Constructing a Pseudorandom Generator Requires an Almost Linear Number of Calls

Thomas Holenstein*
ETH Zurich
thomas.holenstein@inf.ethz.ch

Makrand Sinha‡
University of Washington
makrand@cs.washington.edu

***Abstract*—We show that a black-box construction of a pseudorandom generator from a one-way function needs to make $\Omega\left(\frac{n}{\log(n)}\right)$ calls to the underlying one-way function. The bound even holds if the one-way function is guaranteed to be regular. In this case it matches the best known construction due to Goldreich, Krawczyk, and Luby (SIAM J. Comp. 22, 1993), which uses $O\left(\frac{n}{\log(n)}\right)$ calls.**

***Keywords*-Pseudorandom Generators; One-way Functions; Black-box separation.**

I. Introduction

A. One-way functions and pseudorandom generators

Starting with the seminal works by Yao [1], and Blum and Micali [2], researchers have studied the relationship between various cryptographic primitives, such as one-way functions, pseudorandom generators, pseudorandom functions, and so on, producing a wide variety of results. One particular task which was achieved was the construction of pseudorandom generators from one-way functions, a task which has a history on its own. First, it was shown that one-way permutations imply pseudorandom generators [3], [4]. Later, the result was extended to regular one-way functions [5], and finally it was shown that arbitrary one-way functions imply pseudorandom generators [6].

Unfortunately, the constructions given in [5] and [6] are relatively inefficient (even though they run in polynomial time). Suppose we instantiate the construction given in [5] with a regular one-way functions taking n bits to n bits. Then, it yields a pseudorandom generator whose input is of length[1] $\widetilde{\Theta}(n^3)$ and calls the underlying one-way function $\widetilde{\Theta}(n)$ times. The parameters in [6] are worse: if we instantiate the construction with an (arbitrary) one-way function taking n bits to n bits, we obtain a pseudorandom generator which needs $\widetilde{\Theta}(n^8)$ bits of input, and which does around[2] $\widetilde{\Theta}(n^{12})$ calls to f. The parameters of the security reduction are also very weak.

Naturally, many papers improve the efficiency of these results: [7], [8] show that the result of [6] can be achieved with a more efficient reduction in case one assumes that the underlying one-way function has stronger security than the usual polynomial time security. [9] reduces the input length of the pseudorandom generator in [5] to $\Theta(n\log(n))$. Also it reduces the input length in [6] by a factor of $\widetilde{\Theta}(n)$, and the number of calls by a factor of $\widetilde{\Theta}(n^3)$. Most impressive, [10] reduces the seed length to $\widetilde{\Theta}(n^4)$ and the number of calls to $\widetilde{\Theta}(n^3)$, for the construction of a pseudorandom generator from an arbitrary one-way function. Finally, [11] reduce the seed length in this last construction to $\widetilde{\Theta}(n^3)$.

We remark that the main focus on the efficiency has been on reducing the seed length. This is reasonable, as (private) randomness is probably the most expensive resource.[3] Nevertheless, one would like both the seed length and the number of calls to be as small as possible.

B. Black-box separations

After [2], [1], it was natural to try and prove that one-way functions do imply seemingly stronger primitives, such as key agreement. However, all attempts in proving this failed, and so researchers probably wondered (for a short moment) whether in fact one-way functions *do not* imply key agreement. A moment of thought reveals that this is unlikely to be true: key-agreement schemes seem to exist, and so in fact we believe that—consider the following as a purely logical statement—one-way functions *do* imply key-agreement.

A way out of the dilemma was found by Impagliazzo and Rudich in a break through work [12]. They observed

*This work was supported by the Swiss National Science Foundation Grant No. 200021-132508

‡ Parts of this work was done while the author was a student at ETH Zurich. Work supported by the Excellence Scholarship and Opportunity Programme of the ETH Zurich Foundation.

[1]The $\widetilde{\Theta}$-notation ignores poly-logarithmic factors.

[2]We counted $\widetilde{\Theta}(n^{12})$ calls, but since [6] is not completely explicit about the construction, we make no guarantee that other interpretations are impossible…

[3]We would like to mention that in part this focus also seems to come from the (somewhat arbitrary) fact that people usually set the security parameter equal to the input length. For example, suppose we have a one-way function from n to n bits with security $2^{n/100}$ (meaning that in time $2^{n/100}$ one can invert f only with probability $2^{-n/100}$). If a construction now yields a pseudorandom generator with $m = n^2$ bits of input, the security can at most be $2^{\sqrt{m}/100}$. At this point it becomes tempting to argue that because $m \mapsto 2^{\sqrt{m}/100}$ is a much slower growing function than $n \mapsto 2^{n/100}$, it is crucial to make the input length as small as possible. However, if one introduces a security parameter k, both primitives could have security roughly 2^k. Arguing over the function which maps the input length to the security is not a priori a good idea.

that the proofs of most results such as "one-way functions imply pseudorandom generators" are, in fact, much stronger. In particular, the main technical part of [6] shows that there exists oracle algorithms $g^{(f)}$ and $A^{(\mathrm{Breaker},f)}$ with the following two properties:

- For any oracle, $g^{(f)}$ is an expanding function.
- For any two oracles $(\mathrm{Breaker}, f)$, if Breaker distinguishes the output of $g^{(f)}$ from a random string, then $A^{(\mathrm{Breaker},f)}$ inverts f.

Impagliazzo and Rudich then showed that the analogous statement for the implication "one-way functions imply key-agreement" is simply wrong, giving the first "black-box separation".

After the paper of Impagliazzo and Rudich, many more black box separations have been given (too many to list them all). We use techniques from several papers: in order to prove that there is no black-box construction of collision resistant hash-functions from one-way permutations, Simon [13] introduced the method of giving specific oracles which break the primitive to be constructed. Such oracles (usually called Breaker) are now widely used, including in this paper. Gennaro et al. [14] developed an "encoding paradigm", a technique which allows to give very strong black-box separations, even excluding non-uniform security reductions. This encoding paradigm has first been combined with a Breaker oracle in [15]. In [16] a slightly different extension of [13] is used: their technique analyzes how Breaker behaves in case one modifies the given one-way permutation on a single randomly chosen input. We also use this method.

Some black box separation results are (as we are) concerned with the efficiency of constructing pseudorandom generators. Among other things, Gennaro et al. [14] show that in order to get a pseudorandom generator which expands the input by t bits, a black-box construction needs to do at least $\Omega(t/\log(n))$ calls to the underlying one-way function (this matches the combination of Goldreich-Levin [4] with the extension given in Goldreich-Goldwasser-Micali [17]). In [18], Viola shows that in order for a black-box construction to expand the input by t bits, it needs to do at least one of (a) adaptive queries, (b) sequential computation, or (c) use $\Omega(t \cdot n \log(m))$ bits of input, when the underlying one-way function maps n to m bits. This result has been somewhat strengthened by Lu [19]. The papers [20], [21] both study how much the stretch of a given generator can be enlarged, as long as the queries to the given generator are non-adaptive.

C. Contributions of this paper

A natural question to ask is: "what is the minimum seed length and the minimal number of calls needed for a black-box construction of a pseudorandom generator from a one-way function?"

To the best of our knowledge, it is consistent with current knowledge that a construction has seed length $\Theta(n)$ and does a single call to the underlying one-way function (however, recall that [14] show that in order to get a stretch of t bits, at least $\Omega(t/\log(n))$ calls need to be made).

