Obfuscating Point Functions with Multibit Output *

Ran Canetti[1] and Ronny Ramzi Dakdouk[2]

[1] IBM T. J. Watson Research Center, Hawthorne, NY.
canetti@csail.mit.edu
[2] Yale University, New Haven, CT.
dakdouk@cs.yale.edu

Abstract. We construct obfuscators of point functions with multibit output and other related functions. A point function with multibit output returns a fixed string on a single input point and zero everywhere else. Obfuscation of such functions has a useful application as a strong form of symmetric encryption which guarantees security even when the key has very low entropy: Essentially, learning information about the plaintext is paramount to finding the key via exhaustive search on the key space.
Although the constructions appear to be simple and modular, their analysis turns out to be quite intricate. In particular, we uncover some weaknesses in the current definitions of obfuscation. One weakness is that current definitions do not guarantee security even under very weak forms of composition. We thus define a notion of obfuscation that is preserved under an appropriate composition operation. The constructions can use any obfuscator of point functions under the proposed definition. Alternatively, they can use perfect one way (POW) functions with statistical indistinguishability, or with computational indistinguishability at the price of somewhat weaker security.

Keywords. *obfuscation, composable obfuscation, multibit point function obfuscation, digital locker, point function obfuscation.*

1 Introduction

Program Obfuscation is one of the most intriguing open problems in cryptography. Informally, a program obfuscator (or, simply, an obfuscator) is a compiler that converts a program into another one, called the obfuscated program or code, that has a similar functionality but satisfies certain secrecy requirements. Informally, the secrecy requirement stipulates that whatever "useful" information the obfuscated code reveals is learnable from the program's input/output behavior. In other words, an obfuscated program should not reveal anything useful beyond what's learned by inspecting the program's outputs on inputs of choice. This requirement is formalized by Barak *et al.* [2] through a simulation-based definition called the virtual-blackbox property. The virtual-blackbox property says that every adversary has a corresponding simulator that emulates the output of the adversary given only oracle (i.e., blackbox) access to the same functionality being obfuscated.

* Work supported by NSF grants 0331548 and CFF-0635297, and BSF grant 2006317.

In the same work, Barak *et al.* provide impossibility results regarding general obfuscation, even when the output of the adversary is restricted to predicates. In other words, it is shown that there are certain functionalities and corresponding predicates where these predicates are learnable from any program implementing the functionalities but not so given blackbox access to them. In light of this general negative result, we are forced to study obfuscation of restricted classes of functions if we wish to adopt the definition of [2]. Here, we follow this line of work. In particular, we build on the previous work on point function obfuscation [4, 5, 13, 11] towards obfuscating slightly more complex functions, namely point functions with multibit output. Moreover, we show that obfuscation of point functions are not necessarily secure even under self-composition, a property needed in our analysis. We next go into a more detailed exposition of our work.

Obfuscation of point functions with multibit output. A point function returns 1 on a single input and 0 everywhere else. Formally, $F_x(y) = 1$ if $y = x$ and 0 otherwise. A point function with multibit output generalizes point functions in that it outputs, on a single input, a long string instead of 1. Formally, $F_{x,y}(z) = y$ if $z = x$, and 0 otherwise.

The connection to symmetric encryption. Obfuscators for point functions with multibit output have a useful application as what we call a **digital locker**. A digital locker is a strong form of symmetric encryption which provides meaningful security even the key is taken from a distribution with very low entropy. More specifically, the guarantee is that the complexity of learning anything about the plaintext corresponds to that of finding the key via exhaustive search over the key space. We formalize this privacy notion using the simulation paradigm in a way similar to obfuscation. Namely, we require that the behavior of the adversary on an encryption of message m with key k be simulatable given blackbox access to the multibit point function, $F_{k,m}$. Consequently, obfuscation of point functions with multibit output can be used to realize digital lockers as follows: to encrypt a message m using a key k, simply output an obfuscation of $F_{k,m}$.

Real life applications of digital lockers include password-based encryption where the human-generated password is far from uniform. For instance, Firefox has a password manager that acts as a digital locker [1]. The password manager locks website credentials using a master password chosen by the user. Then, the user has to provide this password in order to unlock the content. It is stressed that the goal here is not to prevent exhaustive search over the keys, but rather to guarantee that this is essentially the only possible attack.

The construction. Even though obfuscation of point functions with multibit output is known in the Random Oracle Model [11], it is not known in the standard model except when the function is drawn from a uniform distribution (specifically, when x in $F_{x,y}$ is uniform) [7] or when the output length of the function is short (specifically, when $|y| = O(log|x|)$) [13]. Here, we provide a transformation from point function obfuscators to obfuscators of point functions with multibit output. The idea is simple. The obfuscation of multibit point functions consists of some number of copies of obfuscated point functions. These copies have the property that the first and the ith copy correspond to an obfuscation of the same point function if and only if the ith bit in the multibit output is 1. In more detail, let $F_{a,b}$ be the multibit point function to be obfuscated, $t = |b|$, and $O(F_a, r)$ be the obfuscation of the point function, F_a, using randomness r. Then, the obfuscation of $F_{a,b}$ consists of $O(F_a, r_0), O(x_1, r_1), ..., O(x_t, r_t)$, where x_i is F_a if $b_i = 1$ and x_i is a uniformly chosen point function otherwise. To recover b from the correct a and this obfuscation, first verify that $O(F_a, r_0)(a) = 1$, then $b = O(x_1, r_1)(a), ..., O(x_t, r_t)(a)$.

On composing obfuscation. The construction described above is very simple and modular, and one expects that its proof be likewise. However, it turns out that this is not the case. To prove the security of the above transformation, we face an issue. Observe that our construction is composed of a concatenation of $t + 1$ obfuscated point functions. Thus, in order for our construction to be secure, the original obfuscation *has* to remain secure under composition. However, we show that the current definition of obfuscation does not guarantee composition. This is also the case even for composing multiple obfuscated copies of the *same* function. Interestingly, the statement still holds even if we consider obfuscation secure in the presence of auxiliary information. We emphasize that this is a fundamental point about the definition of obfuscation that is of independent interest.

In more detail, we show that there exists an obfuscation of point functions that reveals the input when it is self-composed. Specifically, we show an obfuscator, O, such that for any x, it is possible to recover x from $O(F_x, r_1), ..., O(F_x, r_{nlog(n)})$, where $n = |x|$.

Moreover, similar results holds for POW functions and POW functions secure with auxiliary information [4, 5]. At a high level, a POW function can be thought of as an obfuscation of point function. See Appendix A for more details on POW functions and their relation to point function obfuscation.

In light of these negative results, we analyze the above construction using, as the underlying primitive, three different forms of composable obfuscation of point functions. First, if the underlying primitive is a composable obfuscation of point functions (as in simply-composable obfuscation of [11]), then this construction is a composable obfuscation of multibit point functions. This is actually a characterization: composable obfuscation of point functions exists if and only if that of point functions with multibit output exists. Second, we show that our construction is an obfuscation of multibit point functions if the underlying primitive is a statistically indistinguishable POW function.[1] Third, if the primitive is a computationally indistinguishable POW function, then the construction is an obfuscation provided that y in $F_{x,y}$, is "independent" of x.

Finally, we show how to generalize this construction to obfuscate set-membership predicates and functions for polynomial-sized sets. A set-membership predicate outputs 1 if the input belongs to the set and 0 otherwise, while a set-membership function outputs a string, y_i, if the input matches a set member, x_i, and 0 otherwise.

