n) = P_n(x)\big] \Big| \leq \tfrac{1}{4p(n)}$$

On the other hand, since S makes at most $\mathrm{poly}(n)$ queries into I_x, we have:

$$\Pr_{x \in Q_n}[S^{I_x}(1^n) = P_n(x)] \leq \tfrac{1}{2} + \tfrac{\mathrm{poly}(n)}{|Q_n|}$$

By combining the 3 inequalities, we obtain the polynomial bound on $|Q_n|$. □

Next, we consider a hardness result for CIRCUIT-SAT that is implied by the existence of obfuscators for point functions, not necessarily public-coin ones; it shows that some sort of exponential lower bound is indeed necessary to obtain any kind of positive results for obfuscating point functions. This was pointed out to us by Luca Trevisan (private communication, 2004):

PROPOSITION 4.5. *If there exists a nontrivial (i.e., a probabilistic* $2^{o(\#\mathsf{variables})} \cdot \mathrm{poly}(\mathsf{circuitsize})$*-time) algorithm for the* CIRCUIT-SAT *problem, then obfuscating point functions is impossible.*

PROOF. Let $t(\#\mathsf{variables}, \mathsf{circuitsize})$ denote the running time of the CIRCUIT-SAT algorithm, and suppose on the contrary that there exists an obfuscator $\mathcal{O}$ for point functions. Let $\ell(n) = \omega(\log n)$ such that $t(\ell(n), \mathrm{poly}(n)) = \mathrm{poly}(n)$, and let L denote the set of strings $\{0,1\}^n$ whose last $n - \ell(n)$ bits are 0's (so that $|L| = n^{\omega(1)}$). Let A denote a PPT that uses the CIRCUIT-SAT algorithm to decide on input a circuit C of size $\mathrm{poly}(n)$ on $\ell(n)$ variables, whether there is a satisfying assignment for C whose first bit is 1. Then,

$$\Pr_{x \in L}[A(\mathcal{O}(I_x)) = x_1] \geq 1 - \mathrm{neg}(n)$$

(where x_1 denotes the first bit of x) whereas for any nonuniform oracle PPT S_A,

$$\Pr_{x \in L}[S_A^{I_x}(1^n) = x_1] \leq 1/2 + \mathrm{neg}(n)$$

This yields the required contradiction to the virtual black-box property. □

It is easy to show that if there is a nontrivial algorithm for CIRCUIT-SAT, then Assumptions 1.1 and 3.6 do not hold. Together with Theorem 3.2 and Prop 4.4, this yields a fairly good picture of the computational complexity of obfuscating point functions.

5. IMPOSSIBILITY RESULTS

5.1 Strongly black-box simulators

We present a formalization of black-box simulation for virtual black-box similar to that for zero-knowledge [10], wherein there is a universal oracle simulator that accesses the adversary as a black-box. Unlike the setting for zero-knowledge wherein the black-box simulator only accesses the adversary as an oracle, the black-box simulator in our setting accesses two different oracles: the circuit and the adversary.

> (virtual black-box property with a strongly black-box simulator) For every function $\epsilon(n) = 1/n^{O(1)}$, there exists an oracle PPT S such that for nonuniform PPT A, we have: for all sufficiently large n, for all circuits $C \in \mathcal{C}_n$
>
> $$\Big|\Pr\big[A(\mathcal{O}(C)) = 1\big] - \Pr\big[S^{A,C}(1^{|C|}) = 1\big]\Big| \leq \epsilon(n)$$

As in the case for zero-knowledge, we do not allow the size of the black-box simulator S to depend on that for A; this means that the number of oracle queries made by the simulator to both the circuit and the adversary is independent of the size of the adversary. This property is satisfied by the simulator of [5, 6]. In our proof, we only require that the number of queries made to the circuit be independent of the size of the adversary. Ideally, we would like to rule out black-box simulators wherein the number of queries to the circuit is allowed to depend on the size of the adversary.

PROPOSITION 5.1. *Obfuscators for point functions with strongly black-box simulators do not exist.*

PROOF. Let $t = t(n)$ denote an upper bound on the size of the black-box simulator S (for $\epsilon = 1/8$ say), that is, S makes at most t queries to the oracle for the point function. Let L, L'_n be 2 disjoint subsets of $\{0,1\}^n$ with $|L| = |L'_n| = 2t$. Let $\chi_L : \{0,1\}^n \to \{0,1\}$ be the indicator function for L, that is, $\chi_L(x) = 1$ iff $x \in L$, and let A be nonuniform PPT that on input $\mathcal{O}(I_x)$ computes $\chi_L(x)$ by evaluating the obfuscated circuit on every point in L. By approximate functionality, we have

$$\Pr_{x \in L \cup L'_n}[A(\mathcal{O}(I_x)) = \chi_L(x)] \geq 1 - \mathrm{neg}(n)$$

On the other hand, for any S which makes at most t queries to I_x (even if S "knows" A, L and L'_n), we have

$$\Pr_{x \in L \cup L'_n}[S^{A, I_x}(1^n) = \chi_L(x)] \leq 3/4$$

This contradicts the virtual black-box property. □

[4]Note that we require that $\mathcal{O}$ be public-coin in order that in the definition of "somewhat strongly one-way", we can give A_n the ability to compute f.