The reason why no stronger lower bounds are known seems to be that from a one-way *permutation* it is possible to get a pseudorandom generator very efficiently by the Goldreich-Levin theorem [4]: the input length only doubles, and the construction calls the underlying one-way permutation once. Also, almost all black-box separation results which prove that a primitive is unachievable from one-way functions also apply to one-way permutations. The only exceptions to this rule we are aware of is given by [22], [23] where it is shown that one-way permutations cannot be obtained from one-way functions, and [24], where this result is strengthened. However, both these results use a technique which does not seem to apply if one wants to give lower bounds on the efficiency of the construction of pseudorandom generators.[4]

One should note that a very efficient construction of a pseudorandom generator from a one-way function might have implications for practice: it is not inconceivable that in this case, practical symmetric encryption could be based on a one-way function, at least in some special cases where one would like a very high guarantee on the security.

We show in this paper than any construction must make at least $\Omega(\frac{n}{\log(n)})$ calls to the underlying one-way function. While this bound is interesting even for arbitrary one-way functions, it turns out that our proof works with some additional work even if the one-way function is guaranteed to be regular. In this case, the number of calls matches the parameters in [5] (and recall that the length of the seed has been reduced to $O(n\log(n))$ in [9], with the same number of calls to the one-way function).

In our theorem, we exclude a *fully black-box reduction*, using the terminology of [25]. In fact, we give three results.

In our first result, we assume that the construction $g(\cdot)$ when used with security parameter k *only calls the underlying one-way function with the same security parameter* k. We believe that this is a natural assumption, as all constructions we know have this property, and the underlying input length is not immediately defined if g makes calls to $f(k,\cdot)$ for various values of k. The result is stated in Theorem 5.

Next, we study black-box constructions with the same restriction, but where the security reduction is non-uniform. These can be handled with the technique from [14], and in our case it yields Theorem 7.

Finally, we remove the restriction that the construction calls the underlying function with a fixed security parameter. This gives Theorem 8. However, one needs to be careful somewhat, since in this case, the construction calls the given one-way function on a number of input lengths n, and thus

[4]Both proofs use the fact that a one-way permutation satisfies $g(v) \neq g(v')$ for any $v \neq v'$ crucially.

already the expression $\Omega(n/\log(n))$ in our lower bound needs to be specified more exactly. Our theorem uses *the shortest input length* of any call to f (*i.e.*, our lower bound is weakest possible in this case). Also, we remark that this last bound does not exclude the construction of "infinitely often pseudorandom generators", which are secure only for infinitely many security parameters.

II. The Main Theorem

We think of a one-way function as a family $\{f_k\}_{k\geq 0}$, indexed by some security parameter k. The function f_k then takes as input a bitstring of length $n(k)$, and outputs a bitstring of length $n'(k)$. Usually, the case $n(k) = n'(k) = k$ is considered in the literature. We want to distinguish n and k here, as we hope this makes the discussion clearer. However, we will still require that n is polynomially related to k.[5]

Definition 1. *A function $n(k) : \mathbb{N} \to \mathbb{N}$ is a* length function *if there exists $c \in \mathbb{N}$ such that $k^{1/c} \leq n(k) \leq k^c$, $n(k)$ can be computed in time k^c, and $n(k+1) \geq n(k)$ for any k.*

In general, the length $n(k)$ of the input of a one-way function differs from the length $n'(k)$ of the output. In case $n(k) > n'(k)$, it is shown in [26] how to obtain a "public-coin collection of one-way functions", where both the input and the output length are $n'(k)$. Such a collection can be used with known constructions to get a pseudorandom generator, and the number of calls will only depend on $n'(k)$. In case $n(k) < n'(k)$, it is easy to see that one can also get a"public-coin collection of one-way functions" with input and output length $2n(k)$.

Therefore, we can restrict ourselves to the case $n(k) = n'(k)$, and see that otherwise, the parameter $\min(n(k), n'(k))$ is the quantity of relevance to us.

Definition 2. *A* one-way function $f = \{f_k\}_{k\geq 0}$ *is a family of functions $f_k : \{0,1\}^{n(k)} \to \{0,1\}^{n(k)}$, computable in time* $\mathrm{poly}(k)$*, such that for any algorithm A running in time* $\mathrm{poly}(k)$ *the function mapping k to*

$$\Pr_{x,A}[A(k, f_k(x)) \text{ inverts } f_k] \tag{1}$$

is negligible in k.[6]

A pseudorandom generator $g = \{g_k\}_{k\geq 0}$ *is a family of polynomial time computable functions $g_k : \{0,1\}^{m(k)} \to \{0,1\}^{m'(k)}$ with $m'(k) > m(k)$ and such that any algorithm B running in time* $\mathrm{poly}(k)$

$$\Pr_{v,B}[B(k, g_k(v)) = 1] - \Pr_{w,B}[B(k, w) = 1] \tag{2}$$

is negligible in k.

We next define *fully black-box constructions*, but only for the special case of importance to us. Note that we assume that the underlying one way function is regular (a function family $\{f_k\}_{k\geq 0}$ is regular if $|\{x' : f_k(x') = f_k(x)\}|$ only depends on k and not on x).

Definition 3. *A* fully black-box construction of a pseudorandom generator from a regular one-way function *consists of two oracle algorithms (g, A). The* construction $g^{(f)}$ *is a polynomial time oracle algorithm which provides, for each length function $n(k)$ and each ℓ, a function $g_\ell : \{0,1\}^{m(\ell)} \to \{0,1\}^{m'(\ell)}$ with $m'(\ell) > m(\ell)$. For this, g may call f as an oracle.*

Further, the security reduction $A^{(\cdot,\cdot)}(k,\cdot,\cdot)$ *is a* $\mathrm{poly}(k, \frac{1}{\epsilon})$*-time oracle algorithm such that for any regular function f, any inverse polynomial function $\epsilon(\ell)$, and any oracle* Breaker *for which*

$$\begin{aligned}\Pr_{v,\text{Breaker}}&[\text{Breaker}(\ell, g_\ell(v)) = 1] - \\ &\Pr_{w,\text{Breaker}}[\text{Breaker}(\ell, w) = 1] \geq \epsilon(\ell)\end{aligned} \tag{3}$$

for infinitely many ℓ, then

$$\Pr_{x,A}[A^{(\text{Breaker},f)}(k, \epsilon(k), f_k(x)) \text{ inverts } f_k] \tag{4}$$

is non-negligible.

Due to space restrictions we will restrict ourselves to the (most interesting) case where g only calls f on a single security parameter and mention the other results only in passing.

Definition 4. *A black-box construction is* security parameter restricted *if $g(k,\cdot)$ only calls $f(k,\cdot)$ and $A(k,\cdot)$ only calls* $\text{Breaker}(k,\cdot)$ *and $f(k,\cdot)$ for any k.*

Our main contribution is the following theorem:

Theorem 5. *Let $n(k), r(k) \in \mathrm{poly}(k)$ be computable in time $\mathrm{poly}(k)$, and assume that $r(k) \in o(\frac{n(k)}{\log(n(k))})$. There exists no security parameter restricted fully black-box construction of a pseudorandom generator from a one-way function which has the property that $g(k,v)$ does at most $r(k)$ calls to $f(k,\cdot)$.*

The above discussion assumes that the adversary is uniform (*i.e.*, there is a single adversary $A^{(\cdot,\cdot)}$ with oracle access to f and Breaker). However, many black-box results even work in case that A can be a non-uniform circuit, and our result is no exception. We define non-uniform black-box constructions next, and then state our second theorem

[5]The requirement that $n(k) \leq k^c$ is implicit in the definition of one-way functions, as otherwise the one-way function cannot be evaluated in time polynomial in k. The requirement $n(k) \geq k^c$ is different, however. For example, suppose a family $\{f_k\}_{k\geq 0}$ can be evaluated in time $k^{O(1)}$ and has $n(k) = \log^2(n)$. Also, suppose that f_k is a one-way function in the sense that in time $k^{O(1)}$ it cannot be inverted with probability $k^{-O(1)} \leq 2^{-\sqrt{n(k)}}$. If f is additionally regular, fewer than $\Omega(n/\log(n))$ calls are sufficient to construct a pseudorandom generator.