A tighter definition of obfuscation. The standard definition of obfuscation incorporates an unspecified "polynomial slack", in the sense that it allows the simulator to query its

[1] To be accurate, the second construction satisfies approximate functionality only computationally, i.e., it is hard to efficiently find an input point on which the obfuscated function differs from the original one.

oracle an unspecified polynomial number of times, regardless of the complexity of the adversary. This translates to allowing obfuscation schemes that leak secret information on some unspecified polynomial number of functions in a given family. In the context of digital lockers, this allows encryption schemes that, say, reveal the plaintext on a polynomial number of keys. We propose ways to fix this weakness in the definition and constructions of obfuscators and digital lockers; however our solution here is far from being completely satisfactory.

1.1 Related Work

Obfuscating Point Functions in the Random Oracle Model. Lynn *et al.* [11], inspired by the password-hiding scheme in Unix that stores a hash of the password instead of the password itself, propose a similar obfuscation of point functions in the random oracle model. In this model, an obfuscator, O, has oracle access to a truly random function, R. In order to construct an obfuscation of a point function, F_x, O queries R on x to get $z = R(x)$ and then stores z in the obfuscated code, $O(F_x)$. $O(F_x)$ also contains preprocessing code which on input y returns 1 if and only if $R(y) = z$.

It is easy to see that $O(F_x)$ and F_x have approximate functionality (they have the same functionality almost always). Intuitively, $O(F_x)$ is an obfuscation of F_x because R's answers on queries are completely independent and random. So, storing $R(x)$ does not reveal any information about x, but it allows verification of a guess, which is also achievable via oracle access to F_x.

Also, Lynn *et al.* [11] generalize this construction to obfuscate multibit output point functions and set-membership predicates and functions in the random oracle model. To obfuscate a multibit point function, $F_{x,y}$, choose a random r, and output $r, R_1(x,r), R_2(x,r) \oplus y$, where R_1 and R_2 denote the first and second half of the bits of $R(.)$. This construction is secure under composition (as in Definition 2 or the simply-composable definition of [11]). In Section 3.2, we instantiate this scheme. The resulting construction is more efficient than our first one but uses a stronger assumption.

Obfuscating Point Functions in the standard model. Perfectly one-way (POW) functions [4] can be used to obfuscate a point function F_x by replacing the random oracle in [11] with a POW function, H. Here, instead of storing $R(x)$, we store $H(x)$ in the obfuscated code and use the verifier for H to determine if $H(x)$ is a valid hash of the input.

Canetti [4] constructs a POW hash function based on a strong version of the Diffie-Hellman assumption. In particular, it assumes that the Diffie-Hellman assumption holds not only against uniform distributions but also with respect to any well-spread distribution. Moreover, Wee [13] shows how to obfuscate point functions and point functions with logarithmic output based on a strong one-way permutation assumption. Specifically, the assumption is that any polynomial-time machine can invert the permutation on at most a polynomial number of points. The two constructions mentioned so far (and our construction as well) use a weaker notion of obfuscation than the one in [2]. Specifically, the simulator in [4, 13] depends on the simulation-error gap between the adversary and the simulator. (see Definition 1 for more detail).

Canetti *et al.* [5] provide two constructions of POW functions based on standard computational assumptions (in particular, based on either claw-free permutations or one-way permutations). The simulator for these constructions does not depend on the gap. However, the input distribution is assumed to have high min-entropy (n^{ϵ}). Moreover, Futoransky *et al.* [7] show how to obfuscate point functions and point functions with multibit output based on standard assumption. However, the input distribution is assumed to be uniform. Finally, Hofheinz *et al.* [10] obfuscate point functions *deterministically*. However, the secrecy requirement does not guarantee no information leakage, rather that it is hard to recover the input in its entirety. This obfuscation is self-composable because the obfuscator is deterministic. However, it is not composable according to our notion. In particular, different obfuscated point functions can not be securely composed.

Encryption with imperfect randomness. The question of encryption with "imperfect randomness" is studied also in [12, 6, 3], yielding some strong impossibility results. However, these results do not apply to our case since they assume that the parties have no source of perfect randomness, whereas we allow the parties to use perfect randomness other than the key. In our setting, symmetric encryption with imperfect keys can be constructed using randomness extractors in standard ways, as long as the distribution of the key has sufficient min-entropy. Here however we are concerned with the case where there is no a priori guarantee on the min-entropy of the key.

1.2 Organization

In Section 2, we recall common notations and definitions including that of obfuscation, leaving definitions of POW functions to Appendix A. We present our construction and analyze it in Section 3. (We also present a more efficient construction under a stronger assumption in Section 3.2.) In Section 4, we study the issue of composable obfuscation. Finally, we discuss the connection to encryption schemes in Section 5.

2 Preliminaries

Let X_n denote a probability distribution on $\{0,1\}^n$ and U_n the uniform distribution on $\{0,1\}^n$. Then, $\mathbb{X} = \{X_n\}_{n\in\mathbb{N}}$ is called a distribution ensemble (distribution for short). A distribution is called **well-spread** if it has superlogarithmic min-entropy, i.e., $max_k Pr[X_n = k]$ is a negligible function in n. Moreover, $a \leftarrow D_n$ means that a is chosen from $\{0,1\}^n$ according to distribution D_n. Also, denote by $\Delta(X_n, Y_n)$ the statistical difference between the two distributions X_n and Y_n over $\{0,1\}^n$. Formally, $\Delta(X_n, Y_n) = \frac{1}{2}\Sigma_{a\in\{0,1\}^n}|Pr[X_n = a] - Pr[Y_n = a]|$.

A probabilistic function family is a set of efficient probabilistic functions having common input and output domains. Formally, $\mathbf{H}^n = \{H_k\}_{k\in K_n}$ is a function family with key space K_n and randomness domain R_n if, for all $k \in K_n, H_k : I_n \times R_n \to O_n$. A probabilistic function family has **public randomness** if for all k, $H_k(x,r) = r, H'_k(x,r)$ for some deterministic function H'_k. A family ensemble is a collection of function families, i.e., $\mathbb{H} = \{\mathbf{H}^n\}_{n\in\mathbb{N}}$.

Let PPT denote any probabilistic polynomial-time Turing machine, and nonuniform PPT any probabilistic polynomial-sized circuit family. A PPT (respectively nonuniform PPT) A with oracle access to O is denoted by A^O.

A function, μ, is called negligible if it decreases faster than any inverse polynomial. Formally, it is negligible if, for any polynomial p, there exists an N_p such that, for all $n \geq N_p$: $\mu(n) < \frac{1}{p(n)}$. In this work, we reserve μ to denote negligible functions. An uninvertible function, f, with respect to a well-spread distribution, $\mathbb{X}$, is an efficiently computable function that is hard to invert on $\mathbb{X}$. Formally, for any PPT, A, $Pr[x \leftarrow X_n,\ A(f(x)) = x] < \mu(n)$.

A **set-membership predicate**, $F_{S=\{x_1,\ldots,x_t\}} : \{0,1\}^n \to \{0,1\}$, outputs 1 if and only if its input is in S. Here, S is assumed to have at most polynomially many elements. A **set-membership function**, $F_{(x_1,y_1),\ldots,(x_t,y_t)} : \{0,1\}^n \to \{y_1,...,y_t,0\}$ outputs y_i if and only if the input matches x_i.