[5]To avoid introducing additional notation, we use f to denote both the function and its index.

5.2 Obfuscation against general adversaries

As pointed out in [1], we may want to consider a stronger notion of virtual black-box, where we do not restrict the nature of what the adversary is trying to compute given the obfuscated circuit. Informally, we require that the simulator, given just oracle access to a circuit C, produce an output distribution that is computationally indistinguishable from what the adversary computes when given $\mathcal{O}(C)$. In this setting, it suffices to consider adversary that computes the identity function, that is, output $\mathcal{O}(C)$.

> (virtual black-box property against a general adversary) For every polynomial $s(n)$ and every function $\epsilon(n) = 1/n^{O(1)}$, there exists a nonuniform oracle PPT S of size $\text{poly}(s, n, 1/\epsilon)$ such that for all nonuniform PPT A of size s, for all sufficiently large n and for all circuits $C \in \mathcal{C}_n$:
>
> $$\Pr[A(\mathcal{O}(C)) = 1] - \Pr[A(S^C(1^{|C|})) = 1] \,| \leq \epsilon$$

Note that in this definition, we allow the size of the simulator S to depend on the size of the distinguisher. The difference between this definition and that with a weak simulator presented in Section 2.2 is the order of quantifiers (so that the simulator S_n may or may not depend on the distinguisher A_n): informally, upon fixing n, s, ϵ,

- (weak simulator) for all nonuniform PPT A of size s, there exists a nonuniform oracle PPT S of size $\text{poly}(s, n, 1/\epsilon)$ such that for all $C \in \mathcal{C}_n$: $|\Pr[A(\mathcal{O}(C)) = 1] - \Pr[S^C(1^{|C|}) = 1]| \leq \epsilon$.
- (general adversary) there exists a nonuniform oracle PPT S of size $\text{poly}(s, n, 1/\epsilon)$ such that for all nonuniform PPT A of size s, for all $C \in \mathcal{C}_n$: $|\Pr[A(\mathcal{O}(C)) = 1] - \Pr[A(S^C(1^{|C|})) = 1]| \leq \epsilon$.

PROPOSITION 5.2. *A family of circuits $\mathcal{C} = \cup_n \mathcal{C}_n$ is obfuscatable against general adversaries iff $\mathcal{C}$ is (nonuniformly) efficiently and exactly learnable using membership queries.*

PROOF. Obfuscating exactly learnable functions against general adversaries is straight-forward and was observed in [15]: the obfuscator simply takes the input circuit C and outputs the circuit produced by the learning algorithm given oracle access to C; the simulator does essentially the same thing and thus its size will not in fact depend on the size of the distinguisher, even though we allow for that in the definition.

The learning algorithm for an efficiently obfuscatable against general adversaries merely outputs a circuit computing the majority of n^2 independent copies of the simulator. To see why this works, let $s(n)$ denote an upper bound of the size of the circuits produced by the obfuscator, and S the simulator for distinguishers of size s and $\epsilon = 1/4$. For each $y \in \{0,1\}^n$, consider the nonuniform distinguisher eval_y of size s that evaluates its input (which is a circuit) on y. Then, we have: for all $C \in \mathcal{C}_n$ and for all $y \in \{0,1\}^n$:

$$\Pr[\mathsf{eval}_y(S^C(1^n)) = (\mathcal{O}(C))(y) \text{ and } (\mathcal{O}(C))(y) = C(y)] \geq 3/4 - \text{neg}(n)$$

Taking the majority of n^2 independent evaluations of S allows us to take union over all $y \in \{0,1\}^n$ so that for all $C \in \mathcal{C}_n$, with overwhelming probability, the learning algorithm given oracle access to C, produces a circuit that agrees with C everywhere on $\{0,1\}^n$. □

Note that the learning algorithm will be nonuniform if the simulator is nonuniform; however, the nonuniformity only depends on the input length n. Most lower bounds for learnability are based on lower bounds on query complexity or on cryptographic assumptions and may therefore be used to rule out nonuniform learning algorithms and thus obfuscation against general adversaries. In fact, the result relativizes in the sense that if we allow the obfuscation algorithm, the obfuscated circuit, the simulator and the distinguisher access to some oracle, we obtain an efficient and exact learning algorithm with respect to the same oracle.