[6]We say that "$A(f_k(x))$ inverts f_k" if $f_k(A(f_k(x))) = f_k(x)$, and write A below the symbol Pr to indicate that the probability is also over any randomness A may use. We also assume it is clear that x is picked from $\{0,1\}^{n(k)}$.

for non-uniform black-box adversaries (we also change the security of the one-way function from standard security to security $s(k)$ in order to illustrate what results we can get in this case).

Definition 6. *A* non-uniform fully black-box construction of a pseudorandom generator from a regular one-way function with security $s(k)$ *consists of two oracle algorithms (g, A). The* construction $g^{(f)}$ *is a polynomial time oracle algorithm which provides, for each k, a function $g_k : \{0,1\}^{m(k)} \to \{0,1\}^{m'(k)}$ with $m'(k) > m(k)$. For this, g_k may call f_k as an oracle, and $m(k), m'(k)$ may depend on $n(k)$ and $n'(k)$.*

Further, the security reduction $A^{(\cdot,\cdot)}(k,\cdot,\cdot)$ *is an oracle algorithm which does at most $s(k)$ queries, and has the property that for any regular function f and any oracle B for which*

$$\Pr_{v,B}[B(k, g_k(v)) = 1] - \Pr_{w,B}[B(k,w) = 1] \geq \frac{1}{100} \qquad (5)$$

for infinitely many k, there is $h_k \in \{0,1\}^{s(k)}$ such that

$$\Pr_{x,A}[A^{(B,f)}(k, h_k, f_k(x)) \text{ inverts } f_k] > \frac{1}{s(k)} \qquad (6)$$

for infinitely many k.

Similar to before, $A(k,\cdot,\cdot)$ only calls the oracles $f(k,\cdot)$ and $B(k,\cdot)$.

Theorem 7. *Let $r(k)$, $s(k)$, $n(k)$ be given, and assume $r(k) < \frac{n(k)}{1000 \log(s(k))}$ for infinitely many k. Then, there is no* non-uniform *security parameter restricted fully black-box construction of a pseudorandom generator from a one-way function with security s which has the property that $g(k,v)$ does at most $r(k)$ calls to $f(k,\cdot)$.*

We can also study what happens with black-box constructions which are not security parameter restricted. To explain our results in this setting, we need a few more definitions. Suppose we have given an oracle construction (g, A), and fix the oracle f (*i.e.*, the one-way function). For each ℓ we then consider the *shortest* call which $g(\ell, v)$ makes to f for any v:

$$n_f^-(\ell) := \min\{n(k) | \exists v : g^{(f)}(\ell, v) \text{ queries } f(k,\cdot)\}. \qquad (7)$$

Analogously, for each ℓ we consider the maximal number of calls $g(\ell, v)$ makes to f:

$$r_f(\ell) := \max\{r | \exists v : g^{(f)}(\ell, v) \text{ makes } r \text{ queries to } f\}. \qquad (8)$$

Note that both n_f^- and r_f do in general depend on the oracle f.

Our third main theorem is then given in the following:

Theorem 8. *Fix a length function $n(k)$. Let (g, A) be a fully black-box construction of a pseudorandom generator from a regular one-way function. Then, there is an oracle f for which*

$$r_f \in \Omega\Big(\frac{n_f^-}{\log(n_f^-)}\Big)\ . \qquad (9)$$

III. Notation and Conventions

In the rest of the paper, we consider one fixed security parameter $k = \ell$. Then, the input length $n = n(k)$ of the one-way function and the input length $m = m(k)$ of the pseudorandom generator are also fixed.

A. Pseudouniform functions

A pseudouniform function is a family $g = \{g_k\}_{k \geq 0}$ of length preserving functions $g_k : \{0,1\}^{m(k)} \to \{0,1\}^{m(k)}$ such that the output of g_k is indistinguishable from a uniform string. An example is given by the identity function, or any one-way permutation.

Definition 9. *A function family $g = \{g_k\}_{k\geq 0}$ where $g_k : \{0,1\}^{m(k)} \to \{0,1\}^{m(k)}$ of* $\text{poly}(k)$*-time computable functions is pseudouniform if, for all algorithms A running in time* $\text{poly}(k)$ *the function*

$$\left|\Pr_{A,v}[A(k, g_k(v)) = 1] - \Pr_{A,w}[A(k,w)] = 1\right| \qquad (10)$$

is negligible in k.

If we are given a family $\{g_k\}_{k\geq 0}$ which is both pseudouniform and a one-way function, then we can obtain a pseudorandom generator using only one call to g by the Goldreich-Levin Theorem [4]. Conversely, given a pseudorandom generator one can get a pseudouniform one-way function by truncating the output.

Theorem 10. *Suppose that $g = \{g_k\}$ is both a pseudouniform function and also a one-way function. Then, $h_k(v,z) := (g(v), z, \oplus_{i=1}^n v_i z_i)$ is a pseudorandom generator.*

Conversely, if g is a pseudorandom generator with $m(k)$ bits of input, the truncation of g to the first $m(k)$ bits of its output is both pseudouniform and a one-way function.

Thus, we see that giving lower bounds on the construction of pseudorandom generators is equivalent to giving lower bounds on the construction of pseudouniform one-way functions.

B. Normalization

Suppose we have a construction $\{g_k^{(f)}\}_{k\geq 0}$ of a supposedly pseudouniform one-way functions, where k is a security parameter. We make several assumptions on the construction which simplifies the proofs. First, we assume that g never calls f twice with the same input, and does exactly r calls to f. This is easy to achieve: one can modify g to get an equivalent oracle construction with these properties. Next, we enlarge the range of g, and assume that in case two queries of f give the same *answer*, then g outputs a special symbol which encodes a failure. This last restriction is not completely trivial, as it can break some constructions of pseudouniform functions for some choices of underlying one-way functions. However, because of the way we construct the oracles f_k, in our case this is no problem .

Definition 11. *Let* $\{0,1\}^{m*} := \{0,1\}^m \cup \{(\perp, v) | v \in \{0,1\}^m\}$. *An oracle function* $g^{(f)} : \{0,1\}^m \to \{0,1\}^{m*}$ *is* r*-query normalized if* $g(v)$ *never queries* f *with the same input twice, does exactly* r *calls to* f*, and whenever two outputs of* f *agree,* $g^{(f)}(v) = (\perp, v)$.

We will write g instead of $g^{(f)}$ whenever f is clear from the context. Furthermore, we let $g'(v, y_1, \ldots, y_r)$ be the function which never calls f but instead just uses y_i as the reply of f to the ith query.

