

Constructing a Pseudorandom Generator Requires an Almost Linear Number of Calls

Thomas Holenstein*

Makrand Sinha†

May 22, 2012

Abstract

We show that a black-box construction of a pseudorandom generator from a one-way function needs to make $\Omega(\frac{n}{\log(n)})$ 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 $O(\frac{n}{\log(n)})$ calls.

1 Introduction

1.1 One-way functions and pseudorandom generators

Starting with the seminal works by Yao [Yao82], and Blum and Micali [BM84], 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 [Lev87, GL89]. Later, the result was extended to regular one-way functions [GKL93], and finally it was shown that arbitrary one-way functions imply pseudorandom generators [HILL99].

Unfortunately, the constructions given in [GKL93] and [HILL99] are relatively inefficient (even though they run in polynomial time). Suppose we instantiate the construction given in [GKL93] with a regular one-way functions taking n bits to n bits. Then, it yields a pseudorandom generator whose input is of length¹ $\tilde{\Theta}(n^3)$ and calls the underlying one-way function $\tilde{\Theta}(n)$ times. The parameters in [HILL99] 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 $\tilde{\Theta}(n^8)$ bits of input, and which does around² $\tilde{\Theta}(n^{12})$ calls to f . The parameters of the security reduction are also very weak.

Naturally, many papers improve the efficiency of these results: [HHR06a, Hol06] show that the result of [HILL99] can be achieved with a more efficient reduction in case one assumes that

*ETH Zurich, thomas.holenstein@inf.ethz.ch. This work was supported by the Swiss National Science Foundation Grant No. 200021-132508

†University of Washington, makrand@cs.washington.edu. 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.

¹The $\tilde{\Theta}$ -notation ignores poly-logarithmic factors.

²We counted $\tilde{\Theta}(n^{12})$ calls, but since [HILL99] is not completely explicit about the construction, we make no guarantee that other interpretations are impossible...

the underlying one-way function has stronger security than the usual polynomial time security. [HHR06b] reduces the input length of the pseudorandom generator in [GKL93] to $\Theta(n \log(n))$. Also it reduces the input length in [HILL99] by a factor of $\tilde{\Theta}(n)$, and the number of calls by a factor of $\tilde{\Theta}(n^3)$. Most impressive, [HRV10] reduces the seed length to $\tilde{\Theta}(n^4)$ and the number of calls to $\tilde{\Theta}(n^3)$, for the construction of a pseudorandom generator from an arbitrary one-way function. Finally, [VZ12] reduce the seed length in this last construction to $\tilde{\Theta}(n^3)$.

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.

1.2 Black-box separations

After [BM84, Yao82], 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 [IR89]. They observed 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 [HILL99] shows that there exists oracle algorithms $g^{(f)}$ and $A^{(\text{Breaker}, f)}$ with the following two properties:

- For any oracle, $g^{(f)}$ is an expanding function.
- For any two oracles $(\text{Breaker}, f)$, if Breaker distinguishes the output of $g^{(f)}$ from a random string, then $A^{(\text{Breaker}, f)}$ 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 [Sim98] 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. [GGKT05] 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 [HHRS07]. In [HH09] a slightly different extension of [Sim98] 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. [GGKT05] show that in order to

³We 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.

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 [GL89] with the extension given in Goldreich-Goldwasser-Micali [GGM86]). In [Vio05], 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 [Lu06]. The papers [BJP11, MV11] 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.

1.3 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 [GGKT05] 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 [GL89]: 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 [Rud88, KSS11] where it is shown that one-way permutations cannot be obtained from one-way functions, and [MM11], 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 that 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 [GKL93] (and recall that the length of the seed has been reduced to $O(n \log(n))$ in [HHR06b], 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 [RTV04]. 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 [GGKT05], and in our case it yields Theorem 6.

Finally, we remove the restriction that the construction calls the underlying function with a fixed security parameter. This gives Theorem 7. However, one needs to be careful somewhat, since

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

in this case, the construction calls the given one-way function on a number of input lengths n , and thus 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.

2 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 $n(k) : \mathbb{N} \rightarrow \mathbb{N}$ 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 [DHR08] 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 $f_k : \{0, 1\}^{n(k)} \rightarrow \{0, 1\}^{n(k)}$, computable in time $\text{poly}(k)$, such that for any algorithm A running in time $\text{poly}(k)$ 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 $g_k : \{0, 1\}^{m(k)} \rightarrow \{0, 1\}^{m'(k)}$ with $m'(k) > m(k)$ and such that any algorithm B running in time $\text{poly}(k)$

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

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).

⁵The 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.

⁶We 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 $\{0, 1\}^{n(k)}$.

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 $g_\ell : \{0, 1\}^{m(\ell)} \rightarrow \{0, 1\}^{m'(\ell)}$ 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 $\text{poly}(k, \frac{1}{\epsilon})$ -time oracle algorithm such that for any regular function f , any inverse polynomial function $\epsilon(\ell)$, and any oracle Breaker for which

$$\Pr_{v, \text{Breaker}} [\text{Breaker}(\ell, g_\ell(v)) = 1] - \Pr_{w, \text{Breaker}} [\text{Breaker}(\ell, w) = 1] \geq \epsilon(\ell) \quad (3)$$

for infinitely many ℓ , then

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

is non-negligible.

In a large part of the paper we restrict ourselves to the (most interesting) case where g only calls f on a single security parameter.

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 Breaker(k, \cdot) and $f(k, \cdot)$ for any k .

Our main contribution is the following theorem:

Theorem 5. Let $n(k), r(k) \in \text{poly}(k)$ be computable in time $\text{poly}(k)$, 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 in Section 7, and then prove the following theorem (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).

Theorem 6. 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 $f(k, \cdot)$.

In Section 8 we 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 :

$$n_f^-(\ell) := \min\{n(k) | \exists v : g^{(f)}(\ell, v) \text{ queries } f(k, \cdot)\}. \quad (5)$$

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

$$r_f(\ell) := \max\{r | \exists v : g^{(f)}(\ell, v) \text{ makes } r \text{ queries to } f\}. \quad (6)$$

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

Our second main theorem is then given in the following:

Theorem 7. Fix a length function $n(k)$. Let (g, A) be a fully black-box construction of a pseudo-random generator from a regular one-way function. Then, there is an oracle f for which

$$r_f \in \Omega\left(\frac{n_f^-}{\log(n_f^-)}\right). \quad (7)$$

3 Notation and Conventions

In most 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.

3.1 Pseudouniform functions

A pseudouniform function is a family $g = \{g_k\}_{k \geq 0}$ of length preserving functions $g_k : \{0, 1\}^{m(k)} \rightarrow \{0, 1\}^{m(k)}$ 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 8. A function family $g = \{g_k\}_{k \geq 0}$ where $g_k : \{0, 1\}^{m(k)} \rightarrow \{0, 1\}^{m(k)}$ of $\text{poly}(k)$ -time computable functions is pseudouniform if, for all algorithms A running in time $\text{poly}(k)$ the function

$$\left| \Pr_{A,v}[A(k, g_k(v)) = 1] - \Pr_{A,w}[A(k, w)] \right| = 1 \quad (8)$$

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 [GL89]. Conversely, given a pseudorandom generator one can get a pseudouniform one-way function by truncating the output.

Theorem 9. 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.

Proof. The first part follows immediately by the fact that a distinguisher can be converted to a next bit predictor [BM84] and the Goldreich-Levin Theorem [GL89].

For the second part, let $g : \{0, 1\}^{m(k)} \rightarrow \{0, 1\}^{m(k)+1}$ be a pseudorandom generator where we assume without loss of generality that g expands by 1 bit. If the truncation $g' : \{0, 1\}^{m(k)} \rightarrow \{0, 1\}^{m(k)}$ is not pseudouniform, there must be some distinguisher which has non-negligible advantage in distinguishing the output from a uniform random string. Such a distinguisher immediately contradicts the pseudorandomness of g .

Suppose now that $g' : \{0, 1\}^{m(k)} \rightarrow \{0, 1\}^{m(k)}$ is not a one-way function. Then, there exists some (inverse) polynomial $\epsilon(k)$ and some algorithm A which inverts g with probability at least ϵ for infinitely many k .

On some fixed security parameter k we now proceed as follows: first, let p be the probability that A finds a preimage of g' of a uniformly chosen element $y \in \{0, 1\}^m$ (i.e., the probability that $g'(A(y)) = y$ for a uniform random y). This can be arbitrary small, because the distribution is different from the distribution induced by $g'(x)$. Using sampling, we can find an estimate p' of p

such that with probability $1 - 2^{-k}$ the estimate satisfies $|p - p'| \leq \epsilon/4$. If $p' \leq \epsilon/2$, then we can distinguish the output of g from uniform by checking whether A inverts g' on the first $m(k)$ bits.

On the other hand, if $p' \geq \epsilon/2$ we can assume $p \geq \epsilon/4$. Now A immediately gives an inverter for g which inverts a random uniform bitstring of length $m + 1$ with probability at least $\epsilon/8$ (just ignore the last bit, invert g' , and hope the last bit matches). Finally, the probability an inverter for g inverts an output of g is at least twice the probability it inverts a uniform bitstring. Thus, we can get a distinguisher by checking whether A even finds an inverse of g , given only the first m bits of the result. \square

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.

3.2 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. As we will see in the proof of Theorem 5, in our case this is no problem (because of the way we construct the oracles f_k).