2.1 Obfuscation

We adopt the definition of obfuscation used in [4, 13] because obfuscation of point functions is known for this notion only (if the distribution on this class of functions is not restricted). This definition is weaker than the one in [2] because the size of the simulator is allowed to depend on the quality of the simulation. Formally,

Definition 1 (Obfuscation). *Let $\mathbb{F}$ be any family of functions. A PPT, O, is called an* ***obfuscator*** *of $\mathbb{F}$, if:*

1. ***Approximate Functionality*** *For any $F \in \mathbb{F}$: $Pr[\exists x,\ O(F)(x) \neq F(x)]$ is negligible. Here, the probability is taken over the coin tosses of O.*
2. ***Polynomial Slowdown*** *There is a polynomial p such that, for any $F \in \mathbb{F}$, $O(F)$ runs in time at most $p(T_F)$, where T_F is the worst-case running time of F.*
3. ***Weak Virtual Black-box Property*** *For any nonuniform PPT A and any polynomial p, there exists a nonuniform PPT S such that for any $F \in \mathbb{F}$ and sufficiently large n:*
$$|Pr[b \leftarrow A(O(F)) : b = 1] - Pr[b \leftarrow S^F(1^{|F|}) : b = 1]| \leq \frac{1}{p(n)}.$$

3 Obfuscating Point Functions with Multibit Output

We show how to obfuscate point functions with multibit output as well as set-membership predicates and functions for polynomial-sized sets. Because the constructions and proofs for obfuscating set-membership predicates and functions are similar to that for multibit point function, we focus on the latter. We comment on the former in Section 3.1. We also present a more efficient obfuscation of multibit point functions using a stronger assumption in Section 3.2.

We use obfuscated point functions as building blocks in obfuscating point functions with multibit output. The idea is simple. To obfuscate $F_{x,y}$, we encode y bit-by-bit using an obfuscator for F_x. Specifically, if the ith bit of y is 1, it is encoded as an obfuscation of F_x, otherwise, it is encoded as an obfuscation of an independent and uniform point function. In more detail, let H be a randomized obfuscator for point functions. Then the obfuscation contains $H(F_x, r), H(F_{x_1}, r_1), ..., H(F_{x_t}, r_t)$, where $t = |y|$ and $x_i = x$ if the ith bit of y is 1, otherwise, x_i is uniform. The first obfuscated point functions always corresponds to x, and is used to check whether the input is actually x. Now, y can be recovered given $z = x$. First, check that $H(F_x, r)(z) = 1$. If so, for every i, $y_i = H(F_{x_i}, r_i)(z)$.

Formally, we present an obfuscator, O, for the class of multibit output point functions, $\mathbb{F}$. O, on input $F_{x,y}$, where y has length t, selects $r_1, ..., r_{t+1}$ from R_n, the randomness domain of the point function obfuscator, H. It then computes $H(F_x, r_1)$. It also computes $H(F_x, r_{i+1})$ if the $y_i = 1$ and $H(z_{i+1}, r_{i+1})$ otherwise, where z_{i+1} is uniform. Let $u_x = u_1, ..., u_{t+1}$ be the sequence of obfuscated functions just computed. Then O outputs the following obfuscation, $O(F_{x,y})$, with u_x stored in it.

input: a

if $u_1(a) = 0$ **then**
    return 0;
else
    for $i \leftarrow 2$ **to** $t + 1$ **do**
        if $u_i(a) = 1$ **then**
            $y_{i-1} \leftarrow 1$;
        else
            $y_{i-1} \leftarrow 0$;
        return $y = y_1, ..., y_t$;
    end

Algorithm 1: $O(F_{x,y})$

Analysis. This construction is simple and modular. It is possible to replace H by any relative of point function obfuscation such as POW functions (see Appendix A) and analyze the security of the construction based on the security of the underlying primitive. We would like to prove that our construction is secure based on the simple assumption that the underlying primitive is an obfuscation of point functions. However, as we show in Section 4, this is not possible. This is so because the definition of obfuscation does not guarantee even self-composition. Thus, there exist point function obfuscators and POW functions for which this construction is provably insecure.

We investigate the secrecy of this construction based on three underlying primitives with different composition properties. In the first case, we consider the notion of composable obfuscation (as in Definition 2, also known as simply-composable obfuscation in [11]). We show a characterization that composable point function obfuscation exists if and only if composable multibit point function obfuscation exists. In the second case, we show that if H is a statistically indistinguishable POW function, then our construction is secure. Finally, if H is a computationally indistinguishable POW then this construction satisfies a weaker form of obfuscation where y, in $F_{x,y}$, has to be independent of x.

Analysis based on composable obfuscation. In this work, composable obfuscation refers to the fact that concatenating any sequence of obfuscated functions, where the functions are taken from the same class, constitutes an obfuscation for that sequence of functions. This form of composition, also known as simply-composable obfuscation in [11], should not be confused with self-composition which means that concatenating a sequence of independent obfuscation of the same function does not compromise secrecy. Formally,

Definition 2 **(t-Composable Obfuscation, [11]).** *Let $\mathbb{F}$ be any family of functions. A PPT, O, is called a t-composable obfuscator for $\mathbb{F}$, if:*

1. *Approximate functionality and polynomial slowdown are as before.*
2. ***Virtual Black-box property** For any nonuniform PPT, A, and any polynomial, p, there is a nonuniform PPT, S, such that for any functions $F_1, ..., F_{t(n)} \in \mathbb{F}$ (n is a security parameters, e.g., $n = |F_1| = ... = |F_{t(n)}|$) and sufficiently large n:*

$$|Pr[b \leftarrow A(O(F_1), ...O(F_{t(n)}) : b = 1] - Pr[b \leftarrow S^{F_1,...,F_{t(n)}}(1^n) : b = 1]| \leq \frac{1}{p(n)}$$

If O is a t-composable obfuscator for $\mathbb{F}$ for any polynomial t, then it is called a composable obfuscator.

If H satisfies $(t+1)$-composable obfuscation for some t, then our construction can be shown to be an obfuscation of multibit point function with output length t. Approximate functionality and polynomial slowdown follow from the corresponding properties on H. By the virtual black-box property on H, the output of $A(O(F_{x,y}) = O(F_x), O(F_{x_1}), ..., O(F_{x_{t(n)}}))$ can be simulated by $S^{F_x, F_{x_1},...,F_{x_{t(n)}}}(1^n)$, where $x_i = F_x$ if $y_i = 1$ and x_i is uniform otherwise. Moreover, oracle access to $F_x, F_{x_1}, ..., F_{x_{t(n)}}$ can be simulated with oracle access to $F_{x,y}$: If S queries any of its oracle on a point z such that $F_{x,y}(z) = 0$, then answer 0 (this may incur a negligible simulation error only), otherwise, $z = x$ so y can be fully recovered. Thus, this construction satisfies the virtual black-box property.

Observe that our construction is a *composable* obfuscation of multibit point functions with the appropriate parameters. Specifically, if the output length of the multibit point function is restricted to at most t, then this construction is a t'-composable obfuscation if H is $(t+1)t'$-composable. In addition, it is easy to see that the existence of a t-composable obfuscation of multibit point functions implies a t-composable obfuscation of point functions. Formally, we have the following characterization with a proof that follows the above discussion.