C. Notations

Definition 12 (The Query-sets). *The set* $\mathrm{Query}(g, v, f)$ *is* $\{(x_1, y_1), \ldots, (x_r, y_r)\}$, *where* x_i *is the* i*-th query which* g *does to* f *in an evaluation of* $g^{(f)}(v)$*, and* y_i *is the answer given by* f*. The set* $\mathrm{Query}(g', v, y_1, \ldots, y_r))$ *is defined similarly (in particular, it also contains pairs* (x_i, y_i)*). The sets* $\mathrm{QueryX}(g, v, f)$ *and* $\mathrm{QueryY}(g, v, f)$ *contain the* x *and* y*-part of the pairs in* $\mathrm{Query}(g, v, f)$.

For a pair (x^*, y^*), we define

$$f_{(x^*,y^*)}(x) := \begin{cases} y^* & \text{if } x = x^* \\ f(x) & \text{otherwise.} \end{cases} \tag{11}$$

We use the following sets of functions $f : \{0,1\}^n \to \{0,1\}^n$. For a set $\mathcal{Y} \subseteq \{0,1\}^n$ such that $|\mathcal{Y}|$ divides 2^n, $\mathcal{F}(\mathcal{Y})$ is the set of all regular surjective functions $f : \{0,1\}^n \to \mathcal{Y}$. Then, $\mathcal{P}_n$ is the set of all bijective functions $f : \{0,1\}^n \to \{0,1\}^n$, *i.e.*, the permutations. We use $\mathcal{P}$ instead of $\mathcal{P}_n$ when n is clear from the context, and write $f \leftarrow \mathcal{P}_n$ or $f \leftarrow \mathcal{F}(\mathcal{Y})$ to pick a function uniformly from the respective set.

IV. Proofs for the case of Single Call

Basic setting: By Theorem 10, it is sufficient to consider constructions of pseudouniform one-way functions from one-way functions. Thus, suppose a fully black-box construction (g, A) of a pseudouniform one-way function is given. We fix some security parameter k, and consider $g(k, \cdot)$, which only calls $f(k, \cdot)$.

Our task is to come up with a pair $(\mathrm{Breaker}, f)$, such that $\mathrm{Breaker}(k, \cdot)$ either inverts g or distinguishes the output of g from a uniform random string, and yet $A^{(\mathrm{Breaker}, f)}$ will not invert $f(k, \cdot)$ with noticeable probability.

A. The case of a single call

We first study the case where $g^{(f)}$ makes a single call to the underlying one-way function f. To gather some intuition, let us first consider some example constructions which make $r = 1$ calls to f.

Let us take the function $g : \{0,1\}^n \to \{0,1\}^n$ which is defined as $g(v) = v$, so that the function simply outputs the input v. In this case, the function is clearly pseudouniform, therefore $\mathrm{Breaker}$ will need to break the one-way property of g. Consider the canonical breaker $\mathrm{BreakOW}$ presented below which inverts g using exhaustive search.[7]

```
Algorithm BreakOW^{(f)}(w)
for all $v \in \{0,1\}^m$ do
    if $g^{(f)}(v) = w$ then
        return $v$;
return $\perp$;
```

For the example g given above, $\mathrm{BreakOW}$ inverts $g^{(f)}(v)$ for all v and is independent of f, so it does not help in inverting f.

In general though, we want $\mathrm{BreakOW}$ to fulfill slightly opposing goals - we want it to powerful enough to be able to invert $g^{(f)}$ while at the same time it should not help with inverting f. $\mathrm{BreakOW}$ as presented above might help in inverting the function f for another g. To see this consider the example $g : \{0,1\}^{2n} \to \{0,1\}^{2n}$ defined as

$$g(v, r) := \begin{cases} (v, r) & \text{if } r \neq 0^n \\ (f(v), r) & \text{otherwise.} \end{cases} \tag{12}$$

This function is pseudouniform no matter how f is defined. Thus, $\mathrm{Breaker}$ needs to invert g as before. But it is easily seen that if $\mathrm{BreakOW}(y, 0^n)$ returns a preimage of g, clearly A will be able to invert f. Thus, only images (y, r) with $r \neq 0^n$ should be inverted. With this in mind consider the following iteration of $\mathrm{BreakOW}$ where we add a check to forbid such "malicious" queries.

To get some intuition on why the check in $\mathrm{SafeToAnswer}$ is the correct one, consider an adversary which is trying to find $f^{-1}(y)$ using $\mathrm{BreakOW}$. Assume for now that f is a permutation - we will later see that when taking the $\mathrm{Breaker}$ oracle to be $\mathrm{BreakOW}$ we will take f to be a permutation, so this case will suffice. The adversary has to generate a useful w, which can be used to query $\mathrm{BreakOW}$ and get some information about $f^{-1}(y)$. The easiest way the adversary can generate such a w is to pick a random permutation f' and a random input v' and hope that g is biased enough so that $g^{f'}(v') = w$. Moreover, since the adversary wants to obtain information about $f^{-1}(y)$ it may

[7]Note that for this particular example, we can invert $g(v)$ just by outputting v since g is just the identity function. However, this particular way of inverting g does not generalize to other constructions.

```
Algorithm BreakOW$^{(f)}(w)$
procedure SafeToAnswer$(w, Q)$:
// SafeToAnswer does not depend on $f$
  if $\Pr_{f' \leftarrow \mathcal{P}, v'}[g^{(f')}(v') = w] \geq 2^{-m+\frac{n}{30}}$ or
     $\Pr_{f' \leftarrow \mathcal{P}, v'}[\, g^{(f')}(v') = w \mid Q = \text{QueryY}(g, v', f')]$
     $\geq 2^{-m+\frac{n}{30}}$ then
    return false;
  return true;

for all $v \in \{0,1\}^m$ do
  if $g^{(f)}(v) = w$ then
    if SafeToAnswer$(w, \text{QueryY}(g, v, f))$ then
      return $v$;
return $\perp$;
```

as well choose f' conditioned that the answer of the f' query that $g(v')$ makes is y. We will later show that essentially these are the only ways when an adversary can exploit BreakOW: if it cannot do the above then it will not be able to invert a random permutation f.

We point out that adding the SafeToAnswer check has another side effect. Before adding the check BreakOW was too powerful so that it also helped in inverting the underlying function f. But after adding the check, for certain constructions g, it cannot be used to distinguish the output of g from uniform. For example, consider the trivial construction $g(v) = f(v)$. It cannot be inverted by BreakOW and rightfully so, since if f is a one-way function, then so is g. In these cases, we shall show later that we can break the pseudouniformity of g.

In any case consider the following quantity $p(g)$. This is the probability that BreakOW inverts $g(v)$ by returning v (actually, not quite: BreakOW might return a different preimage of $g(v)$ before it enumerates v – in any case, the probability that BreakOW inverts g is at least $p(g)$)

$$p(g) := \Pr_{\substack{f \leftarrow \mathcal{P} \\ v \leftarrow \{0,1\}^m}}[\text{SafeToAnswer}(g^{(f)}(v), \text{QueryY}(g, f, v))]$$

It is easy to see that if g satisfies $p(g) \geq \frac{1}{2}$, then BreakOW$^{(f)}$ will invert $g(v)$ with noticeable probability.

Lemma 13. *Let $g^{(\cdot)} : \{0,1\}^m \to \{0,1\}^m$ be a construction for a Pseudouniform one-way function. If $p(g) \geq \frac{1}{2}$, then*

$$\Pr_{f \leftarrow \mathcal{P}, v}[\text{BreakOW}(g^{(f)}(v))\ \textit{inverts}\ g^{(f)}] \geq \frac{1}{2}.$$

Our next goal is to show that SafeToAnswer indeed works: BreakOW is unlikely to help inverting f, when f is uniformly drawn from $\mathcal{P}$.