Definition 10. Let $\{0, 1\}^{m*} := \{0, 1\}^m \cup \{(\perp, v) | v \in \{0, 1\}^m\}$. An oracle function $g^{(f)} : \{0, 1\}^m \rightarrow \{0, 1\}^{m*}$ 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 $g'(v, y_1, \dots, y_r)$ be the function which never calls f but instead just uses y_i as the reply of f to the i th query.

3.3 Notations

Definition 11 (The Query-sets). The set $\text{Query}(g, v, f)$ is $\{(x_1, y_1), \dots, (x_r, y_r)\}$, 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 $\text{Query}(g', v, y_1, \dots, y_r)$ is defined similarly (in particular, it also contains pairs (x_i, y_i)). The sets $\text{QueryX}(g, v, f)$ and $\text{QueryY}(g, v, f)$ contain the x and y -part of the pairs in $\text{Query}(g, v, f)$.

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

$$f_{(x^*, y^*)}(x) := \begin{cases} y^* & \text{if } x = x^* \\ f(x) & \text{otherwise.} \end{cases} \quad (9)$$

We use the following sets of functions $f : \{0, 1\}^n \rightarrow \{0, 1\}^n$. 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 $f : \{0, 1\}^n \rightarrow \mathcal{Y}$. Then, \mathcal{P}_n is the set of all bijective functions $f : \{0, 1\}^n \rightarrow \{0, 1\}^n$, 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.

4 Overview of the Proofs

We now try to provide some intuition of the proofs. We concentrate on the proof of Theorem 5, and only say a few words about the other theorems in the end.

Basic setting By the discussion above, 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 $(\text{Breaker}, f)$, such that $\text{Breaker}(k, \cdot)$ either inverts g or distinguishes the output of g from a uniform random string, and yet $A^{(\text{Breaker}, f)}$ will not invert $f(k, \cdot)$ with noticeable probability.

4.1 The case of a single call

We first study the case where $g^{(f)}$ does a single call to the underlying one-way function.

Example constructions We first discuss three example constructions for $g^{(f)}$, which all do $r = 1$ calls to f .

The first example $g : \{0, 1\}^n \rightarrow \{0, 1\}^n$ is defined as $g(v) = f(v)$, i.e., the function simply applies the given one-way function. Clearly, g will be one-way, so that Breaker must distinguish the output of g from a random function; we will call such a breaker BreakPU. In this case, our proof will pick $f : \{0, 1\}^n \rightarrow \{0, 1\}^n$ as a very degenerate function (for example with image set of size $|\mathcal{Y}| = 2^{\log^2(n)}$). It is intuitive that BreakPU can distinguish the output of g from a uniform random string without helping to invert f .

The second example $g : \{0, 1\}^n \rightarrow \{0, 1\}^n$ is defined as $g(v) = v$, so that the function simply outputs the input v . In this case, clearly the function is pseudouniform, therefore Breaker will break the one-way property of g using exhaustive search. We will call such a breaker BreakOW.

The last example $g : \{0, 1\}^{2n} \rightarrow \{0, 1\}^{2n}$ is defined as

$$g(v, r) := \begin{cases} (v, r) & \text{if } r \neq 0^n \\ (f(v), r) & \text{otherwise.} \end{cases} \quad (10)$$

This function is pseudouniform no matter how f is defined. Thus, Breaker needs to invert g . One sees that it needs to be careful in that: if BreakOW($y, 0^n$) returns a preimage of g , clearly A will be able to invert. Thus, only images (y, r) with $r \neq 0^n$ should be inverted.

Inverting constructions with one call It turns out that we can describe BreakOW(w) in general as follows: enumerate all possible inputs v , and evaluate $g^{(f)}(v)$ on each of them. In case $g^{(f)}(v) = w$, BreakOW considers the output y which appeared in this evaluation as answer to the query done to f . It then considers the probability that w is the output in case nothing about f or v is known, but conditioned on y to appear in the evaluation (assuming that f is chosen as a permutation). If this probability is large (concretely, larger than $2^{-m+n/30}$), BreakOW refuses to answer. Otherwise, it returns v .

A very quick intuition why this might not help to invert f is as follows: suppose an algorithm $A^{(\text{BreakOW}, f)}(y)$ tries to invert y . In order to do use BreakOW, A needs to find some useful w for this y . However, BreakOW ensures that it only inverts w which are not very likely to be outputs for this y , so that A is unlikely to find a matching w . Thus, we can hope that A will fail.

We will sketch the actual proof that f remains one-way given BreakOW later.

Invert or distinguish constructions with one call We now distinguish two cases: if BreakOW inverts g for a randomly chosen f with probability (say) $\frac{1}{2}$, clearly we are done. Otherwise, it must be that very often in random evaluations of g , once the output of f is fixed to y , certain values w are much more likely (if the rest, i.e., v , and f , are still chosen at random). In this case, we first pick $\mathcal{Y} \subseteq \{0,1\}^n$, $|\mathcal{Y}| = 2^{\log^2(n)}$ as image set. We then show that there is some small set $W(\mathcal{Y})$ depending only on \mathcal{Y} , such that if we pick f from $\mathcal{F}(\mathcal{Y})$ and v uniformly at random, with high probability $g^{(f)}(v) \in W$. Thus, we can distinguish the output of $g^{(f)}(v)$ from a uniform random string by just checking whether it is in W , and this without even knowing the details of f (namely, we can still pick $f : \{0,1\}^n \rightarrow \mathcal{Y}$ uniformly at random).

The reason that $g^{(f)}(v) \in W(\mathcal{Y})$ is likely should be intuitive: we know that conditioning on some fixed y highly biases the output w , and because there are only few $y \in \mathcal{Y}$, the output should still be biased overall.

The underlying one-way function remains one-way We still need to argue that BreakOW does not help to invert a random permutation f . For this, suppose $A^{(\text{BreakOW}, f)}(y_0)$ tries to invert $y_0 = f(x_0)$. Pick a random x^* and consider the function $f^* = f_{(x^*, y_0)}$, as defined in Section 3.3. Also, let BreakOW* be defined as Breaker, except that it uses f^* instead of f when it evaluates g in the exhaustive search.

Intuitively, if $A^{(\text{BreakOW}, f)}(y_0)$ is likely to return x_0 , then $A^{(\text{BreakOW}^*, f^*)}(y_0)$ must be at least somewhat likely to return x^* , because x^* has the same distribution as x_0 from A 's point of view (the same argument was previously used in [HH09], and in a more convoluted way in [Sim98]). This means that the two runs of A have to differ in some call with noticeable probability. It is unlikely that they differ in a call to f , since x^* was picked at random and A makes few calls to f . Thus, they have to differ in some call to BreakOW with noticeable probability.

However, it turns out that $\text{BreakOW}^*(w) \neq \text{BreakOW}(w)$ for any w with very low probability: it only happens in two cases. First, if x^* is the query which $g^{(f)}(\text{BreakOW}(w))$ makes to f , but there is only one such query, so this happens with probability 2^{-n} (over the choice of x^*).

The other case is if there is some v for which the output of $g^{(f)}(v)$ changes to w when we replace f with f^* .

Now, recall the check BreakOW performs before it outputs v . This check is equivalent to the following: enumerate all pairs (v', y') , and count the number for which $g(v') = w$ in case f answers the only query with y' . If this number is larger than $2^{n/30}$, refuse to return v .

This now implies that there can only be $2^{n/30}$ values for x^* for which the output changes to w , and so this case is unlikely as well.

4.2 Multiple calls

The case when g can make more than 1 call is significantly more difficult than the case where g makes a single call. It turns out that most of the issues which arise can be discussed already for $r = 2$ calls, so we restrict the discussion to this case in this section.

Construction with many calls Of course, the same examples as before still work. Thus, BreakOW(w) still does the same check before returning v : does conditioning on one of the two query answers y_1 and y_2 given by f in the evaluation of $f(v)$ make w much more likely? If so, it refuses to answer.

However, it turns out that we can restrict BreakOW(w) even more: it should also not return a preimage v if conditioning on having seen *both* outputs y_1 and y_2 in an evaluation makes the output

w more likely. As it turns out, we only know how to prove that BreakOW does not help invert f with this additional restriction.

A useful example might be the construction, which takes as input a $v = (x_1, x_2)$ of length $2n$, and is defined as

$$g^{(f)}(x_1, x_2) = \begin{cases} (f(x_1), f(x_2)) & \text{if } f(x_1) = f(x_2) \oplus (1, \dots, 1) \\ (x_1, x_2) & \text{otherwise.} \end{cases} \quad (11)$$

This will be a pseudouniform function, because usually $f(x_1) \neq f(x_2) \oplus (1, \dots, 1)$. Also, we see that an adversary A which tries to use BreakOW to invert y would presumably call BreakOW on input $(y, y \oplus (1, \dots, 1))$. However, using the additional restriction above, BreakOW will definitely not return the inverse A is looking for.

We make two additional remarks: 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 $r \notin o(n/\log(n))$.

Second, there is a question on whether above one should condition on y_1 being the first output, and y_2 being the second output, or just on both y_1 and y_2 appearing as an output. We choose the latter, as it seems more natural in the concentration bound explained below. It seems we can be relatively careless with this, because $r^r \ll 2^n$.

The underlying one-way function still remains one-way Again, we need to argue why BreakOW does not help to invert f . As before, we can show that we only need to prove that with high probability over the choice of x^* we have $\text{BreakOW}^*(w) = \text{BreakOW}(w)$. Previously, this followed by a simple counting argument. Now, it becomes more difficult.