Theorem 1. *Composable obfuscators of point functions with multibit output exist if and only if composable obfuscators of point functions exist.*

Specifically, if a point function obfuscator, H, is $(t+1)t'$-composable (as in Definition 2) then the above construction is a t'-composable obfuscation of multibit point functions with output length t. On the other hand, a t-composable obfuscation of multibit point functions implies a t-composable obfuscation of point functions.

Analysis based on statistical indistinguishability. Suppose $\mathbb{G}$ is a statistically indistinguishable POW family ensemble (see Appendix A for the formal definition). We can replace H by $\mathbb{G}$ in the above construction. Specifically, the obfuscator, O, samples a key, k, for $\mathbb{G}$ and replaces $H(x, .)(a)$ with $V(a, G_k(x, .))$, where V is the verification algorithm for $\mathbb{G}$. This results in an obfuscation of point function with multibit output except with *computational approximate functionality* [13], i.e, no adversary can efficiently find a point on which the original function differs from the obfuscated one. This relaxation to approximate functionality is necessary when using statistical POW functions because they can not be statistically collision resistant. On the other hand, we argue that the result satisfies the virtual-blackbox property. Informally, from the fact that $\mathbb{G}$ is a statistical POW function we can conclude that an obfuscation of $F_{x,y}$, where x is taken from a well-spread distribution and y is arbitrary, is statistically close to a sequence of hashes of random inputs. It follows that for all but polynomially many x, an obfuscation of $F_{x,y}$ is indistinguishable from random hashes. Consequently, we get a simulator that runs the adversary on random hashes unless x is taken from that polynomial set, in which case the simulator can recover y and run the adversary on an obfuscation of $F_{x,y}$. Formally,

Theorem 2. *Let $\mathbb{G}$ be a statistically $(t+1)$-indistinguishable POW function (as in Definition 8). Then, the above construction is an obfuscation of point functions with multibit output length t (as in Definition 1), where approximate functionality is only computational.*

Proof (Sketch). Polynomial slowdown follows immediately from the fact that $\mathbb{G}$ has a polynomial output length. Also, by public verification and collision resistance of POW functions (definition 6), it follows that O satisfies computational approximate functionality.
Virtual black-box property. Recall, the definition of statistical indistinguishability says that for any well-spread distribution, $\mathbb{X}$:

$$\Delta(G_k(X_n, R_n^1), ..., G_k(X_n, R_n^{(t+1)(n)}), G_k(U_n^1, R_n^1), ..., G_k(U_n^{t(n)}, R_n^{(t+1)(n)})$$

is negligible, where each distribution R_n^i (respectively, U_n^i) is the same as R_n (respectively, U_n).

Using the fact that for any function, λ, $\Delta(\lambda(X), \lambda(Y)) \leq \Delta(X, Y)$, we have for any distribution,$\mathbb{XY}$ on (x, y), where the corresponding distribution on x is well-spread:

$$\Delta(O(F_{X_n,Y_n}), G_k(U_n^1, R_n^1), ..., G_k(U_n^{t(n)}, R_n^{(t+1)(n)}) \qquad (1)$$

is negligible. (We assume without loss of generality that $O(F_{x,y})$ consists only of the $t+1$ $\mathbb{G}$-hashes.)

Using the same technique from the proof of Theorem 4 in [4], it can be shown that $O(F_{x,y})$ is indistinguishable from $\mathbb{G}$-hashes of uniform strings on all but a polynomial number of x. That is, for any nonuniform PPT, A, and any polynomial, p, there exists a polynomial size family of sets, $\{L_n\}$, such that for sufficiently large n, and $x \notin L_n$ and any y:

$$|Pr[b \leftarrow A(O(F_{x,y})) : b = 1] - Pr[u_1, ..., u_{t+1} \leftarrow U_n, ..., U_n,$$
$$r_1, ..., r_{t+1} \leftarrow R_n, ..., R_n,\ b \leftarrow A(G_k(u_1, r_1), ..., G_k(u_{t+1}, r_{t+1})) : b = 1]| \leq \frac{1}{p(n)}. \qquad (2)$$

Intuitively, this is true because otherwise there is a super-polynomial number of values for x (with a corresponding value for y), on which A can distinguish $O(F_{x,y})$ from hashes of random strings. By defining a well-spread distribution, e.g., a uniform distribution, on this superpolynomial number of values for x, A violates (1).

Now, for any nonuniform PPT, A, and a polynomial, p, we construct a nonuniform PPT, S that simulates A. S receives the polynomial-size set, L_n, as an advice string. It checks if the oracle, $F_{x,y}$, responds with the nonzero value, y, to any element in the set, L_n. If so, then S can compute $O(F_{x,y})$ and simulate A on it. Otherwise, x is not in L_n, so S runs A on hashes of random inputs. By (2), this is close to a true simulation. For more detail, we refer the reader to the proof of Theorem 4 in [4]. □

Analysis based on computational indistinguishability. We would like to weaken the assumption in Theorem 2 to computational indistinguishability. However, it is not clear how to use computational indistinguishability, i.e., $G_k(x, r_1), ..., G_k(x, r_{t+1})$ is computationally indistinguishable from hashes of uniform, to conclude that $O(F_{x,y})$ is indistinguishable from hashes of random inputs. It seems that the problem lies in the potential dependence of y on x, e.g., y may be equal to x. This is not a problem in the statistical case because we can use the fact that statistical difference does not increase by applying the same function on both distributions. In the computational setting, if we use the traditional blackbox reduction, we need to construct $O(F_{x,y})$ from hashes of x and then run A on it. However, it is not clear how to do so if $y = x$. On the other hand, suppose y is independent of x, e.g., y is taken independently from a uniform distribution. Then, for some y, it is possible to compute $O(F_{x,y})$ given hashes of x, $G_k(x, r_1), ..., G_k(x, r_{t+1})$, by replacing $G_k(x, r_i)$ with a hash of a random string if the ith bit of y is 0. Thus, we know that computational indistinguishability gives us a weaker notion of obfuscation where the simulator depends on the distribution on y. Whether computational indistinguishability gives us the standard virtual-blackbox property remains unknown. Nevertheless, this weak obfuscation can be used as a digital locker as described in the introduction. The caveat is that the message being encrypted should be independent of the encryption key. This is the case if, for instance, the message is chosen without knowledge of the key.

Formally, the virtual black-box property becomes: for any nonuniform PPT A, any polynomial p, and any (efficiently samplable) distribution $\mathbb{Y}$, there exists a nonuniform PPT S such that for any x and sufficiently large n:

$$|Pr[y \leftarrow Y_n,\ b \leftarrow A(O(F_{x,y})) : b = 1] - Pr[y \leftarrow Y_n,\ b \leftarrow S^{F_{x,y}}(1^{|F_{x,y}|}) : b = 1]| \leq \frac{1}{p(n)}. \tag{3}$$

Also, we remark that this construction has either approximate or computational approximate functionality depending on whether the POW function satisfies statistical or computational collision resistance. Formally, we have the following theorem whose proof follows that of Theorem 2 and the above discussion, and is not recreated here.