Lemma 14. *Let $g^{(\cdot)} : \{0,1\}^m \to \{0,1\}^m$ be an oracle construction of a pseudouniform one-way function. Let $A^{f,\text{BreakOW}}$ be an arbitrary algorithm making at most $2^{\frac{n}{20}}$ queries to f and to* BreakOW*. Then, the probability that A inverts $f(x)$ is at most*

$$\Pr_{f \leftarrow \mathcal{P}, x, A}[A^{f,\text{BreakOW}}(f(x))\ \textit{inverts}\ f] \leq 2^{-\frac{n}{30}}\ .$$

To prove the above lemma we employ the technique used by Haitner and Holenstein [16], which is a variation of the technique used by Simon [13]. The main idea is to study what happens if f is modified slightly by mapping a second, randomly chosen element x^* to y^* where y^* is the element that A is trying to invert. We show that such a change will likely go unnoticed by $A(y^*)$, and it will not find the new preimage. After the change, however, both preimages of y^* are equally likely to be the original one, so $A(y^*)$ could not have found the original one either.

To make the above notion more precise, let us fix now some permutation f, some $y^* \in \{0,1\}^n$ and some $w \in \{0,1\}^m$ and compare runs of BreakOW$^{(f)}(w)$ and BreakOW$^{(f_{(x^*,y^*)})}(w)$ for a random element $x^* \in \{0,1\}^n$ (Recall that $f_{(x^*,y^*)}$ is the same as f except on y^*). The next lemma then shows that the result of these two runs is equal with high probability.

Lemma 15 (BreakOW is robust)**.** *Fix f, y^*, w. Then*

$$\Pr_{x^*}[\text{BreakOW}^{(f)}(w) \neq \text{BreakOW}^{(f^*)}(w)] \leq 2^{-\frac{4n}{5}}$$

where $f^* = f_{(x^*,y^*)}$.

Once we have the above lemma, it readily implies Lemma 14. The proof can be found in the full version of this paper [27].

Next we prove Lemma 15. To simplify the notation, we introduce the following definition for a set $Q_{f,y^*,w}$. The set $Q_{f,y^*,w}$ represents one of the critical cases in which the execution of BreakOW$^{(f)}(w)$ and BreakOW$^{(f^*)}(w)$ can differ. For the case of $r = 1$ call, we will show by a counting argument that the set $Q_{f,y^*,w}$ is smaller than $2^{n/10}$ in size from which Lemma 15 will follow.

Definition 16. *Let $g^{(\cdot)} : \{0,1\}^m \to \{0,1\}^m$ be a 1-query oracle construction. For $f : \{0,1\}^n \to \{0,1\}^n$, $y^* \in \{0,1\}^m$, and $w \in \{0,1\}^m$, the set $Q_{f,y^*,w}$ contains all pairs (x^*, v^*) with the following properties:*

(a) $g^{(f^*)}(v^*) = w$
(b) $x^* \in \text{QueryX}(g, v^*, f^*)$, *i.e.,* $g^{(f^*)}(v^*)$ *queries* x^*
(c) SafeToAnswer$(w, \text{QueryY}(g, v^*, f^*))$,

where $f^* = f_{(x^*,y^*)}$.

The next lemma proves that for all f, y^* and w, the set $Q_{f,y^*,w}$ is small in case g is a 1-query construction.

Lemma 17. *Let $g^{(\cdot)} : \{0,1\}^m \to \{0,1\}^m$ be a 1-query oracle construction. Then for all $f \in \mathcal{P}, w \in \{0,1\}^m$ and*

$y^* \in \{0,1\}^n$,

$$|Q_{f,y^*,w}| \leq 2^{n/10}.$$

We will defer the proof of the above lemma for later and first prove how it implies Lemma 15.

Proof of Lemma 15: Let v be the result of $\mathrm{BreakOW}^{(f)}(w)$. The value $v^* = \mathrm{BreakOW}^{(f^*)}(w)$ can only be different from v in two cases: either v^* occurs in the enumeration of BreakOW *after* v (where we think of $\perp$ as the last value which occurs in the enumeration). Then x^* must be in $\mathrm{Query}(g, v, f)$. Alternatively, v^* occurs *before* v, then x^* must be in $Q_{f,y^*,w}$. Since the union of these two sets has fewer than $2^{\frac{n}{5}}$ elements the result follows. ■

Next we prove Lemma 17 which in the case of one call follows from a simple counting argument.

Proof of Lemma 17: For the sake of contradiction assume that $|Q_{f,y^*,w}| > 2^{n/10}$. Firstly note that for any pair (v^*, x^*), x^* is the query made by $g^{(\cdot)}(v^*)$ and hence is determined by v^*. Now pick a random $v^* \in \{0,1\}^m$. The probability that the corresponding pair (v^*, x^*) belongs to $Q_{f,y^*,w}$ is at least $2^{-m+\frac{n}{10}}$. Now consider the evaluation of $g^{(f^*)}(v^*)$. Note that in the evaluation of $g^{(f^*)}(v^*)$, the answer of the query x^* made by g is fixed to be y^*. Since the probability that $(v^*, x^*) \in Q_{f,y^*,w}$ is at least $2^{-m+\frac{n}{10}}$, it follows that,

$$\Pr_v[g^{(f^*)}(v^*) = w \mid f^*(x^*) = y^*] > 2^{-m+\frac{n}{10}}.$$

The last inequality implies that $\mathrm{SafeToAnswer}(w, \mathrm{QueryY}(g, v^*, f^*))$ is false which violates the condition (c) in the definition of $Q_{f,y^*,w}$. ■

B. Breaking the Pseudouniformity

As pointed out in the last subsection, Oracle BreakOW described above works well in the case $p(g) \geq \frac{1}{2}$, but might not be powerful enough to distinguish the output of g from uniform for certain constructions g, such as the example discussed before $g^{(f)}(v) = f(v)$. For this construction, the problem arises since we cannot hope to invert g without inverting f. We will show that in cases such as these (when $p(g) \leq 1/2$), we can break the pseudouniformity of g.

Let us take a closer look at what the condition $p(g) \leq 1/2$ implies. In this case, once the output of the query to f made by g is fixed to be y, certain values w are much more likely (if v is still chosen at random). Thus, it is not too far fetched to hope that if f is a very degenerate random function $f : \{0,1\}^n \to \mathcal{Y}$ for some set $\mathcal{Y} \subseteq \{0,1\}^n$ which is small, then often $g^{(f)}(v)$ will be one of few possible values in a set $W(\mathcal{Y})$ which depends only on $\mathcal{Y}$. We can then distinguish the output of g from uniform by just verifying membership in the set $W(\mathcal{Y})$ and we can do this without even knowing the details of f (namely, we can still pick $f : \{0,1\}^n \to \mathcal{Y}$ uniformly at random) and hence this should not help in inverting f.

Next we formalize the above intuition. The following lemma proves that such a set exists $W(\mathcal{Y})$ exists when $\mathcal{Y}$ is chosen to be very small. How small do we want $\mathcal{Y}$ to be ? Note that for each fixed y, the number of w that can fail the second condition in SafeToAnswer (namely the probability considered in SafeToAnswer is at least $2^{-m+\frac{n}{30}}$) can be at most $2^{m-\frac{n}{30}}$. Hence, if there are $\ll 2^{n/30}$ elements in $\mathcal{Y}$, W can be of size at $|\mathcal{Y}| \cdot 2^{m-\frac{n}{30}}$ which is much smaller than 2^n and we would be in good shape.