To see why, consider

$$g^{(f)}(x_1, x_2) = \begin{cases} 0^{2n} & \text{if } x_1 = x_2 = 1^n \\ 1^{2n} & \text{otherwise, if also } x_1 = f(x_1) \text{ and } x_2 \neq f(x_2) \\ (x_1, x_2) & \text{otherwise} \end{cases} \quad (12)$$

One can check that neither conditioning on a value of y_1 , y_2 , or on a pair (y_1, y_2) makes some output w of g much more likely. Therefore, $\text{BreakOW}(w)$ will simply return some preimage found.

Suppose now that f was picked in a very unlikely way: $f(x) = x$ for any x . Then, $\text{BreakOW}(1^{2n})$ will return \perp , signifying that no preimage was found. On the other hand, for any x^* and any f^* as above, $\text{BreakOW}^*(1^{2n})$ will return (x_1, x^*) for some x_1 . Thus, for *some* functions f , BreakOW^* can behave very differently from BreakOW.

It is, however, possible to show that functions f for which this happens are very unlikely. In case $r = 2$, a usual Chernoff bound is sufficient for that. For r larger than 2, a concentration bound for polynomials in the style as proven by [KV00] seems to be needed. We will use a bound from [Hol11], and show in Section 6 how it can be used to show that for almost all functions f , $\text{BreakOW}^f(w) \neq \text{BreakOW}^{f^*}(w)$ has very low probability (over the choice of x^*).

It turns out that this concentration bound breaks down if $r \in \Omega(n/\log(n))$.

4.3 Non-uniform security reductions

The above considerations prove Theorem 5, which exclude constructions with uniform security proofs. The technique given in [GGKT05] allows to give security proofs which also hold against non-uniform security proofs, and we can apply this technique in our context. We apply this technique in Section 7, giving Theorem 6.

4.4 On the security parameter restriction

Given our techniques, one might suspect that the restriction on the security parameter is inherent to them. However, as we show in Section 8, this is not the case. Our proof will only break the resulting pseudouniform one-way function only for infinitely many security parameters k , instead of for all but finitely many k as one might hope.

This last restriction is inherent, at least as long as one only uses underlying regular one-way functions. The reason is that constructions exist which do fewer than $n/\log(n)$ calls, and yield a pseudorandom generator for infinitely many security parameters.

In order to get rid of the restriction, we use the following idea: We consecutively find infinitely many values ℓ for which $g(\ell, \cdot)$ does fewer than $\frac{n}{\log(n)}$ queries, where n is the *shortest* input length which g queries on security parameter ℓ . After this, we simultaneously fix $f(k, \cdot)$ for all k which g can access. The idea is that underlying to $f(k, \cdot)$, there could be a *single* one-way function for many different values of k . Thus, we can reduce our task to the problem solved in the previous sections.

Of course some technical problems arise. These are dealt with in Section 8.

5 The Breaker Oracles

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 Breaker_k to be one of these two oracles, depending on the combinatorial structure of g_k .

5.1 The inverting oracle

The first oracle is called BreakOW. It inverts g in some cases, and is given as algorithm below, but we first explain it informally. On input $w \in \{0, 1\}^m$, 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( $w, Q$ ):           //SafeToAnswer does not depend on  $f$ 
  for all  $B \subseteq Q$ :
    if  $\Pr_{f' \leftarrow \mathcal{P}, v'}[g^{(f')}(v') = w | B \subseteq \text{QueryY}(g, v', f')] \geq 2^{-m + \frac{n}{30}}$ 
      return false
  return true
done