Theorem 3. *If $\mathbb{G}$ is a computationally $(t + 1)$-indistinguishable POW function, then the above construction is an obfuscation of point function with output length t, where the virtual-blackbox property is as in (3).*

3.1 Obfuscating Set-membership Predicates and Functions

To obfuscate a set-membership predicate, simply obfuscate the point functions on every element in the set (this is feasible because the set has a polynomial size), and then store all the obfuscated functions in a randomly permuted order. To determine whether a particular input is in the set, we only need to check whether any of the obfuscated functions outputs 1 on this input. It can be shown that this construction is an obfuscation of set-membership predicate based on composable obfuscation of point functions. Moreover, to obfuscate a set-membership function, $F_{(x_1,y_1),\ldots,(x_t,y_t)}$, we only need to run the obfuscator for the multibit output point function on each F_{x_i,y_i}, and then store these obfuscated functions in a randomly permuted order. It can be shown that composable obfuscation of point functions is a necessary and sufficient condition for the security of this construction.

3.2 A More Efficient Obfuscation of Point Functions with Multibit Output

We note that the obfuscation of point function with multibit output in the RO model [11] can be instantiated by using a stronger assumption on the underlying primitive. The end result is a more efficient construction than the one described previously. Specifically, let $\mathbb{G}$ be a POW function with public randomness. To obfuscate $F_{x,y}$, select r_1 and r_2 uniformly from the randomness domain of $\mathbb{G}$ and output $G_k(x, r_1), r_2, z$, where $G_k(x, r_2) = (r_2, v)$ and $z = y \oplus v$.[2] To recover y from (a, b, c) and x', first check that $V(x', a) = 1$, if so, then return $y = c \oplus v$, where $G_k(x', b) = (b, v)$. Even though this construction is more efficient than the first one, it suffers from two problems. First, in order to completely hide y, it is not sufficient that $\mathbb{G}$ be indistinguishable as in Definition 9 rather its output has to be *indistinguishable from uniform*. If, for example, the first bit of the hash is always 0, then the first bit of y is revealed. Second, for the proof to go through, we need to assume that $\mathbb{G}$ is *statistically* indistinguishable from *uniform* because y may depend on x. Contrast this assumption with the one used in Theorem 2, where $\mathbb{G}$ is statistically indistinguishable from *hashes of uniform strings*.

4 On Composable Obfuscation of Point Functions

In Section 3, we provided a transformation from an obfuscation of a point function to an obfuscation of a point function with multibit output. This transformation requires an essential property on the given obfuscation, specifically, composition. In other words, our construction assumes that we have an obfuscation of a point function such that security is not compromised when multiple obfuscated functions are given. Notably, Theorems 1, 2, and 3 all assume that H satisfies some form of composable security. Since the obfuscator is probabilistic, composable security is nontrivial. In this section, we address this question. Specifically, does the basic definition of obfuscation imply composition? From a different angle, Canetti *et al.* [5] ask if semantic perfect one-wayness implies indistinguishable perfect one-wayness or if t-indistinguishable POW functions

[2] Without loss of generality, we assume that y and v have the same length. Otherwise, the input should be of a longer input, say $x0^t$.

are $t+1$-indistinguishable. We answer these questions negatively: such primitives are not necessarily secure even under self-composition. [3] In more detail, we show that weak c-indistinguishable POW functions (where the probability is taken over the choices of the seed as well, [5]) are not necessarily $c+1$-indistinguishable for any constant c. We also show that POW functions, POW functions with auxiliary input, and obfuscation of point functions do not imply composition. Specifically, 1-indistinguishable POW functions and obfuscation of point functions are not necessarily secure for a polynomial number of copies. Moreover, even though 1-indistinguishable POW functions with auxiliary input is also c-indistinguishable for any constant c, it is not necessarily t-indistinguishable with auxiliary input for a polylogarithmic t.

In Section 4.1, we show a tight impossibility result for weak POW functions. Specifically, we show that for any constant c, weak c-indistinguishable POW functions are not weakly $c+1$-indistinguishable. We also show that if t is polynomial, then weak t-indistinguishable POW functions are not weakly $n(t+1)^2$-indistinguishable. In Section 4.2, we prove that sematic POW functions, 1-indistinguishable POW functions, and point function obfuscation are not secure if composed roughly $nlog(n)$ times. Moreover, if we consider the same functions with respect to auxiliary information, then we have a tighter result where they are not secure with respect to auxiliary information if composed superlogarithmically-many times.

4.1 Weak POW functions Are Not Self-composable in General

A weak POW function deviates from Definition 9 in that the probability is taken over the choices of the function key as well. Here, we show that a weak c-indistinguishable POW function with respect to the uniform distribution may not be $c+1$ indistinguishable for any constant c. The idea is simple: we take any weak $3c$-indistinguishable POW function and convert it into a new function that is c-indistinguishable but the output contains shares of the input such that it is easy to compute the input from $c+1$ hashes. Informally, we add c uniform strings to the original seed and make sure that a hash of the input using any one of those c strings appears in the output with probability $\frac{1}{c+1}$. Also, with the same probability the exclusive-or of the input and all the aforementioned hashes appears in the output. Therefore, if the output of the function contains all c hashes and the exclusive-or of these hashes with the input, then it is easy to recover the input.

Formally, let $\mathbb{H}$ be any (possibly weak) $3c$-indistinguishable POW function with key space, K_n, and public randomness. We also assume that $\mathbb{H}$ is also $3c$-indistinguishable from uniform. Define a new family ensemble, $\mathbb{G}$, with a key space $(K_n, \underbrace{R_n, ..., R_n}_{c})$, an input domain $(\{0,1\}^n, \{0,1\}^n)$, and randomness domain $(R_n, \{0,1\}^{logc})$, as follows:

$$G_{k,u_1,...,u_c}((x_1,x_2),(r_1,r_2)) =$$

$$\begin{cases} r_2, H_k(x_1,r_1), H_k(x_2,r_1), H_k(x_1,u_{r_2}) & \text{if } r_2 \neq 0 \\ r_2, H_k(x_1,r_1), H_k(x_1,u_1) \oplus H_k(x_1,u_2) ... \oplus H_k(x_1,u_c) \oplus x_2 & \text{if } r_2 = 0 \end{cases}$$

[3] Recall, self-composition refers to concatenation of multiple outputs of a randomized function on the *same* input.

Now, observe that it is easy to recover x_2 from $G_{k,u_1,\ldots,u_c}((x_1,x_2),(r_1^0,0)),\ldots,$ $G_{k,u_1,\ldots,u_c}((x_1,x_2),(r_1^c,c))$. Thus, $\mathbb{G}$ is not $(c+1)$-indistinguishable because $c+1$ randomly-chosen hashes of (x_1,x_2) have distinct r_2 (i.e., match the aforementioned hashes) with probability $\frac{(c+1)!}{(c+1)^{c+1}}$. On the other hand, we argue that $\mathbb{G}$ is a weak c-indistinguishable POW function with respect to the uniform distribution . First, completeness and collision resistance follow from that on $\mathbb{H}$. Second,

$$G_{k,u_1,\ldots,u_c}((x_1,x_2),(r_1^1,r_2^1)),\ldots,G_{k,u_1,\ldots,u_c}((x_1,x_2),(r_1^c,r_2^c))$$

is indistinguishable from

$$G_{k,u_1,\ldots,u_c}((v_1,x_2),(r_1^1,r_2^1)),\ldots,G_{k,u_1,\ldots,u_c}((v_c,x_2),(r_1^c,r_2^c))$$

by the $3c$-indistinguishability property on $\mathbb{H}$, where $v_1,\ldots,v_c$ are uniform and independent. Moreover, by the $3c$-indistinguishability from uniform, we have

$$G_{k,u_1,\ldots,u_c}((v_1,x_2),(r_1^1,r_2^1)),\ldots,G_{k,u_1,\ldots,u_c}((v_c,x_2),(r_1^c,r_2^c))$$

is indistinguishable from

$$G_{k,u_1,\ldots,u_c}((v_1,w_1),(r_1^1,r_2^1)),\ldots,G_{k,u_1,\ldots,u_c}((v_c,w_c),(r_1^c,r_2^c)),$$

where $w_1,\ldots,w_c$ are uniform and independent.