Lemma 18. *Let $g^{(\cdot)} : \{0,1\}^m \to \{0,1\}^m$ be a 1-query normalized oracle construction with $p(g) \leq \frac{1}{2}$, $\frac{n}{100} \in \mathbb{N}$. There exists $\mathcal{Y} \subseteq \{0,1\}^n$ of size $|\mathcal{Y}| = 2^{\frac{n}{100}}$ and a set $W \subseteq \{0,1\}^m$ of size $|W| \leq 2^{m-\frac{n}{100}}$ such that*

$$\Pr_{\substack{f \leftarrow \mathcal{F}(\mathcal{Y}) \\ v \leftarrow \{0,1\}^m}}[g^{(f)}(v) \in W] \geq \frac{1}{2}. \tag{13}$$

The full proof is omitted here, but the basic idea is to pick $\mathcal{Y}$ uniformly at random, and let W be the elements w for which $\neg\mathrm{SafeToAnswer}(w, Q)$ can happen for some $Q \subseteq \mathcal{Y}$. A simple counting argument shows that there cannot be many elements w for which this is possible. Furthermore, up to minor terms, the probability in (13) corresponds to $1 - p(g)$.

Let g be such that $p(g) \leq 1/2$. Fix the sets $\mathcal{Y}$ and W and consider the following oracle BreakPU for breaking the pseudouniformity of g. Note that the sets $\mathcal{Y}$ and W might depend on g.

```
Algorithm BreakPU^W(w)
if $w \in W$ then
    return 1;
return 0
```

Since Lemma 18 gives us that $|W| \leq 2^{m-n/100}$, the probability that a random $w \in \{0,1\}^m$ satisfies $w \in W$ is at most $2^{-\frac{n}{100}}$. At the same time the probability that $g^{(f)}(v) \in W$ for random v and $f \leftarrow \mathcal{F}(\mathcal{Y})$ is at least $1/2$. It follows that BreakPU breaks the pseudouniformity of g.

All that remains to prove is that BreakPU does not help significantly in inverting a uniformly random function $f : \{0,1\}^n \to \mathcal{Y}$. This is intuitive, since BreakPU does not even depend on f (besides the choice of $\mathcal{Y}$). The next lemma gives a formal statement. We remark that this lemma also follows directly from [14, Theorem 1]. To see this, note that we can pick f as follows: first pick a random regular function $p : \{0,1\}^n \to \mathcal{Y}$ and then set $f = \pi \circ p$ for some permutation π; by [14, Theorem 1], f is $2^{|\mathcal{Y}|^{(1/5)}}$-hard to invert even given p.

Lemma 19. *Let A be an arbitrary oracle algorithm making at most $2^{\frac{n}{1000}}$ queries, $|\mathcal{Y}| = 2^{\frac{n}{100}}$, $\frac{n}{1000} \in \mathbb{N}$. Then,*

$$\Pr_{f \leftarrow \mathcal{F}(\mathcal{Y}), x, A}[A^{f,\mathrm{BreakPU}}(f(x)) \text{ inverts } f] \leq 2^{-\frac{n}{1000}}\ , \tag{14}$$

where $\mathrm{BreakPU} = \mathrm{BreakPU}^W$ *for an arbitrary set* W.

With Lemmas 13, 14, 18 and 19 in hand it is not too difficult to prove the following theorem.

Theorem 20. *Let* $n(k) \in \mathrm{poly}(k)$ *be computable in time* $\mathrm{poly}(k)$. *There exists no security parameter restricted fully black-box construction of a pseudorandom generator from a one-way function which has the property that* $g(k, v)$ *makes at most* 1 *call to* $f(k, \cdot)$.

V. The Case of $r > 1$ calls

For the case of $r > 1$ calls, the basic idea of the proof remains the same as before. We will give two oracles, each of which breaks one of the two security properties of g. The first oracle inverts g with noticeable probability, and the second oracle distinguishes the output of g from a uniform random string. For each security parameter k we will then set $\mathrm{Breaker}_k$ to be one of these two oracles, depending on the combinatorial structure of g_k.

A. The inverting oracle

The first oracle is a generalization of BreakOW defined before. It inverts g in some cases, and is given as algorithm below, but we first explain it informally. On input $w \in \{0,1\}^m$, $\mathrm{BreakOW}(w)$ first enumerates all possible inputs $v \in \{0,1\}^m$ of g in lexicographic order. For each of them it checks whether $g^{(f)}(v) = w$. If so, it checks whether returning v could help some algorithm A to invert f. For this, it calls the procedure SafeToAnswer. Roughly speaking, SafeToAnswer will return false in case this fixed w correlates strongly with some outputs $y \in \{0,1\}^n$ of f which occurred during the evaluation of $g^{(f)}(v)$. More exactly, SafeToAnswer enumerates all possible subsets B of the answers f gave in the evaluation of $g^{(f)}(v)$. It then computes the probability that an evaluation outputs w, conditioned on the event that the evaluation produces all outputs in B. If this probability is much larger than 2^{-m}, SafeToAnswer will return false.

```
Algorithm BreakOW^{(f)}(w)
procedure SafeToAnswer($w, Q$):
  for all $B \subseteq Q$ do
    if
      $\Pr_{f' \leftarrow \mathcal{P}, v'} [\, g^{(f')}(v') = w \mid B \subseteq \mathrm{QueryY}(g, v', f')]$
        $\geq 2^{-m+\frac{n}{30}}$ then
        return false;
  return true;

for all $v \in \{0,1\}^m$ do
    if $g^{(f)}(v) = w$ then
        if SafeToAnswer($w, \mathrm{QueryY}(g, v, f)$) then
            return $v$;
return $\perp$;
```

We next define the quantity $p(g)$. This is a lower bound on the probability that BreakOW inverts $g(v)$ by returning v

$$p(g) := \Pr_{\substack{f \leftarrow \mathcal{P} \\ v \leftarrow \{0,1\}^m}} [\mathrm{SafeToAnswer}(g^{(f)}(v), \mathrm{QueryY}(g, f, v))] \tag{15}$$

It is easy to see that in case $p(g) \geq \frac{1}{2}$, then $\mathrm{BreakOW}^{(f)}$ will invert $g(v)$ with noticeable probability.

Lemma 21. *Let* $g^{(\cdot)} : \{0,1\}^m \to \{0,1\}^{m*}$ *be a normalized oracle construction. If* $p(g) \geq \frac{1}{2}$, *then*

$$\Pr_{f \leftarrow \mathcal{P}, v} [\mathrm{BreakOW}(g^{(f)}(v))) \textit{ inverts } g^{(f)}] \geq \frac{1}{2}. \tag{16}$$

We next claim that BreakOW is unlikely to help inverting f, when is uniformly drawn from $\mathcal{P}$.

Lemma 22. *Let* $g^{(\cdot)} : \{0,1\}^m \to \{0,1\}^{m*}$ *be an* r*-query normalized oracle construction,* $r < \frac{n}{100 \log(2n+m)}$. *Let* $A^{f, \mathrm{BreakOW}}$ *be an arbitrary algorithm making at most* $2^{\frac{n}{20}}$ *queries to* f *and to* BreakOW. *Then, the probability that* A *inverts* $f(x)$ *is at most*

$$\Pr_{f \leftarrow \mathcal{P}, x, A} [A^{f, \mathrm{BreakOW}}(f(x)) \textit{ inverts } f] \leq 2^{-\frac{n}{30}} . \tag{17}$$

To prove the above lemma recall the proof of Lemma 14. We fix some permutation f, some $y^* \in \{0,1\}^n$ and some $w \in \{0,1\}^m$ and compare runs of $\mathrm{BreakOW}^{(f)}(w)$ and $\mathrm{BreakOW}^{(f_{(x^*,y^*)})}(w)$ for a random element $x^* \in \{0,1\}^n$. The next lemma shows that the result of these two runs is equal with high probability in case $|Q_{f,y^*,w}|$ is small.