for all  $v \in \{0, 1\}^m$  do
```

<pre> if $g^{(f)}(v) = w$ then if $\text{SafeToAnswer}(w, \text{QueryY}(g, v, f))$ then return v return \perp </pre>
--

We next define the 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 $p(g)$).

$$p(g) := \Pr_{\substack{f \leftarrow \mathcal{P} \\ v \leftarrow \{0,1\}^m}} [\text{SafeToAnswer}(g^{(f)}(v), \text{QueryY}(g, f, v))] \quad (13)$$

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

Lemma 12. *Let $g^{(\cdot)} : \{0,1\}^m \rightarrow \{0,1\}^{m*}$ be a normalized oracle construction. If $p(g) \geq \frac{1}{2}$, then*

$$\Pr_{f \leftarrow \mathcal{P}, v} [\text{BreakOW}(g^{(f)}(v)) \text{ inverts } g^{(f)}] \geq \frac{1}{2}. \quad (14)$$

Proof. Pick v and f at random and call $\text{BreakOW}(w)$ for $w = g^{(f)}(v)$. When BreakOW enumerates all possible values v , at one point it will pick the actual chosen value v unless it has returned a preimage of w before. With probability at least $\frac{1}{2}$, $\text{SafeToAnswer}(w, Q)$ returns true where $Q = \text{QueryY}(g, v, f)$, in which case $\text{BreakOW}(w)$ will return some inverse of w . \square

Our next goal is a more interesting claim: BreakOW is unlikely to help inverting f , when is uniformly drawn from \mathcal{P} . For this, we introduce the following definition (which is motivated by the soon to follow Lemma 15).

Definition 13. *Let $g^{(\cdot)} : \{0,1\}^m \rightarrow \{0,1\}^{m*}$ be an r -query normalized oracle construction. For $f : \{0,1\}^n \rightarrow \{0,1\}^n$, $y^* \in \{0,1\}^m$, 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) $\text{SafeToAnswer}(w, \text{QueryY}(g, v^*, f^*))$,

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

We will prove the next lemma in Section 6 (some intuition on why this is true can be found in Section 6.1). It states that with very high probability over the choice of f , the set $Q_{f,y^*,w}$ is small.

Lemma 14. *Let $g^{(\cdot)} : \{0,1\}^m \rightarrow \{0,1\}^{m*}$ be an r -query normalized oracle construction, $r \leq \frac{n}{100 \log(n)}$. For all (w, y^*) we have*

$$\Pr_{f \leftarrow \mathcal{P}} [|Q_{f,w,y^*}| > 2^{\frac{n}{10}}] < 2^{-2^{\frac{n}{100r}}}. \quad (15)$$

Fix now some permutation f , some $y^* \in \{0,1\}^n$ and some $w \in \{0,1\}^m$. Compare runs of $\text{BreakOW}^{(f)}(w)$ and $\text{BreakOW}^{(f_{(x^*, y^*)})}(w)$ 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 15. Fix f, y^*, w . If $|Q_{f,y^*,w}| \leq 2^{\frac{n}{10}}$, then

$$\Pr_{x^*}[\text{BreakOW}^{(f)}(w) \neq \text{BreakOW}^{(f^*)}(w)] \leq 2^{-\frac{4n}{5}} \quad (16)$$

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

Proof. Let v be the result of $\text{BreakOW}^{(f)}(w)$, and v^* the result of $\text{BreakOW}^{(f^*)}(w)$. We distinguish two cases.

First, suppose that $v^* = \perp$ or that v^* occurs in the enumeration of BreakOW after v . This can only happen if $x^* \in \text{QueryX}(g, v, f)$, because if not, $\text{BreakOW}^{(f^*)}(w)$ will behave exactly the same in the iteration of v , and so it must also return v .

Second, suppose that $v = \perp$ or that v occurs in the enumeration of BreakOW after v^* . We claim that in this case $(x^*, v^*) \in Q_{f,y^*,w}$. Clearly, conditions (a) and (c) in Definition 13 must hold, as otherwise $\text{BreakOW}^{(f^*)}(w)$ will not output v^* . Condition (b) must also hold. Otherwise we have that $g^{(f)}(v^*) = w$ (because of (a) and the fact that x^* has not been queried) and $\text{QueryY}(g, v^*, f) = \text{QueryY}(g, v^*, f^*)$. This would imply that $\text{SafeToAnswer}(w, \text{QueryY}(g, v^*, f)) = \text{SafeToAnswer}(w, \text{QueryY}(g, v^*, f^*))$ and so we see that if (b) would not hold, $\text{BreakOW}^{(f)}(w) = v^*$.

Since the union of the sets $\text{QueryX}(g, v, f)$ and $Q_{f,y^*,w}$ has fewer than $2^{\frac{n}{5}}$ elements the result follows. \square

Now we can show that BreakOW usually does not help to invert f .

Lemma 16. Let $g^{(\cdot)} : \{0,1\}^m \rightarrow \{0,1\}^{m^*}$ be an r -query normalized oracle construction, $r < \frac{n}{100 \log(2n+m)}$. 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

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

Proof. First, because f is picked from the set of permutations \mathcal{P} , we see that

$$\Pr_{x, f \leftarrow \mathcal{P}, A} [A^{(\text{BreakOW}, f)}(f(x)) \text{ inverts } f(x)] = \Pr_{x, f \leftarrow \mathcal{P}, A} [A^{(\text{BreakOW}, f)}(f(x)) = x] \quad (18)$$

Fix now an arbitrary function f . In case f is such that for all pairs (w, y^*) the bound $|Q_{f,w,y^*}| \leq 2^{\frac{n}{10}}$ holds, we get for any x and any fixed randomness of A

$$\Pr_{x^*} [A^{(\text{BreakOW}, f)}(f(x)) \neq A^{(\text{BreakOW}^*, f^*)}(f(x))] \leq 2^{\frac{n}{20}} 2^{-\frac{4n}{5}} < 2^{-\frac{n}{20}} \quad (19)$$

where $f^* = f_{(x^*, f(x))}$, $\text{BreakOW} = \text{BreakOW}^{(f)}$, and $\text{BreakOW}^* = \text{BreakOW}^{(f^*)}$. This holds because any of the $2^{\frac{n}{20}}$ calls to either oracle will return the same answer with probability $2^{-\frac{4n}{5}}$ (using Lemma 15 for calls to BreakOW , for calls to f this is obvious).

We can also pick x and f at random, then we get

$$\begin{aligned} & \Pr_{f, x, x^*} [A^{(\text{BreakOW}, f)}(f(x)) \neq A^{(\text{BreakOW}^*, f^*)}(f(x))] \\ & \leq \Pr_f [\exists (w, y^*) : |Q_{f,w,y^*}| > 2^{\frac{n}{10}}] + 2^{-\frac{n}{20}} \\ & \leq 2^{m+n-2^{\frac{n}{100r}}} + 2^{-\frac{n}{20}} < 2^{-\frac{n}{20}+1}, \end{aligned} \quad (20)$$

where we applied Lemma 14.

Still fixing the randomness of A , we see that

$$\Pr_{f,x}[A^{(\text{BreakOW},f)}(f(x)) = x] \leq \Pr_{f,x^*,x}[A^{(\text{BreakOW}^*,f^*)}(f(x)) = x] + 2^{-\frac{n}{20}+1} \quad (21)$$

$$= \Pr_{f,x^*,x}[A^{(\text{BreakOW}^*,f^*)}(f(x)) = x^*] + 2^{-\frac{n}{20}+1} \quad (22)$$

$$\leq \Pr_{f,x^*,x}[A^{(\text{BreakOW},f)}(f(x)) = x^*] + 2^{-\frac{n}{20}+2} \quad (23)$$

$$= 2^{-n} + 2^{-\frac{n}{20}+2} < 2^{-\frac{n}{30}}, \quad (24)$$

where we get (22) because the triples $(f^*, f(x), x)$ and $(f^*, f(x), x^*)$ have exactly the same distribution. We used (20) to get (21) and (23).

Since this holds for each random choice A can make, it must also hold overall. \square

5.2 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, \dots, 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, it is not too far fetched to hope that if f is a function $f : \{0,1\}^n \rightarrow \mathcal{Y}$ 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 the following lemma.

Lemma 17. *Let $g^{(\cdot)} : \{0,1\}^m \rightarrow \{0,1\}^{m^*}$ 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*

$$\Pr_{\substack{f \leftarrow \mathcal{F}(\mathcal{Y}) \\ v \leftarrow \{0,1\}^m}} [g^{(f)}(v) \in W] \geq \frac{1}{2} - r^2 2^{-\frac{n}{100r}} \quad (25)$$

Proof. We pick $\mathcal{Y} \subseteq \{0,1\}^n$ of size $2^{\frac{n}{100r}}$ uniformly at random, and then set

$$W = \left\{ w \mid \exists Q \subseteq \mathcal{Y} : |Q| = r \wedge \neg \text{SafeToAnswer}(w, Q) \right\}. \quad (26)$$

We start by showing that $|W| \leq 2^{m-\frac{n}{100}}$. There are fewer than $(|\mathcal{Y}|)^r = 2^{\frac{n}{100}}$ subsets $Q \subseteq \mathcal{Y}$ of size $|Q| = r$, and for each of them, SafeToAnswer considers $2^r \leq 2^{\frac{n}{100}}$ subsets B . For each B , there can be at most $2^{m-\frac{n}{30}}$ elements w which have probability at least $2^{-m+\frac{n}{30}}$ conditioned on $B \subseteq \text{QueryY}(g, v, f')$. Thus, in total there can be at most $2^{m-\frac{n}{30}+\frac{n}{100}+\frac{n}{100}} \leq 2^{m-\frac{n}{100}}$ elements in W .

To see (25), we note first that

$$\Pr_{\substack{\mathcal{Y}, f \leftarrow \mathcal{F}(\mathcal{Y}) \\ v \leftarrow \{0,1\}^m}} [g^{(f)}(v) \in W] = \Pr_{\substack{\mathcal{Y}, v \leftarrow \{0,1\}^m \\ (y_1, \dots, y_r) \leftarrow (P(r, \mathcal{Y}))}} [g'(v, y_1, \dots, y_r) \in W], \quad (27)$$

where the distribution $P(r, \mathcal{Y})$ over \mathcal{Y}^r is the distribution of $(f(x_0), \dots, f(x_{r-1}))$ for some fixed pairwise disjoint values x_0, \dots, x_{r-1} and $f \leftarrow \mathcal{F}(\mathcal{Y})$. It has the following two properties. First, the probability that $P(r, \mathcal{Y})$ gives r pairwise disjoint outputs is at least $1 - \frac{r^2}{|\mathcal{Y}|}$ (by a union bound). Second, all tuples (y_1, \dots, y_r) in which the elements are pairwise disjoint have the same probability when the probability is also over the choice of \mathcal{Y} .

Thus,

$$\Pr_{\substack{\mathcal{Y}, v \leftarrow \{0,1\}^m \\ (y_1, \dots, y_r) \leftarrow (P(r, \mathcal{Y}))}} [g'(v, y_1, \dots, y_r) \in W] \quad (28)$$

$$\geq \Pr_{\substack{\mathcal{Y}, v \leftarrow \{0,1\}^m \\ (y_1, \dots, y_r) \leftarrow (P(r, \mathcal{Y}))}} [|\{y_1, \dots, y_r\}| = r] \times \quad (29)$$

$$\Pr_{\substack{\mathcal{Y}, v \leftarrow \{0,1\}^m \\ (y_1, \dots, y_r) \leftarrow (P(r, \mathcal{Y}))}} [g'(v, y_1, \dots, y_r) \in W \mid |\{y_1, \dots, y_r\}| = r] \quad (30)$$

$$\geq \Pr_{\substack{\mathcal{Y}, v \leftarrow \{0,1\}^m \\ (y_1, \dots, y_r) \leftarrow (P(r, \mathcal{Y}))}} [|\{y_1, \dots, y_r\}| = r] \times \quad (31)$$

$$\Pr_{(y_1, \dots, y_r)} [g'(v, y_1, \dots, y_r) \in W], \quad (32)$$

where in this last probability the values y_1, \dots, y_r are picked uniformly without repetition. Next, we see that

$$\Pr_{(y_1, \dots, y_r)} [g'(v, y_1, \dots, y_r) \in W] \quad (33)$$