Moreover, this result can be generalized to any polynomial t. If $\mathbb{H}$ is $3t$-indistinguishable from uniform, then $\mathbb{G}$ is a weak t-indistinguishable POW function with respect to the uniform distribution. On the other hand, $\mathbb{G}$ is not $n(t+1)^2$-indistinguishable with respect to the uniform distribution. This is so because all the $(t+1)$ "shares" appear in $n(t+1)^2$ hashes with overwhelming probability. This result is stated formally in the following theorem.

Theorem 4. *Let $\mathbb{H}$ be any weak POW function that is $3t$-indistinguishable from uniform and has public randomness. Then for any constant $c \leq t$, there exist weak POW functions that are c-indistinguishable (respectively, t-indistinguishable) with respect to the uniform distribution but not $c+1$-indistinguishable (respectively, $n(t+1)^2$-indistinguishable) with respect to the uniform distribution.*

4.2 Point Function Obfuscation and POW Functions Are Not Self-composable in General

We show that POW functions, POW functions with auxiliary input, obfuscation of point functions, and obfuscation of point functions with auxiliary input are not generally self-composable. Also, we note that the obfuscation of point functions in [13] is not self-composable as well. The idea is simple, we start with a POW function and append to its output a hardcore bit, specifically the inner product between the input and a random string. This hardcore bit does not compromise security of a single hash. However, the function becomes completely insecure for polynomially many hashes as the input can be recovered with high probability by solving a linear system of equations.

Here, we present the proof for the case of POW functions with auxiliary input only. Let $\mathbb{H}$ be a POW function that is 1-indistinguishable with auxiliary input. Define a new family ensemble, $\mathbb{G}$:

$$G_k(x, (r_1, r_2)) = r_2, H_k(x, r_1), \langle x, r_2 \rangle,$$

where $\langle x, r_2 \rangle$ is the inner product of x and r_2 mod 2. We argue that $\mathbb{G}$ is 1-indistinguishable with auxiliary input. First, completeness and collision resistance follow from that on $\mathbb{H}$. Moreover, for any uninvertible function F, $F(x), H(x, r_1), r_2$ is one-way in x because $\mathbb{H}$ is 1-indistinguishable with auxiliary input. Therefore, by Goldreich-Levin theorem [8], we have that $F(x), r_2, H(x, r_1), \langle x, r_2 \rangle$ is indistinguishable from $F(x), r_2, H(x, r_1), b$, where b is uniform. Moreover, by 1-indistinguishability with auxiliary input on $\mathbb{H}$, $F(x), r_2, H(x, r_1), b$, is indistinguishable from $F(x), r_2, H(U_n, r_1), b$.

On the other hand, $\mathbb{G}$ is not polylogarithmically indistinguishable with auxiliary input. To see that, let F be a function that outputs the last $n - \omega(1)log(n)$ bits of its input. Then, F is uninvertible with respect to the uniform distribution. However, we argue that given $F(x)$ and a polylogarithmic number of hashes, x can be recovered completely by solving a system of linear equations. Formally,

Lemma 1. *For any two constants c and ϵ, there exists a t, which is polylogarithmic in n (specifically, $t = \omega(1)log(n)log\frac{\omega(1)log(n)}{-ln(\frac{1}{n^c}+\epsilon)}$) and a PPT, A, such that for any $k \in K_n$:*

$$Pr[x \leftarrow U_n,\ r_1, ..., r_t \leftarrow R_n^G, ..., R_n^G,\ A(F(x), G_k(x, r_1), ..., G_k(x, r_t))] \geq \frac{1}{n^c}.$$

Proof. Let A be a PPT that ignores all $\mathbb{H}$ hashes ($H_k(x, .)$) but plugs-in the values of the last $n - \omega(1)log(n)$ bits of x in the system of linear equations: $r_1^2, \langle x, r_1^2 \rangle, ..., r_t^2, \langle x, r_t^2 \rangle$. We show that by solving this system we can recover x with probability $\frac{1}{n^c}$. Given the last $n - \omega(1)log(n)$ bits of x revealed by F, we can recover x from $\omega(1)log(n)$ linearly independent equations on the first $\omega(1)log(n)$ bits. Thus, we need to show that we have this many linearly independent equations in t uniformly chosen equations with probability $\frac{1}{n^c}$. First, observe that a uniform and independent r is linearly independent from $\omega(1)log(n) - 1$ or less equations with probability at least $\frac{1}{2}$. Consequently, the probability that t equations contain $\omega(1)log(n)$ linearly independent equations is at least:

$$(1 - \frac{1}{2^{log\frac{\omega(1)log(n)}{-ln(\frac{1}{n^c}+\epsilon)}}})^{\omega(1)log(n)} \geq e^{ln(\frac{1}{n^c}+\epsilon)} - \epsilon = \frac{1}{n^c}.$$

□

Using the same construction, $\mathbb{G}$, it is possible to show that 1-indistinguishable POW functions (respectively obfuscation of point functions) are not necessarily t-indistinguishable (respectively, secure under t-self-composition), where $t = nlog\frac{n}{-ln(\frac{1}{n^c}+\epsilon)}$. As a concrete example, the same analysis can be used to show that the obfuscation of point function in [13] is not secure when composing t obfuscated copies of the same point function. These results can be stated formally as follows.

Theorem 5. *If there exists a* 1*-indistinguishable POW function (respectively, a point function obfuscation) with auxiliary input then there exists another* 1*-indistinguishable POW function (respectively, another point function obfuscation) with auxiliary input such that for any constants* c *and* ϵ*, the latter is not* t*-indistinguishable (respectively, is not a* t*-self-composable point function obfuscation) with auxiliary input with respect to the uniform distribution , where* $t = \omega(1)log(n)log\frac{\omega(1)log(n)}{-ln(\frac{1}{n^c}+\epsilon)}$.

Moreover, if there exists a 1*-indistinguishable POW function (respectively, a point function obfuscation) then there exists another* 1*-indistinguishable POW function (respectively, another point function obfuscation) such that for any constants* c *and* ϵ*, the latter is not* t*-indistinguishable (respectively, is not a* t*-self-composable point function obfuscation) with respect to the uniform distribution, where* $t = nlog\frac{n}{-ln(\frac{1}{n^c}+\epsilon)}$.

5 On the Relationship Between Obfuscation of Point Functions with Multibit Output and Symmetric Encryption

It is interesting to note that obfuscation of point functions with multibit output and symmetric encryption are similar. At the conceptual level, they capture the same idea except with a subtle difference. First, both of them satisfy the same correctness property. In particular, an encryption scheme (respectively, obfuscation of point function with multibit output) allows the recovery of the message (respectively, y) given the key (respectively, x). Second, they share similar privacy requirements. An obfuscation hides the special output, y, of the function, $F_{x,y}$ unless x is given. Likewise, a symmetric encryption should ensure the privacy of the message unless the adversary possesses the key. However, the former primitive differs from the latter in that its behavior is defined over all possible input x, while the decryption scheme leaves the behavior undefined on wrong keys. In other words, one may, at least conceptually, think of an obfuscation of point functions with multibit output as a *special* form of encryption, where wrong keys are promptly detected by the decryption algorithm.