Lemma 23. *Fix* f, y^*, w. *If* $|Q_{f,y^*,w}| \leq 2^{\frac{n}{10}}$, *then*

$$\Pr_{x^*} [\mathrm{BreakOW}^{(f)}(w) \neq \mathrm{BreakOW}^{(f^*)}(w)] \leq 2^{-\frac{4n}{5}} \tag{18}$$

where $f^* = f_{(x^*,y^*)}$.

The next Lemma claims that with high probability over the choice of f, the set $Q_{f,y^*,w}$ is small as in the case of two calls. In the case, of one call we could prove that $Q_{f,y^*,w}$ is small for *every* permutation f. For r larger than 1, we can only prove this *with high probability* over $f \leftarrow \mathcal{P}$ and to prove the above lemma a concentration bound for polynomials in the style as proven by [28] seems to be needed. We will use a bound from [29], and show in Section VI how it can be used to prove the next Lemma. It turns out that this concentration bound also breaks down if $r \in \Omega(n/\log(n))$.

Lemma 24 (Concentration Lemma)**.** *Let* $g^{(\cdot)} : \{0,1\}^m \to \{0,1\}^{m*}$ *be an* r*-query normalized oracle construction,* $r \leq \frac{n}{100 \log(n)}$. *For all* (w, y^*) *we have*

$$\Pr_{f \leftarrow \mathcal{P}} \left[|Q_{f,w,y^*}| > 2^{\frac{n}{10}} \right] < 2^{-2^{\frac{n}{100r}}} . \tag{19}$$

B. The distinguishing oracle

Oracle BreakOW described above works well in case $p(g) \geq \frac{1}{2}$. Therefore, we now concentrate on the case $p(g) \leq \frac{1}{2}$. In this case, there are elements $y_1, \ldots, y_b$ such that conditioned on those occurring as outputs of f, some elements w are much more likely than others (in fact, on a random evaluation we have probability at least $\frac{1}{2}$ that a subset of the y's produced satisfies this). Thus, if f is a function $f : \{0,1\}^n \to \mathcal{Y}$ for some set $\mathcal{Y} \subseteq \{0,1\}^n$ which is small, then often $g^{(f)}(v)$ will be one of few possible values. Formally, we can prove Lemma 25 in the same spirit as Lemma 18.

Remark. It turns out that if BreakOW inverts $g(v)$ with low probability, we can choose $\mathcal{Y} \subseteq \{0,1\}^n$ as small as $2^{\Theta(n/r)}$, and conditioned on f being from $\mathcal{F}(\mathcal{Y})$, the output of g is very biased. Since $\mathcal{Y}$ is superpolynomial only as long as $r \in o(n/\log(n))$, we see that f stops being a one-way function once $r \notin o(n/\log(n))$.

Lemma 25. *Let $g^{(\cdot)} : \{0,1\}^m \to \{0,1\}^{m*}$ be an r-query normalized oracle construction with $p(g) \leq \frac{1}{2}$, $\frac{n}{1000r} \in \mathbb{N}$. There exists $\mathcal{Y} \subseteq \{0,1\}^n$ of size $|\mathcal{Y}| = 2^{\frac{n}{100r}}$ and a set $W \subseteq \{0,1\}^m$ of size $|W| \leq 2^{m - \frac{n}{100}}$ such that*

$$\Pr_{\substack{f \leftarrow \mathcal{F}(\mathcal{Y}) \\ v \leftarrow \{0,1\}^m}}[g^{(f)}(v) \in W] \geq \frac{1}{2} - r^2 2^{-\frac{n}{100r}} \qquad (20)$$

Let now BreakPU^W be the oracle which on input w returns 1 if and only if $w \in W$. The next lemma states that $\text{BreakPU}(W)$ does not help significantly in inverting f.

Lemma 26. *Let A be an arbitrary oracle algorithm making at most $2^{\frac{n}{1000r}}$ queries, $|\mathcal{Y}| = 2^{\frac{n}{100r}}$, $\frac{n}{1000r} \in \mathbb{N}$. Then,*

$$\Pr_{f \leftarrow \mathcal{F}(\mathcal{Y}), x, A}[A^{f, \text{BreakPU}}(f(x)) \textit{ inverts } f] \leq 2^{-\frac{n}{1000r}} , \quad (21)$$

where $\text{BreakPU} = \text{BreakPU}^W$ *for an arbitrary set* W.

C. Proving the main result

The above lemmas can be used to prove Theorem 5. For space reasons we omit it here; however, this should be pretty obvious. The only issues are translating it to the asymptotic version, the normalization, and then handling pseudorandom generators via the connection to pseudouniform one-way functions. These are, however, all minor.

VI. Proof of Concentration Lemma

Due to space restrictions we can not include the proof of Lemma 24, but in this section we provide some intuition about the proof.

Fix (f, y^*, w), and assume that $(x^*, v^*) \in Q_{f,y^*,w}$. Consider the query-answer pairs $\{(x_1, y_1), \ldots, (x_r, y_r)\} = \text{Query}(g, v^*, f_{(x^*,y^*)})$ which occur in an evaluation of $g^{(f_{(x^*,y^*)})}(v^*)$. The pair (x^*, y^*) must be in this set, as otherwise conditions (a) or (b) of Definition 16 would not hold, and to simplify the discussion we make the (unrealistic) assumption that always $(x^*, y^*) = (x_r, y_r)$. Now consider the set $T = \{(x_1, y_1), \ldots, (x_{r-1}, y_{r-1})\}$. Let us call T an *incrementor* for $|Q_{f,y^*,w}|$, because whenever f satisfies $f(x_i) = y_i$ for $i \in \{1, \ldots, r-1\}$, the set $Q_{f,y^*,w}$ grows by 1.[8]

Now, still fixing (f, y^*, w), the total number of such "incrementors" for $|Q_{f,y^*,w}|$ is at most $2^{(r-1)n + \frac{n}{30}}$. To see this, we argue that otherwise, (for y_r being the answer of the r-th query in the evaluation)

$$\Pr_{f' \leftarrow \mathcal{P}, v'}[g^{f'}(v') = w | y_r = y^*] \geq 2^{-m + \frac{n}{30}} , \qquad (22)$$

because any of the incrementors survive[9] the picking of f with probability roughly[10] $2^{-(r-1)n}$. Thus, if there are $2^{(r-1)n + \frac{n}{30}}$ incrementors, in expectation $2^{\frac{n}{30}}$ will survive the picking of f, and if we pick one[11] of the $2^{\frac{n}{30}}$ values v^* which survived we get an element for which $g^{f'}(v') = w$ (conditioning on $y_r = y^*$). Now, (22) roughly contradicts $\text{SafeToAnswer}(w, Q)$ for $B = \{y^*\}$ (up to some issues due to our simplifying assumption that (x^*, y^*) is always (x_r, y_r), but since $r^r < 2^n$ they do not matter much).