$$\geq \Pr_{(y_1, \dots, y_r)} [\neg \text{SafeToAnswer}(g'(v, y_1, \dots, y_r), \{y_1, \dots, y_r\})] \quad (34)$$

$$= 1 - p(g) \quad (35)$$

because without repetition the y_i have exactly the same distribution as in the definition of $p(g)$. In total,

$$\Pr_{\substack{\mathcal{Y}, f \leftarrow \mathcal{F}(\mathcal{Y}) \\ v \leftarrow \{0,1\}^m}} [g^{(f)}(v) \in W] \geq \left(1 - \frac{r^2}{|\mathcal{Y}|}\right)(1 - p(g)). \quad (36)$$

□

Let now $\text{BreakPU}(W)$ 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 . This is intuitive, since it does not even depend on f (besides the choice of \mathcal{Y}). Furthermore, this lemma also follows directly from [GGKT05, Theorem 1]. To see this, note that we can pick f as follows: first pick any regular function $p : \{0,1\}^n \rightarrow \mathcal{Y}$ and then set $f = \pi \circ p$ for some permutation π ; by [GGKT05, Theorem 1], f is $2^{|\mathcal{Y}|^{(1/5)}}$ -hard to invert even given p . We provide a proof anyhow for completeness.

Lemma 18. *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,*

$$\Pr_{f \leftarrow \mathcal{F}(\mathcal{Y}), x, A} [A^{f, \text{BreakPU}}(f(x)) \text{ inverts } f] \leq 2^{-\frac{n}{1000r}}, \quad (37)$$

where $\text{BreakPU} = \text{BreakPU}(W)$ for an arbitrary set W .

The proof is similar to the proof of Lemma 16.

Proof. We first note that

$$\Pr_{\substack{f \leftarrow \mathcal{F}(\mathcal{Y}) \\ x, A}} [A^{f, \text{BreakPU}}(f(x)) \text{ inverts } f(x)] = \frac{2^n}{|\mathcal{Y}|} \Pr_{\substack{f \leftarrow \mathcal{F}(\mathcal{Y}) \\ x, A}} [A^{f, \text{BreakPU}}(f(x)) = x] \quad (38)$$

because for any fixed $f(x)$, the value of x is still uniform among the $\frac{2^n}{|\mathcal{Y}|}$ preimages.

Now, fix any x and any f , and let $q = 2^{\frac{n}{1000r}}$ be the upper bound on the number of queries by A . Keeping the randomness of A fixed,

$$\Pr_{x^*}[A^{f,\text{BreakPU}}(f(x)) \neq A^{f^*,\text{BreakPU}}(f(x))] \leq \frac{q}{2^n}, \quad (39)$$

where $f^* = f_{(x^*, f(x))}$, because the output of $A^{f^*,\text{BreakPU}}(f(x))$ can only differ from $A^{f,\text{BreakPU}}(f(x))$ in case x^* is one of the elements on which A queried f .

As in the proof of Lemma 16,

$$\Pr_{\substack{f \leftarrow \mathcal{F}(\mathcal{Y}) \\ x, x^*}}[A^{f,\text{BreakPU}}(f(x)) = x] \leq \Pr_{\substack{f \leftarrow \mathcal{F}(\mathcal{Y}) \\ x, x^*}}[A^{f^*,\text{BreakPU}}(f(x)) = x] + \frac{q}{2^n} \quad (40)$$

$$= \Pr_{\substack{f \leftarrow \mathcal{F}(\mathcal{Y}) \\ x, x^*}}[A^{f^*,\text{BreakPU}}(f(x)) = x^*] + \frac{q}{2^n} \leq \frac{2q+1}{2^n}. \quad (41)$$

Together,

$$\Pr_{x \leftarrow \{0,1\}^n, f}[A^{f,\text{BreakPU}}(f(x)) \text{ inverts } f(x)] \leq \frac{2^n}{|\mathcal{Y}|} \frac{2q+1}{2^n} = \frac{2q+1}{|\mathcal{Y}|}. \quad (42)$$

□

5.3 Proving the main result

The above lemmas can be used to prove Theorem 5, which we restate here for reference.

Theorem 5. *Let $n(k), r(k) \in \text{poly}(k)$ be computable in time $\text{poly}(k)$, 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 $f(k, \cdot)$.*

Proof. In order to get a contradiction, we assume otherwise. Because of Theorem 9, we can also assume that we have a fully black-box reduction which gives a pseudouniform one-way function (which is defined in a way analogous to Definition 3).

Thus, suppose we have some construction (g, A) . We want to instantiate the construction with length preserving one-way functions, where the input and output length equals the security parameter k , i.e., $n(k) := n'(k) := k$. The construction must work for this choice by definition.

We can assume that $\frac{n(k)}{1000r(k)} \in \mathbb{N}$ for all but finitely many k , because we can increase $r(k)$ such that this holds and such that still $r(k) \in o(\frac{n(k)}{\log(n(k))})$.

We now make sure that our construction is normalized. For this, we modify g such that it makes exactly $r(k)$ pairwise disjoint queries to f ; clearly, this is no problem.

We then define

$$\tilde{g}^{(f_k)}(v) := \begin{cases} (\perp, v) & \text{if two queries of } f_k \text{ yield the same output} \\ g_k^{(f_k)}(v) & \text{otherwise.} \end{cases} \quad (43)$$

Next, we will provide, for each k separately, two oracles f and $B = \text{Breaker}$. We construct these oracles such that B breaks the security property of g_k for all but finitely many k , and yet the probability that $A^{\text{Breaker}, f}$ inverts f is negligible.

For this, we consider $p(\tilde{g}_k)$ for each k separately. If $p(\tilde{g}_k) \geq \frac{1}{2}$ we set Breaker_k to be BreakOW . By Lemma 12 we see that for these k

$$\Pr_{f_k \leftarrow \mathcal{P}_{k,v}} [\text{Breaker}_k(\tilde{g}_k^{(f_k)}(v)) \text{ inverts } \tilde{g}_k^{(f_k)}] \geq \frac{1}{2}. \quad (44)$$

By Lemma 16 we also see that, if k is large enough,

$$\Pr_{f_k \leftarrow \mathcal{P}_{k,x,A}} [A^{f_k, \text{Breaker}_k}(f_k(x)) \text{ inverts } f_k] \leq 2^{-\frac{n(k)}{30}}. \quad (45)$$

Note that in this case, \tilde{g}_k behaves the same as g_k , because no two queries to f_k can output the same value. Applying Markov's inequality, for fraction at least $\frac{1}{10}$ of the functions f_k we have

$$\Pr_v [\text{Breaker}_k(g_k^{(f_k)}(v)) \text{ inverts } g_k^{(f_k)}] \geq \frac{1}{10} \quad (46)$$

Furthermore, for fraction at least $\frac{99}{100}$ of the functions f_k we have

$$\Pr_{x,A} [A^{f_k, \text{Breaker}_k}(f_k(x)) \text{ inverts } f_k] \leq 100 \cdot 2^{-\frac{n(k)}{30}}. \quad (47)$$

We pick a function f_k for which both (46) and (47) are satisfied.

If $p(\tilde{g}_k) \leq \frac{1}{2}$, Lemma 17 gives a set $W_k \subseteq \{0,1\}^{m(k)}$ and $\mathcal{Y}_k \subseteq \{0,1\}^{n(k)}$. For n large enough, A satisfies the requirements of Lemma 18, and we see that

$$\Pr_{\substack{f_k \leftarrow \mathcal{F}(\mathcal{Y}_k) \\ A, x \leftarrow \{0,1\}^{n(k)}}} [A^{\text{BreakPU}(W_k), f_k}(f_k(x)) \text{ inverts } f_k(x)] \leq 2^{-\frac{n}{1000r}}, \quad (48)$$

which is negligible. By Lemma 17

$$\Pr_{\substack{f_k \leftarrow \mathcal{F}(\mathcal{Y}_k) \\ v \leftarrow \{0,1\}^{m(k)}}} [\tilde{g}_k^{(f_k)}(v) \in W_k] \geq \frac{1}{4}, \quad (49)$$

again for k is large enough. Because \mathcal{Y} is of superpolynomial size, the probability that \tilde{g}_k outputs (\perp, v) is still negligible. Thus, we can argue as before, and there is some choice of f_k for which

$$\Pr_{A, x \leftarrow \{0,1\}^{n(k)}} [A^{\text{Breaker}_k, f_k}(f_k(x)) \text{ inverts } f_k(x)] \leq 2^{-\frac{n}{110r}}, \text{ and} \quad (50)$$

$$\Pr_{v \leftarrow \{0,1\}^{m(k)}} [g_k^{(f_k)}(v) \in W_k] \geq \frac{1}{10}. \quad (51)$$

We fix such a choice of for f_k and set $\text{Breaker}_k := \text{BreakPU}(W_k)$.

We conclude that while the statement analogous to (3) holds (for breaking the either the pseudouniformity or for inverting g), the statement (4) fails to hold, and so we get a contradiction. \square

6 Proof of Lemma 14

In this section, we give the proof of Lemma 14. However, before giving the proof, we provide some intuition in Section 6.1 (which can be skipped if desired).

6.1 Intuition

Fix (f, y^*, w) , and assume that $(x^*, v^*) \in Q_{f, y^*, w}$. Consider the query-answer pairs $\{(x_1, y_1), \dots, (x_r, y_r)\} = \text{Query}(g, v^*, f_{(x^*, y^*)})$ 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 13 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 $T = \{(x_1, y_1), \dots, (x_{r-1}, y_{r-1})\}$. Let us call T an *incrementor* for $|Q_{f, y^*, w}|$, because whenever f satisfies $f(x_i) = y_i$ for $i \in \{1, \dots, r-1\}$, the set $Q_{f, y^*, w}$ grows by 1.⁷

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)

$$\Pr_{f' \leftarrow \mathcal{P}, v'}[g^{f'}(v') = w | y_r = y^*] \geq 2^{-m + \frac{n}{30}}, \quad (52)$$

because any of the incrementors survive⁸ the picking of f with probability roughly⁹ $2^{-(r-1)n}$. 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 one¹⁰ 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, (52) 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 [KV00] 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 17, 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 [SSS95] (see [Rao08, IK10]). A proof of a Kim-Vu style concentration bound in this form was given by the first author in [Hol11].

6.2 The polynomial P_{w, y^*}

To prove Lemma 14, we will first find a polynomial P_{w, y^*} of degree r in variables $F_{(x, y)}$ for all $x, y \in \{0, 1\}^n$. The polynomial will have the following property: fix an arbitrary function $f : \{0, 1\}^n \rightarrow \{0, 1\}$, and set the variables $F_{(x, y)}$ as follows:

$$F_{(x, y)} = \begin{cases} 1 & \text{if } f(x) = y \\ 0 & \text{otherwise.} \end{cases} \quad (53)$$

We will see that the value of P_{w, y^*} for these values (evaluated over \mathbb{R} or \mathbb{N}) gives an upper bound on $|Q_{f, y^*, w}|$. We denote this value by $P_{w, y^*}(f)$.

⁷Ignoring a few reasons why this might not be true sometimes...like the fact that SafeToAnswer might return false.

⁸Formally, surviving means that $f(x_i) = y_i$ for all pairs (x_i, y_i) in the incrementor.

⁹Ignoring very slight dependence in this discussion which arises from the fact that f is picked as a permutation.

¹⁰Only one incrementor with a fixed v^* can survive with our assumptions.

The polynomial P_{w,y^*} is obtained by a run of algorithm $\text{BuildPolynomial}(w, y^*)$.

Algorithm $\text{BuildPolynomial}(w, y^*)$