At a more technical level, another difference arises, regarding the assumption on the key distribution. Recall that symmetric encryption requires uniform keys. On the other hand, an obfuscation of point functions with multibit output does not assume anything about the distribution on x. Specifically, it provides a definition of privacy for any x. Thus, casting the former primitive as an encryption scheme ,i.e., as $O(F_{key,message})$, gives us an encryption scheme with the same privacy as defined for obfuscation. In other words, any predicate computed from the ciphertext can also be computed by exhaustively searching for the right key to recover the message. Formally,

Definition 3 (Single-message encryption for any key). *A symmetric encryption scheme,* (E, D)*, satisfies privacy for* ***any*** *key if for any nonuniform PPT* A*, and any polynomial* p*, there exists a nonuniform PPT* S *such that for any key,* k*, any message,* m*, and sufficiently large* n*:*

$$|Pr[b \leftarrow A(E(k,m)) : b = 1] - Pr[b \leftarrow S^{F_{k,m}}(1^n) : b = 1]| \leq \frac{1}{p(n)}.$$

Observe that in the special case where the key is uniform or even sampled from a well-spread distribution, Definition 3 implies that whatever predicate computed from the ciphertext can be computed *without it (and without oracle access to $F_{k,m}$)*. Formally, an encryption scheme satisfying Definition 3 also satisfies the following privacy property.

Definition 4 (Single-message encryption with well-spread keys). *A symmetric encryption scheme, (E, D), satisfies privacy for* ***well-spread*** *keys if for any nonuniform PPT A, and any polynomial p, there exists a nonuniform PPT S such that for any well-spread distribution, $\mathbb{K} = \{K_n\}_{n\in\mathbb{N}}$, any message m, and sufficiently large n:*

$$|Pr[k \leftarrow K_n,\ b \leftarrow A(E(k,m)) : b = 1] - Pr[b \leftarrow S(1^n) : b = 1]| \leq \frac{1}{p(n)}.$$

Although Definitions 3 and 4 consider single-message encryption, encryption of multiple messages can be readily achieved using appropriately composable obfuscation of point functions with multibit output.

5.1 Weakness of Definition 3

It may seem that Definition 3 captures our intuition that the only way of breaking the encryption scheme is through exhaustively searching for the correct key. However, it turns out that this definition is not strong enough. Specifically, there are encryption schemes that satisfy this definition but reveal the plaintext when the key is taken from a polynomial-size set. For instance, modify any encryption scheme that satisfies Definition 3 so that it reveals the plaintext when the key is one of the first n lexicographically-ordered keys. The new scheme still satisfies this definition because the simulator can query the oracle on those n keys to recover the message. However, this scheme does not match our intuitive requirement. This is so because an adversary can, in constant time, output the first bit of the plaintext on the first n keys but the simulator needs $O(n)$ time to do the same. We stress that this weakness is already inherent in the notion of obfuscation, not just in the application to encryption.

Coming up with a realizable definition that captures our intuition about encryption with low-entropy keys is interesting, given the potential applications in password-based encryption. Here, we take a step in this direction. We strengthen Definition 3 by restricting the number of queries of the simulator to some fixed polynomial in the running time of the adversary and the simulation error. In more detail, for any key, k, the number of queries the simulator makes in the worst case is bounded by a fixed polynomial in the worst-case running-time of the adversary, and the simulation error.

Definition 5 (t-secure encryption). *A symmetric encryption scheme, (E, D), is t-secure if for any nonuniform PPT A, and any polynomial p, there exists a nonuniform PPT S such that for any key, k, any message, m, and sufficiently large n:*

$$|Pr[b \leftarrow A(E(k,m)) : b = 1] - Pr[b \leftarrow S^{F_{k,m}}(1^n) : b = 1]| \leq \frac{1}{p(n)},$$

where S makes at most $t(R_{A,k,m}, n, p)$ queries and $R_{A,k,m}$ is the worst-case running time of A on $E(k,m)$, taken over the coin tosses of A and E.

The definition of obfuscation can also be strengthened in a similar way. Obviously, the smaller t is, the stronger the security guarantee. For instance, if an encryption scheme (respectively, obfuscation) is t-secure then it (respectively, the obfuscator) can not do certain "stupid" things such as outputting the plaintext (respectively, the original function) in the clear on more than $\frac{nt(|E(.,.)|,n,n)}{n-1}$ keys (respectively, $\frac{nt(|O(.)|,n,n)}{n-1}$ functions). We note that the construction in Section 3 satisfies this definition for some specific t. However, the question remains as to how small t can be made.

6 Acknowledgements

We are grateful to Joan Feigenbaum and Hoeteck Wee for useful comments and discussions. In particular, permuting the obfuscated programs for obfuscating set-membership predicates and functions was gratefully pointed out by Hoeteck Wee. Also, we thank the anonymous referees for constructive feedback and suggestions that helped in improving the final draft. Notably, the weakness in the definition of obfuscation and digital lockers was highlighted by a referee.

References

1. Firefox password manager. *http://www.firefoxtutor.com/61/securing-firefox-passwords/*.
2. B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. Vadhan, and K. Yang. On the (im)possibility of obfuscating programs. *Proceedings of CRYPTO 2001*, Springer LNCS 2139:1–18, 2001.
3. C. Bosley and Y. Dodis. Does privacy require true randomness? *Theory of Cryptography Conference: TCC 2007*, 2007.
4. R. Canetti. Towards realizing random oracles:hash functions that hide all partial information. *Proceedings of Crypto 1997*, Springer LNCS 1294:455–469, 1997.
5. R. Canetti, D. Micciancio, and O. Reingold. Perfectly one-way probabilistic hash functions. *Proceedings of the 30th ACM Symposium on Theory of Computing*, pages 131–140, 1998.
6. Y. Dodis and J. Spencer. On the (non)universality of the one-time pad. *43rd Symposium on Foundations of Computer Science*, 2002.
7. A. Futoransky, E. Kargieman, C. Sarraute, and A. Waissbein. Foundations and applications for secure triggers. *eprint, 284*, 2005.
8. O. Goldreich and L. Levin. Hard-core predicates for any one-way function. *Proceedings of the 21st ACM symposium on Theory of computing*, 1989.
9. Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Science*, 28:270–299, 1984.
10. D. Hofheinz, J. Malone-Lee, and M. Stam. Obfuscation for cryptographic purposes. *Theory of Cryptography Conference: TCC 2007*, pages 214–232, 2007.
11. B. Lynn, M. Prabhakaran, and A. Sahai. Positive results and techniques for obfuscation. *Proceedings of Eurocrypt 2004, Springer LNCS 3027*, pages 20–39, 2004.
12. J. McInnes and B. Pinkas. On the impossibility of private key cryptography with weakly random keys. *Proceedings of Crypto 1990*, 1990.
13. H. Wee. On obfuscating point functions. *Proceedings of the 37th ACM symposium on Theory of computing*, pages 523 – 532, 2005.