Thus, there are at most $2^{(r-1)n + \frac{n}{30}}$ incrementors for $|Q_{f,y^*,w}|$, and so in expectation $|Q_{f,w,y^*}| \leq 2^{\frac{n}{30}}$. However, we need to prove that the $|Q_{f,w,y^*}|$ is small with (very) high probability, and not in expectation. Luckily for us, Kim and Vu [28] proved a concentration bound which can be applied in our setting – translated to our setting, they show that concentration *does* hold if several conditions are given. First, it needs to hold that all probabilities checked in SafeToAnswer are smaller than $2^{-m + \frac{n}{30}}$ (which is, besides Lemma 25, the reason that SafeToAnswer is defined in the way it is defined). Second, they roughly require that $r^r < 2^n$, which holds in our case, because we assume that $r \notin \Omega(\frac{n}{\log(n)})$. Finally, they require that the events $f(x_1) = y_1$ and $f(x_2) = y_2$ are independent—which of course is a problem, because this does not hold in our case. Luckily, it turns out that this last requirement can be relaxed somewhat using a proof technique implicit in [30] (see [31], [32]). A proof of a Kim-Vu style concentration bound in this form was given by the first author in [29] and we use it to prove Lemma 24. The full proof can be found in the full version of the paper [27].

VII. Acknowledgements

We thank Colin Zheng for pointing out a mistake in an earlier version of this paper.

[8]Ignoring a few reasons why this might not be true sometimes... like the fact that SafeToAnswer might return false.

[9]Formally, surviving means that $f(x_i) = y_i$ for all pairs (x_i, y_i) in the incrementor.

[10]Ignoring very slight dependence in this discussion which arises from the fact that f is picked as a permutation.

[11]Only one incrementor with a fixed v^* can survive with our assumptions.

References

[1] A. C. Yao, "Theory and applications of trapdoor functions (extended abstract)," in *The 23rd Annual Symposium on Foundations of Computer Science*, 1982, pp. 80–91.

[2] M. Blum and S. Micali, "How to generate cryptographically strong sequences of pseudo-random bits," *SIAM Journal on Computing*, vol. 13, no. 4, pp. 850–864, 1984.

[3] L. A. Levin, "One-way functions and pseudorandom generators," *Combinatorica*, vol. 7, no. 4, pp. 357–363, 1987.

[4] O. Goldreich and L. A. Levin, "A hard-core predicate for all one-way functions," in *Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing*, 1989, pp. 25–32.

[5] O. Goldreich, H. Krawczyk, and M. Luby, "On the existence of pseudorandom generators," *SIAM Journal on Computing*, vol. 22, no. 6, pp. 1163–1175, 1993.

[6] J. Håstad, R. Impagliazzo, L. A. Levin, and M. Luby, "A pseudorandom generator from any one-way function," *SIAM Journal on Computing*, vol. 28, no. 4, pp. 1364–1396, 1999.

[7] I. Haitner, D. Harnik, and O. Reingold, "Efficient pseudorandom generators from exponentially hard one-way functions," in *ICALP (2)*, 2006, pp. 228–239.

[8] T. Holenstein, "Pseudorandom generators from one-way functions: A simple construction for any hardness," in *TCC 2006*, 2006, pp. 443–461.

[9] I. Haitner, D. Harnik, and O. Reingold, "On the power of the randomized iterate," in *Advances in Cryptology — CRYPTO 2006*, ser. Lecture Notes in Computer Science, C. Dwork, Ed., vol. 4117, 2006.

[10] I. Haitner, O. Reingold, and S. Vadhan, "Efficiency improvements in constructing pseudorandom generators from one-way functions," in *Proceedings of the Forty-Second Annual ACM Symposium on Theory of Computing*, 2010.

[11] S. P. Vadhan and C. J. Zheng, "Characterizing pseudoentropy and simplifying pseudorandom generator constructions," in *STOC*, 2012, pp. 817–836.

[12] R. Impagliazzo and S. Rudich, "Limits on the provable consequences of one-way permutations," in *Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing*, 1989, pp. 44–61.

[13] D. R. Simon, "Finding collisions on a one-way street: Can secure hash functions be based on general assumptions?" in *Advances in Cryptology — EUROCRYPT '98*, ser. Lecture Notes in Computer Science, K. Nyberg, Ed., vol. 1403, 1998, pp. 334–345.

[14] R. Gennaro, Y. Gertner, J. Katz, and L. Trevisan, "Bounds on the efficiency of generic cryptographic constructions," *SIAM Journal of Computing*, vol. 35, no. 1, pp. 217–246, 2005.

[15] I. Haitner, J. J. Hoch, O. Reingold, and G. Segev, "Finding collisions in interactive protocols – a tight lower bound on the round complexity of statistically-hiding commitments," in *The 48th Annual Symposium on Foundations of Computer Science*, 2007, pp. 669–679.

[16] I. Haitner and T. Holenstein, "On the (im)possibility of key dependent encryption," in *TCC 2009*, 2009, pp. 202–219.

[17] O. Goldreich, S. Goldwasser, and S. Micali, "How to construct random functions," *Journal of the ACM*, vol. 33, no. 4, pp. 792–807, 1986.

[18] E. Viola, "On constructing parallel pseudorandom generators from one-way functions," in *IEEE Conference on Computational Complexity*, 2005, pp. 183–197.

[19] C.-J. Lu, "On the complexity of parallel hardness amplification for one-way functions," in *Proceedings of the Third conference on Theory of Cryptography*, ser. TCC'06, 2006, pp. 462–481.

[20] J. Bronson, A. Juma, and P. A. Papakonstantinou, "Limits on the stretch of non-adaptive constructions of pseudo-random generators," in *TCC*, 2011, pp. 504–521.

[21] E. Miles and E. Viola, "On the complexity of non-adaptively increasing the stretch of pseudorandom generators," in *TCC*, 2011, pp. 522–539.

[22] S. Rudich, "Limits on the provable consequences of one-way functions," Tech. Rep., 1988.

[23] J. Kahn, M. Saks, and C. Smyth, "The dual bkr inequality and rudich's conjecture," *Combinatorics Probability and Computing*, vol. 20, no. 2, pp. 257–266.

[24] T. Matsuda and K. Matsuura, "On black-box separations among injective one-way functions," in *TCC*, 2011, pp. 597–614.

[25] O. Reingold, L. Trevisan, and S. P. Vadhan, "Notions of reducibility between cryptographic primitives," in *TCC 2004*, ser. Lecture Notes in Computer Science, M. Naor, Ed., vol. 2951, 2004, pp. 1–20.

[26] N. Dedic, D. Harnik, and L. Reyzin, "Saving private randomness in one-way functions and pseudorandom generators," in *TCC*, 2008, pp. 607–625.

[27] T. Holenstein and M. Sinha, "Constructing a pseudorandom generator requires an almost linear number of calls," *CoRR*, vol. abs/1205.4576, 2012.

[28] J. H. Kim and V. H. Vu, "Concentration of multivariate polynomials and its applications." *Combinatorica*, vol. 20, no. 3, pp. 417–434, 2000.

[29] T. Holenstein, "Some concentration bounds," 2011, manuscript.

[30] J. P. Schmidt, A. Siegel, and A. Srinivasan, "Chernoff-hoeffding bounds for applications with limited independence," *SIAM J. Discrete Math.*, vol. 8, no. 2, pp. 223–250, 1995.

[31] A. Rao, "Parallel repetition in projection games and a concentration bound," in *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, 2008, pp. 1–10.

[32] R. Impagliazzo and V. Kabanets, "Constructive proofs of concentration bounds," in *APPROX-RANDOM*, 2010, pp. 617–631.