```

 $P_{w,y^*} := 0$ 
forall  $(y_1, \dots, y_r) \in \{0, 1\}^r$  do
    if  $\text{SafeToAnswer}(w, \{y_1, \dots, y_r\})$  then
        forall  $v \in \{0, 1\}^m$  do
            if  $g'(v, y_1, \dots, y_r) = w \wedge y^* \in \{y_1, \dots, y_r\}$  then
                 $T := \{(x_i, y_i) : i \in \{1, \dots, r\} \text{ and } y_i \neq y^* \text{ and } \dots$ 
                     $x_i \text{ is the } i\text{th query done by } g'(v, y_1, \dots, y_r)\}$ 
                 $P_{w,y^*} := P_{w,y^*} + \prod_{(x,y) \in T} F_{(x,y)}$ 
return  $P_{w,y^*}$ 

```

For readers who did not skip the intuition, we can connect this with Section 6.1. The term $\prod_{(x,y)} F_{(x,y)}$ corresponds to an incrementor, and we note that if the incrementor survives the picking, the summand in the polynomial will evaluate to 1.

Lemma 19. *For any f, w, y^* we have $|Q_{f,w,y^*}| \leq P_{w,y^*}(f)$.*

Proof. Pick f at first, and then consider a run of BuildPolynomial . We show that for each pair $(x^*, v^*) \in Q_{f,w,y^*}$ the procedure $\text{BuildPolynomial}(w, y^*)$ adds a monomial to P_{w,y^*} which evaluates to 1 under f .

Fix now a pair $(x^*, v^*) \in Q_{f,w,y^*}$, and let $(x_1, y_1), \dots, (x_r, y_r)$ be the pairs of queries and answers made to $f_{(x^*, y^*)}$ in an evaluation of $g^{(f_{(x^*, y^*)})}(v^*)$. It must be that $x^* \in \{x_1, \dots, x_r\}$, because of condition (b) in Definition 13, and so $(x^*, y^*) \in \{(x_1, y_1), \dots, (x_r, y_r)\}$. $\text{SafeToAnswer}(w, \{y_1, \dots, y_r\})$ must also hold (as otherwise $(x^*, v^*) \notin Q_{f,w,y^*}$). Thus, when BuildPolynomial enumerates the values (y_1, \dots, y_r) and v^* , it adds $\prod_{(x,y) \in T} F_{(x,y)}$ to P_{w,y^*} , which is 1 for the assignment given by f to the variables (note that T does not contain (x^*, y^*)). \square

6.3 Derivatives of P_{w,y^*}

Let now $B \subset \{F_{(x,y)}\}$ be a subset of the random variables $F_{(x,y)}$. For any multilinear polynomial P in the variables $\{F_{(x,y)}\}$ we let $\partial_B P$ be the formal derivative of P with respect to the variables in B . For example, $\partial_{\{F_{(1,1)}, F_{(2,2)}\}}(F_{(1,1)}F_{(2,2)}F_{(3,3)} + F_{(1,1)}F_{(3,3)}F_{(4,4)}) = F_{(3,3)}$.

Let \mathcal{F}^* be the distribution over the variables $F_{(x,y)}$ in which each $F_{(x,y)}$ is 1 with probability $\frac{1}{2^n}$ and 0 otherwise, and *all variables are independent*. When we pick the variables according to this distribution, they usually cannot have been derived from a function f as in (53). Nevertheless, this distribution is useful to express combinatorial properties of our polynomials. We denote the value of the polynomial evaluated at such a point \mathbf{F} by $P_{w,y^*}(\mathbf{F})$.

Lemma 20. *For any $B \subseteq (\{0, 1\}^n)^2$ and any (w, y^*) :*

$$\mathbb{E}_{\mathbf{F} \leftarrow \mathcal{F}^*} [(\partial_B P_{w,y^*})(\mathbf{F})] \leq 2^{\frac{2n}{30}}. \quad (54)$$

Proof. Suppose otherwise, and fix a triple (B, w, y^*) for which (54) fails to hold. We will derive a contradiction.

The polynomial $\partial_B P_{w,y^*}$ is the sum of all monomials in P_{w,y^*} which contain the factor $\prod_{(x,y) \in B} F_{(x,y)}$, but with this factor removed. Each such summand contributes $2^{-n(r-1-|B|)}$ to the expectation

in (54), and so there are at least $2^{\frac{2n}{30} + n(r-1-|B|)}$ monomials containing a factor $\prod_{(x,y) \in B} F_{(x,y)}$ in P_{w,y^*} . This implies that there are at least that many monomials containing a factor of the form $\prod_{(x,y) \in B} F_{(*,y)}$, where $F_{(*,y)}$ is an arbitrary variable $F_{(x',y')}$ with $y' = y$.

The following algorithm adds $2^{-n(r-1-|B|)}$ to a counter for each such monomial in P_{w,y^*} . Therefore, it outputs at least $2^{\frac{2n}{30}}$.

Algorithm UpperBound(B, w, y^*)