A Perfectly One-way Probabilistic Hash Functions

A perfectly one-way hash function, POW for short, is a probabilistic function that satisfies collision resistance and hides all information about its input. Due to its probabilistic nature, such a function is coupled with an efficient verification algorithm that determines, given (x, y), whether y is a valid hash of x. Usually, collision resistance of *deterministic* hash functions requires that it is hard to find two input strings mapped to the same hash. However, because these functions are probabilistic by nature, we need to modify collision resistance to take the verification process into account. In particular, collision resistance says that it is hard to find two input and one output strings such that the verification scheme accepts the output as a valid hash of both input points. Formally,

Definition 6 (Public Verification, [4]). *A family ensemble, $\mathbb{H} = \{\mathbf{H}^n\}_{n\in\mathbb{N}}$, satisfies **public verification** if there exists a deterministic polynomial-time algorithm V such that:*

1. *Completeness:* $\forall k \in K_n,\ x \in \{0,1\}^n,\ r \in R_n,\ \ V(x, H_k(x, r)) = 1.$
2. *Collision Resistance: For any nonuniform PPT, A:*

$$Pr[k \leftarrow K_n,\ (x_1, x_2, y) \leftarrow A(k) :\ x_1 \neq x_2 \wedge V(x_1, y) = V(x_2, y) = 1] < \mu(n).$$

There are several ways to formulate information hiding, some of which are not equivalent. We start with the most basic definition, namely semantic perfect one-wayness, and later present two more definitions, namely, statistical and computational indistinguishability. Semantic perfect one-wayness has its roots in semantic security of probabilistic encryption [9] which requires that every function that can be computed given the ciphertext can also be computed without it. However, the notion of secrecy in this setting is slightly weaker than semantic security because a hash can be used to verify whether a guess is correct or not. This notion is captured by a simulation-based definition which requires that every predicate computable given a hash can also be computed by a simulator with oracle access to the corresponding point function. Formally,

Definition 7 (Semantic Perfect One-wayness, [4]). *A family ensemble $\mathbb{H} = \{\mathbf{H}^n\}_{n\in\mathbb{N}}$, is called **semantically perfectly one-way** if it satisfies public verification (Definition 6) and, for any nonuniform PPT, A, and polynomial, p, there exists a nonuniform PPT S such that for sufficiently large n, any k, and any x:*

$$|Pr[r \leftarrow R_n,\ b \leftarrow A(k, H_k(x, r)) : b = 1]\ -$$

$$Pr[r \leftarrow R_n,\ b \leftarrow S^{F_x}(k) : b = 1]| \leq \frac{1}{p(n)}.$$

Recall F_x is the point function on x.

Remark 1. Note that semantic perfect one-wayness corresponds in a straightforward way to the virtual blackbox property required for obfuscating point functions in Definition 1. Thus, a function satisfying definition 7 is an obfuscation of a point function (with computational approximate functionality). However, the converse may not be true.

In more detail, let $\mathbb{H}$ be a semantic POW function. To obfuscate F_x, sample a seed, k, and random string, r, for $\mathbb{H}$ and output the obfuscation, $O(F_x) = k, H_k(x, r)$. The new function, $O(F_x)$, simply computes the predicate $V(., H_k(x, r))$. It can be shown that O is an obfuscator for the class of point functions. Completeness and collision resistance on $\mathbb{H}$ imply computational approximate functionality while semantic perfect one-wayness implies the virtual-blackbox property. On the other hand, an obfuscation of point functions may not be a POW function because approximate functionality does not rule out collisions chosen in an adversarial way.

As mentioned in the introduction, neither Definition 1 nor Definition 7 is sufficient for the security of our construction in Section 3 because they do not guarantee composition. Thus, we analyze our construction based on primitives with different composable properties. Two of these primitives are statistical and computational POW functions, which are defined in the rest of this appendix.

Statistical Perfect One-wayness. Statistical information hiding is captured by requiring statistical closeness between hashes of the same input and those of different inputs.

Definition 8 (Statistical t-Indistinguishability). *A family ensemble $\mathbb{H} = \{\mathbf{H}^n\}_{n\in\mathbb{N}}$, where $H_k : \{0,1\}^n \times R_n \to \{0,1\}^{l(n)}$ for some polynomial l, is called **statistically t-indistinguishable** if it satisfies public verification (Definition 6) and for any well-spread distribution $\mathbb{X} = \{X_n\}_{n\in\mathbb{N}}$ and any $k \in K_n$,*

$$\Delta(\underbrace{H_k(X_n, R_n^1), ..., H_k(X_n, R_n^{t(n)})}_{t(n)}, \underbrace{H_k(U_n^1, R_n^1), ..., H_k(U_n^{t(n)}, R_n^{t(n)})}_{t(n)}) \le \mu(n),$$

where each distribution R_n^i (respectively, U_n^i) is the same as R_n (respectively, U_n).

Moreover, if $\mathbb{H}$ is statistically t-indistinguishable for any polynomial, t, then it is called statistically indistinguishable.

We note that the first construction in [5] is slightly weaker than Definition 8 in that the input distribution has n^ϵ min-entropy instead of superlogarithmic min-entropy. Constructing functions with the latter property remains an open problem.

Computational Perfect One-wayness. Computational perfect one-wayness differs from statistical perfect one-wayness in two main ways. The first and obvious difference is that indistinguishability holds for *polynomially-bounded adversaries* only. Second, computational perfect one-wayness differs depending on whether we take the presence of auxiliary information into account. In this context, we restrict the notion of auxiliary information to uninvertible functions about the input.

Instead of explicitly writing two definitions, one with auxiliary information and another without it, we present here one definition only. To take both cases into account, we use the convention that auxiliary information is surrounded by boxes. So, by removing $\boxed{\text{the words in boxes}}$ from Definition 9, we get the first definition while keeping the boxes gives us the second one. Formally,

Definition 9 (t-Indistinguishability).

*Let $\mathbb{X} = \{X_n\}_{n\in\mathbb{N}}$ be any well-spread distribution. Let F be any (possibly probabilistic) uninvertible function. A family ensemble $\mathbb{H} = \{\mathbf{H}^n\}_{n\in\mathbb{N}}$, where $H_k : \{0,1\}^n \times R_n \to \{0,1\}^{l(n)}$ for some polynomial l, is called **t-indistinguishable** with respect to $\mathbb{X}$, $\boxed{\text{with auxiliary input } F}$, if it satisfies public verification (Definition 6) and for any $k \in K_n$ and any PPT A:*

$$|Pr[x \leftarrow X_n,\ \boxed{z \leftarrow F(x)},\ (r_1, ..., r_t) \leftarrow (R_n, ..., R_n) :$$

$$A(k, \boxed{z}, H_k(x, r_1), ..., H_k(x, r_t)) = 1] \quad -$$

$$Pr[x \leftarrow X_n,\ (u_1, ..., u_t) \leftarrow (U_n, ..., U_n),\ \boxed{z \leftarrow F(x)},\ (r_1, ..., r_t) \leftarrow (R_n, ..., R_n) :$$

$$A(k, \boxed{z}, H_k(u_1, r_1), ..., H_k(u_t, r_t)) = 1]| \leq \mu(n).$$

If $\mathbb{H}$ is t-indistinguishable $\boxed{\text{with any auxiliary input } F}$ with respect to any well-spread distribution $\mathbb{X}$, then it is called t-indistinguishable $\boxed{\text{with auxiliary input}}$. Moreover, if it is t-indistinguishable $\boxed{\text{with auxiliary input}}$ for any polynomial t, then it is called indistinguishable $\boxed{\text{with auxiliary input}}$.