The next result follows from the fact that point functions are not exactly learnable (since a uniformly chosen point function is statistically indistinguishable from the all-zeroes function given a polynomial number of membership queries).

THEOREM 5.3. *Obfuscating point functions against general adversaries is impossible. Furthermore, the proof relativizes and hence the result extends to the non-programmable random oracle model.*

This does not contradict the obfuscator for point functions in [15] achieving virtual black-box property with a general adversary because their simulator is allowed to program the random oracle.

6. CONCLUSION

We point out several interesting directions for future work:

- Is the dependency on the distinguishing probability in the size of the simulator necessary?
- Obfuscating multi-point functions: a natural extension of function points are multi-point functions $\{I_{x_1,\ldots,x_k} : \{0,1\}^n \to \{0,1\}\}$, for $(x_1, \ldots, x_k) \in (\{0,1\}^n)^k$ where $I_{x_1,\ldots,x_k}(y) = I_{x_1}(y) \vee \ldots \vee I_{x_k}(y)$. Also, obfuscating point functions with $\text{poly}(n)$ multi-bit output.
- Obfuscating AC_0: obfuscating TC_0 is impossible [1], whereas we can obfuscate NC_0 against general adversaries (since NC_0 is exactly learnable). Note that the techniques of [1] do not extend to AC_0 since pseudorandom functions do not exist in AC_0 [12]. In addition, AC_0 can implement point functions, so the results of Section 4 do apply in this setting.
- Obfuscating point functions with a-priori information, as discussed in [3].
- Explore the possibility of realizing other cryptographic constructions proven secure in the random oracle model under assumptions similar to Assumption 1.1.

7. ACKNOWLEDGEMENTS

I am especially grateful to David Molnar and David Wagner for introducing me to the problem, and to Luca Trevisan, Salil Vadhan, and an anonymous referee for insightful discussions on the subject and for constructive feedback that greatly improved the presentation and focus of this work. I would also like to thank many friends and

colleagues, in particular Andrej Bogdanov, Ran Canetti, Yevgeniy Dodis, Amit Sahai, Jason Waddle, Emanuele Viola, Ke Yang and the audience at the Feb 9 MIT CIS Seminar for encouragement, valuable feedback and helpful discussions.

8. REFERENCES

[1] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. Vadhan, and K. Yang. On the (im)possibility of obfuscating programs. In *Proc. Crypto '01*, 2001.

[2] M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM Journal of Computing*, 13(4):850–864, 1984.

[3] R. Canetti. Towards realizing random oracles: Hash functions that hide all partial information. In *Proc. Crypto '97*, 1997.

[4] R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited. In *Proc. 30th STOC*, 1998.

[5] R. Canetti, D. Micciancio, and O. Reingold. Perfectly one-way probabilistic hash functions. In *Proc. 30th STOC*, 1998.

[6] Y. Dodis and A. Smith. Correcting errors without leaking partial information. In *these proceedings*, 2005.

[7] C. Dwork, M. Naor, and A. Sahai. Concurrent zero-knowledge. In *Proc. 30th STOC*, 1998.

[8] R. Gennaro and L. Trevisan. Lower bounds on efficiency of generic cryptographic constructions. In *Proc. 41st FOCS*, 2000.

[9] O. Goldreich. *Foundations of Cryptography: Basic Tools*. Cambridge University Press, 2001.

[10] O. Goldreich and H. Krawczyk. On the composition of zero-knowledge proof systems. *SIAM Journal on Computing*, 25(1):169–192, 1996.

[11] O. Goldreich and L. Levin. Hard-core predicates for any one-way function. In *Proc. 21st STOC*, 1989.

[12] N. Linial, Y. Mansour, and N. Nissan. Constant depth circuits, fourier transform, and learnability. *Journal of the ACM*, 40(3):607–620, 1993.

[13] A. Lubotzky, R. Philips, and P. Sarnak. Ramanujan graphs. *Combinatorica*, 8(3):261–277, 1988.

[14] M. Luby. *Pseudorandomness and Cryptographic Applications*. Princeton University Press, 1996.

[15] B. Lynn, M. Prabhakaran, and A. Sahai. Positive results and techniques for obfuscation. In *Proc. Eurocrypt '04*, 2004.

[16] J. B. Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In *Proc. Crypto '02*, 2002.

[17] D. Wagner and I. Goldberg. Proofs of security for the unix password hashing algorithm. In *Proc. Asiacrypt '00*, 2000.

[18] A. Yao. Theory and applications of trapdoor functions. In *Proc. 23rd FOCS*, 1982.