```

 $E_{B,w,y^*} := 0$ 
 $B' := \{y^*\} \cup \{y : (x, y) \in B\}$ 
forall  $(y_1, \dots, y_r) \in (\{0, 1\}^n)^r$  do
    if  $B' \subseteq \{y_1, \dots, y_r\}$  then
        if SafeToAnswer( $w, \{y_1, \dots, y_r\}$ ) then
            forall  $v \in \{0, 1\}^m$  do
                if  $g'(v, y_1, \dots, y_r) = w$  then
                     $E_{B,w,y^*} := E_{B,w,y^*} + 2^{-n(r-|B'|)}$ 
return  $E_{B,w,y^*}$ 

```

Consider now an arbitrary (y_1, \dots, y_r) for which E_{B,w,y^*} gets increased in this algorithm. We want to show that $\neg \text{SafeToAnswer}(w, \{y_1, \dots, y_r\})$, i.e., we want to show that

$$\Pr_{f',v'}[g^{(f')}(v') = w | B'' \subseteq \text{QueryY}(g, v', f')] \geq 2^{-m + \frac{n}{30}} \quad (55)$$

for some $B'' \subseteq \{y_1, \dots, y_r\}$. Of course, it suffices to show this for $B'' = B'$.

To see that (55) holds for $B'' = B'$ we compute

$$\Pr_{f',v'}[g^{(f')}(v') = w | B' \subseteq \text{QueryY}(g, v', f')] \quad (56)$$

$$= \frac{\Pr_{f',v'}[g^{(f')}(v') = w \wedge B' \subseteq \text{QueryY}(g, v', f')]}{\Pr_{f',v'}[B' \subseteq \text{QueryY}(g, v', f')]} \quad (57)$$

$$= \frac{\Pr_{v',y_1,\dots,y_r}[g'(v', y_1, \dots, y_r) = w \wedge B' \subseteq \{y_1, \dots, y_r\}]}{\Pr_{y_1,\dots,y_r}[B' \subseteq \{y_1, \dots, y_r\}]} \quad (58)$$

$$\geq \frac{2^{(r-|B'|)n + \frac{2n}{30}} 2^{-m-rn}}{\binom{r}{|B'|} (|B'|!) 2^{-(|B'|)(n-1)}} \quad (59)$$

$$\geq \frac{2^{-m-|B'|n + \frac{2n}{30}}}{2^{r \log(r)} 2^{-|B'|(n-1)}} = 2^{-m + \frac{2n}{30} - r \log(r) - r} \geq 2^{-m + \frac{n}{30}}, \quad (60)$$

where in (58) and afterwards, y_1, \dots, y_r are picked uniformly from $\{0, 1\}^n$, but without repetition. The numerator in (59) can then be seen as follows: first, note that the probability only decreases if one picks the y_i with repetition, but additionally requires them to be different for the event to occur. After that, one notices that there must be at least $2^{(r-|B'|)n + \frac{2n}{30}}$ tuples (v, y_1, \dots, y_r) for which E_{B,w,y^*} gets increased in the algorithm UpperBound. The denominator follows by noting that we can first choose how to make the assignment of the values in B' to the elements (y_1, \dots, y_r) (there are $\binom{r}{|B'|} (|B'|!)$ possibilities for this), and then checking whether this assignment occurs, which happens with probability at most $2^{-|B'|(n-1)}$. Thus, we get that $\neg \text{SafeToAnswer}(w, \{y_1, \dots, y_r\})$ must hold for any tuple where E_{B,w,y^*} is increased, which is the required contradiction. \square

6.4 Kim-Vu style concentration

In a fundamental paper [KV00], Kim and Vu consider low degree polynomials P in variables x_1, \dots, x_ℓ , and show that if $\partial_B P$ can be bounded (as in Lemma 20), then P will be concentrated around its expectation, assuming the variables x_i are picked independently at random.

Because in our case the variables are not picked independently, we need to use a different bound (other than that, the original Kim-Vu bound would be strong enough for our purpose). The bound we use requires the following concept of almost independence.

Definition 21. A distribution P_x over $\{0, 1\}^\ell$ is (δ, m) -almost independent if for all sets M of size $|M| < m$ and any $j \notin M$

$$\Pr_{x \leftarrow P_x} [x_j = 1 | \forall i \in M : x_i = 1] \leq \Pr_{x \leftarrow P_x} [x_j = 1] (1 + \delta) \quad (61)$$

We use the following bound, which is proven in [Hol11]. It uses a technique first used implicitly by [SSS95] and which was later used in [Rao08] to prove concentration bounds for parallel repetition, and by [IK10] to prove constructive concentration results.

For a polynomial P in variables x_j , and a distribution P_x over these variables, we let P_x^* be the distribution obtained by picking each x_j independently of the others, but with the marginal distribution given by P_x . We then set $\mu^* = \mathbb{E}_{x \leftarrow P_x^*} [P(x)]$ and $E^* = \max_{\emptyset \subsetneq B \subseteq \{x_1, \dots, x_\ell\}} \mathbb{E}_{x \leftarrow P_x^*} [\partial_B P(x)]$.

Theorem 22. Let P_x be an (δ, rm) -almost independent distribution over $\{0, 1\}^\ell$. Let $P(x)$ be a polynomial of degree at most r in the variables x_i , i.e., $P(x) = \sum_{j=1}^n v_j$ with $v_j = \prod_{i \in e_j} x_i$, where $|e_j| \leq r$.

Then,

$$\Pr_{x \leftarrow P_x} [P(x) \geq \mu^* (1 + \epsilon)] \leq \left(\frac{(1 + \delta)^r (1 + \frac{r^r m^r E^*}{\mu^*})}{1 + \epsilon} \right)^m. \quad (62)$$

Using this bound, we can now prove Lemma 14.

Proof (of Lemma 14). We use Theorem 22 on the polynomial P_{w,y^*} , where we set $\delta = 1$, $\epsilon = 2^{\frac{9n}{100}} / \mu^*$ and $m = 2^{\frac{n}{100r}}$. We note first that indeed the random variables $F_{(i,j)}$ are (δ, rm) -independent: conditioning on $F_{(x,y)} = 1$ is the same as conditioning on $f(x) = y$, and so we can see that one needs to condition on at least 2^{n-1} such events in order to double the probability that $F_{(x,y)} = 1$ for any (x, y) .

Thus, Theorem 22 yields:

$$\Pr_{f \leftarrow \mathcal{P}} [P_{w,y^*}(f) \geq \mu^* + 2^{\frac{9n}{100}}] \leq \left(\frac{2^r \max(2, \frac{2^r r^r 2^{\frac{n}{100}} E^*}{\mu^*})}{\frac{2^{\frac{9n}{100}}}{\mu^*}} \right)^{2^{n/100r}} \quad (63)$$

$$\leq \max \left(\frac{2 \cdot 2^r \mu^*}{2^{\frac{9n}{100}}}, \frac{2 \cdot 2^r r^r 2^{\frac{n}{100}} E^*}{2^{\frac{9n}{100}}} \right)^{2^{n/100r}} \quad (64)$$

$$\leq \left(\frac{1}{2} \right)^{2^{n/100r}}, \quad (65)$$

where we applied Lemma 20 to bound both μ^* and E^* in the last step. An application of Lemma 19 finishes the proof. \square

7 Non-uniform reductions and superpolynomial security

Theorem 5 excludes the existence of a *uniform* black-box reduction constructing a pseudorandom generator from a one-way function with few calls. Potentially, one way to overcome this lower bound would be to give a non-uniform security reduction, in which case the result would be weaker, but still very interesting. Such non-uniform construction can be excluded by the techniques given in [GGKT05], and we apply their technique here to prove that our lower bound applies to non-uniform constructions as well.

Furthermore, we also generalize our results to one-way functions with different security.

Definition 23. 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 $g_k : \{0, 1\}^{m(k)} \rightarrow \{0, 1\}^{m'(k)}$ 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

$$\Pr_{v, B}[B(k, g_k(v)) = 1] - \Pr_{w, B}[B(k, w) = 1] \geq \frac{1}{100} \quad (66)$$

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

$$\Pr_{x, A}[A^{(B, f)}(k, h_k, f_k(x)) \text{ inverts } f_k] > \frac{1}{s(k)} \quad (67)$$

for infinitely many k .

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

In an actual reduction, one would of course expect that it works given a much weaker condition than (66). In particular, a reasonable reduction will invert f with some probability if the constant $\frac{1}{100}$ is replaced by any polynomial. Excluding constructions which even adhere to Definition 23 is of course then stronger.

7.1 BreakOW does not non-uniformly invert

We first show that no non-uniform oracle algorithm with access to BreakOW inverts a random permutation f .

Lemma 24. Let $g^{(\cdot)} : \{0, 1\}^m \rightarrow \{0, 1\}^{m^*}$ be an r -query normalized oracle construction, $\frac{n}{100r} \in \mathbb{N}$. Fix an oracle function $C^{(\text{BreakOW}, f)}(y)$ making at most $q < 2^{\frac{n}{10}}$ queries to its oracles. Let \mathcal{W} be the set which contains all permutations $f \in \mathcal{P}$ for which both

$$\Pr_{y \leftarrow \{0, 1\}^n} [C^{(\text{BreakOW}, f)}(y) = f^{-1}(y)] \geq 2^{-\frac{n}{20}} \quad , \text{ and} \quad (68)$$

$$\forall w, y^* : |Q_{f, w, y^*}| \leq 2^{\frac{n}{10}} \quad (69)$$

holds. Then, $\frac{|\mathcal{W}|}{|\mathcal{P}|} \leq 2^{-2^{n/2}}$.

Proof. As in [GGKT05], we find an encoding of f which is $2^{\frac{n}{2}}$ bits shorter than $\log(2^n!)$ (the minimal length of a bitstring needed to describe an arbitrary permutation on 2^n elements). The

encoding has the property that f can be recovered (given C) from it. Actually, it is somewhat easier to describe the encoding simply as a injective function \mathcal{F} mapping onto a set with fewer than $(2^n!)2^{-2^{n/2}}$ elements, which is of course equivalent.

Fix some function f which satisfies both (68) and (69). We first find a large subset S of the images of f , which has the property that \mathcal{F} does not need to describe how f maps elements of $f^{-1}(S)$ to S , and yet \mathcal{F} will be injective. For this, we first modify C such that whenever it queries $v = \text{BreakOW}^{(f)}(w)$, it afterwards evaluates $g^{(f)}(v)$ on the result (unless BreakOW returned \perp). Then, the following algorithm outputs S .

Algorithm BuildSets(f)

Modify C as in the text

$I := \{y : C^{(\text{BreakOW}, f)}(y) = f^{-1}(y)\}$

while $I \neq \emptyset$ **do**

$y^* \leftarrow I$

 //An arbitrary element of I

$S := S \cup \{y^*\}$

 Let Q be the answers of f to the queries done by $C^{(\text{BreakOW}, f)}(y^*)$.

for $x^* \in \{0, 1\}^n$ **do**

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

if there is w such that $C^{(\text{BreakOW}, f)}(y^*)$ calls $\text{BreakOW}^{(f)}(w)$ and $\text{BreakOW}^{(f)}(w) \neq \text{BreakOW}^{(f^*)}(w)$ **then**

$Q := Q \cup \text{QueryY}(g, \text{BreakOW}^{(f^*)}(w), f)$

$I := I \setminus (Q \cup \{y^*\})$

return S

We show that $|S| \geq s := \frac{2^{19n/20}}{2^{n/10}2^{n/5} + r2^{n/10} + 1} \geq 2^{\frac{3n}{5}}$. First, from (68) we see that $|I| \geq 2^{\frac{19n}{20}}$. We claim that for each y , Q has size at most $|Q| \leq 2^{\frac{n}{10}}2^{\frac{n}{5}} + r2^{\frac{n}{10}}$ when it is removed from I . We get this since C makes at most $2^{\frac{n}{10}}$ calls to $\text{BreakOW}(w)$, and Lemma 15 implies that for each of these calls, there can be at most $2^{\frac{n}{5}}$ elements x^* for which $\text{BreakOW}^{(f^*)}(w) \neq \text{BreakOW}^{(f)}(w)$. Further, g makes at most $r2^{\frac{n}{10}}$ calls to f (due to our modification above this is a bit larger than $2^{\frac{n}{10}}$).

Let now $S' \subseteq S$ be some subset of size s , set $t = 2^n - s$, and let x_0, \dots, x_{t-1} be the elements of $\{0, 1\}^n$ which are *not* preimages of elements in S' , in lexicographic order. We show in the next paragraph that the map \mathcal{F} which maps $f \mapsto (x_0, f(x_0), \dots, x_{t-1}, f(x_{t-1}))$ is injective. The number of possible images can be counted by first considering the possible sets $\{x_0, \dots, x_{t-1}\}$ and $\{f(x_0), \dots, f(x_{t-1})\}$ (there are $\binom{2^n}{t} = \binom{2^n}{s}$ of those) and then considering the $t!$ permutations from the first to the second set, which shows that

$$\frac{|\mathcal{W}|}{|\mathcal{P}|} \leq \binom{2^n}{s}^2 \frac{(2^n - s)!}{2^n!} = \binom{2^n}{s} \frac{1}{s!} \leq \left(\frac{e^2 2^n}{s^2}\right)^s \leq 2^{-s}. \quad (70)$$

It remains to show that the map is injective. To see this, suppose that $f_1 \neq f_2$ satisfy $\mathcal{F}(f_1) = \mathcal{F}(f_2)$. Then, $f_1^{-1}(y) = f_2^{-1}(y)$ for all y for which $f_1^{-1}(y) \neq C^{(\text{BreakOW}, f_1)}(y)$ (the pair $(f_1^{-1}(y), y)$ appears in $\mathcal{F}(f_1)$, and so it must also appear in $\mathcal{F}(f_2)$). Since this holds analogous for f_2 , there must be (x_1, x_2, y) such that $x_1 = C^{(\text{BreakOW}, f_1)}(y) = f_1^{-1}(y) \neq f_2^{-1}(y) = C^{(\text{BreakOW}, f_2)}(y) = x_2$.

Since the two answers of $C(y)$ differ in the two runs, there must be some call of C to an oracle with the same input, but for which the two answers differ. This cannot be a call to the oracle f_1 or f_2 , as otherwise this call would appear in $\mathcal{F}(f_1)$ and in $\mathcal{F}(f_2)$. Thus, it must be that some for some w we have $v_1 := \text{BreakOW}^{(f_1)}(w) \neq \text{BreakOW}^{(f_2)}(w) =: v_2$, and $\text{BreakOW}(w)$ is actually called by

C in both experiments. Suppose without loss of generality that v_1 occurs first in the enumeration within BreakOW. Then, one of the queries which $g^{(f_1)}(v_1)$ does must have answer y , as otherwise all elements of $\text{Query}(g, v_1, f_1)$ would appear in $\mathcal{F}(f_1)$, and so $\text{BreakOW}(w)^{(f_2)}(w) = v_1$ as well. Thus, one of the answers was y , and since the other answers appear in $\mathcal{F}(f_1)$, f_2 and f_1 behave the same for these answers. But this implies that $\text{BreakOW}^{(f_2^*)}(w) \neq \text{BreakOW}^{(f_2)}(w)$, where $f_2^* = (f_2)_{x_1, y}$, and so $(x_1, f_2(x_1))$ must appear in $\mathcal{F}(f_2)$, which contradicts $\mathcal{F}(f_1) = \mathcal{F}(f_2)$. \square

7.2 Non-uniform black-box separation

We can now prove Theorem 6, which we restate for convenience.

Theorem 6. *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 $f(k, \cdot)$.*

Proof. As previously, we use Theorem 9 and thus assume we have a non-uniform fully black-box reduction which yields a pseudouniform one-way function.

Thus, we suppose we are given (g, A) . Again we set $n(k) := n'(k) := k$, and let $m(k)$ be the input length of g as provided by the reduction. Let $r(k)$ the number of calls to f . As before we assume that $\frac{n(k)}{100r(k)} \in \mathbb{N}$, and modify g so it is normalized.

For all k with $r(k) \geq \frac{n(k)}{1000 \log(s(k))}$ we let Breaker_k be the function which always outputs 0 and f_k a permutation which is one-way against circuits of size $2^{\frac{n}{2}}$, which exists by [GGKT05].

Otherwise, we consider $p(g_k)$. If $p(g_k) \geq \frac{1}{2}$ we set Breaker_k to be BreakOW. Lemma 12 again implies that BreakOW helps to invert g , but now we apply Lemma 24 and the union bound to get a function which is hard to invert for all h_k .

If $p(g_k) \leq \frac{1}{2}$, Lemma 17 gives a set $W_k \subseteq \{0, 1\}^{m(k)}$ and $\mathcal{Y}_k \subseteq \{0, 1\}^{n(k)}$ for which the output of $g^{(f)}$ is likely distinguished from uniform by BreakPU.

Writing the function f as $f = \pi \circ p$ for some a random permutation π on \mathcal{Y} and a regular function $p : \{0, 1\}^n \rightarrow \mathcal{Y}$ we can apply Theorem 1 of [GGKT05] (which also holds if the circuit has oracle gates to BreakPU). We can thus find a function which is hard for A and any advice string h_k , and yet the output $g^{(f)}$ will be distinguished from uniform. \square

8 Non-security parameter restricted constructions

8.1 Fixing the polynomial in the construction

Suppose that we have given a black-box construction (g, A) of a pseudouniform one-way function from a one-way function together with its security reduction. The requirement on the efficiency of the construction is that for every choice of $(f, \text{Breaker})$, both g and A should run in polynomial time. In other words, for any $(f, \text{Breaker})$ there should be $c \in \mathbb{N}$ such that $f(k, \cdot)$ and $\text{Breaker}(k, \cdot)$ run in time k^c . Note that c can depend on f and Breaker.

There do exist constructions (g, A) which are polynomial for any $(f, \text{Breaker})$, but where c indeed depends inherently on the oracle.¹¹ However, it turns out that it is always possible to fix

¹¹ An example follows: suppose the function $g(k, v)$ first queries $f(0, \mathbf{0}), f(1, \mathbf{0}), \dots, f(\log(\ell), \mathbf{0})$. If all answers were the 0-string, execute some algorithm which runs in linear time. Otherwise, let c' be the index of the first answer which differs, and execute an algorithm which runs in time $k^{c'+1}$.

finitely many outputs of the oracles f and Breaker such that after fixing these, c is independent of the choice of the remaining positions.

A prefix $(f^*, \text{Breaker}^*)$ is simply the truth table for these oracles for lengths up to some integer k_0 ; oracles $(f, \text{Breaker})$ agree with the prefix if their truth table up to length k_0 equals the one given by the prefix. A prefix $(f^{(2)}, \text{Breaker}^{(2)})$ extends a prefix $(f^{(1)}, \text{Breaker}^{(1)})$ if the truth table of $(f^{(2)}, \text{Breaker}^{(2)})$ is larger than the truth table of $(f^{(1)}, \text{Breaker}^{(1)})$, and they agree everywhere where $(f^{(1)}, \text{Breaker}^{(1)})$ is defined.

Lemma 25. *Suppose a black-box reduction $(g^{(f)}, A^{(\text{Breaker}, f)})$ is given, and fix some length function $n(k)$. There exists a prefix $(f^*, \text{Breaker}^*)$ and $c \in \mathbb{N}$ such that for any pair $(f, \text{Breaker})$ which agrees with $(f^*, \text{Breaker}^*)$ we have the following properties:*

1. $g^{(f)}(\ell, \cdot)$ makes at most ℓ^c queries to $f(k, \cdot)$, and all of these queries satisfy $k \leq \ell^c$
2. $A^{(\text{Breaker}, f)}(k, w)$ makes at most k^c queries to $\text{Breaker}(\ell, \cdot)$, and all of these queries satisfy $\ell \leq k^c$
3. $A^{(\text{Breaker}, f)}(k, w)$ makes at most k^c queries to $f(k', \cdot)$, and all of these queries satisfy $k' \leq k^c$.

Proof. Suppose not, let $d \in \mathbb{N}$, and suppose we have given any prefix $(f^{(d)}, \text{Breaker}^{(d)})$. Then, there exists a pair $(f, \text{Breaker})$ of oracles which agree with $(f^{(d)}, \text{Breaker}^{(d)})$ and where one of 1, 2, or 3 is violated for $c = (d + 1)$. Fix a length k for which this is violated, and find a prefix $(f^{(d+1)}, \text{Breaker}^{(d+1)})$ of $(f, \text{Breaker})$ such that all queries done for up to security parameter k are fixed in the prefix.

Thus, there is an infinite sequence of prefixes $\{(f^{(i)}, \text{Breaker}^{(i)})\}_{i \geq 0}$ such that $(f^{(i+1)}, \text{Breaker}^{(i+1)})$ extends $(f^{(i)}, \text{Breaker}^{(i)})$, and for any $d \in \mathbb{N}$ there is an input which violates one of the conclusions of the lemma.

Clearly, such an infinite sequence defines a pair $(f, \text{Breaker})$ for which (g, A) is not polynomial. \square

8.2 Excluding general reductions

We now come to the proof of Theorem 7, which we restate for convenience.

Theorem 7. *Fix a length function $n(k)$. Let (g, A) be a fully black-box construction of a pseudo-random generator from a regular one-way function. Then, there is an oracle f for which*

$$r_f \in \Omega\left(\frac{n_f^-}{\log(n_f^-)}\right). \quad (7)$$

Preparations for the proof As before, we prove the analogous statement for pseudouniform one-way functions. Also, we assume that the theorem is not true, and that we have given (g, A) , and show that we can find oracles $(f, \text{Breaker})$ which contradict the assumption that (g, A) is a fully black-box construction.

We can assume that $g(\ell, v)$ never queries $f(k, x)$ twice for any (k, x) . Also, we use Lemma 25, which fixes a prefix for $(f, \text{Breaker})$ and gives us a constant c for which the properties in Lemma 25 are satisfied, we will use this constant throughout the proof.

We now choose k_0 and set $\ell_0 = k_0^c$ such that neither $f(k_0, \cdot)$ nor $\text{Breaker}(\ell_0, \cdot)$ has been defined by Lemma 25. Furthermore, define all oracles $\text{Breaker}(\ell, \cdot)$ for $\ell < \ell_0$ which have not been defined yet to oracles which do nothing (i.e., constantly output \perp). Analogously, define all oracles $f(k, \cdot)$ for $k < k_0$ which have not been defined yet to random permutations of length $n(k)$.

Next, we pick a constant \tilde{c} for later. We require that it satisfies that for any $k \in \{\ell^{1/c}, \dots, \ell^c\}$ we have $\ell^{1/\tilde{c}} \leq n(k) \leq \ell^{\tilde{c}}$; this is possible because $n(k)$ is a length function.

Overview and some basics of the proof In the main part of the proof, we define the oracles $\text{Breaker}(\ell, \cdot)$ and $f(k, \cdot)$. We essentially use one iteration for each ℓ , and increase ℓ over time. At the beginning of iteration ℓ , we will have defined the oracles $\text{Breaker}(1, \cdot), \dots, \text{Breaker}(\ell - 1, \cdot)$ and $f(k, \cdot)$ for any $k < \ell^{1/c}$.

At this point, we enumerate each n which is possibly the length of the shortest query made by $g(\ell, \cdot)$, ignoring the length of those for which $f(k, \cdot)$ has been defined already.

For each such n , we consider the probability

$$q_{\ell,n} := \Pr_{v,f} \left[g^{(f)}(\ell, v) \text{ queries } f \text{ on security parameters } k \geq \ell^{1/c} \text{ a total of at most } \frac{n}{d \log(n)} \text{ times, and for all these queries } f(k, \cdot) \text{ we have } n(k) \geq n \right]$$

where $f(k, \cdot)$ is chosen as random permutation for any $k \geq \ell^{1/c}$. The parameter d will be defined later, and is slowly growing as $\ell \rightarrow \infty$.

We then distinguish two cases: The first case is if $q_{\ell,n} \leq \ell^{-\tilde{c}-2}$ for all n .

In this case, we define $\text{Breaker}(\ell, \cdot) := \perp$, so that it does nothing on this length, increase ℓ , and go to the next iteration. We will show that infinitely often $q_{\ell,n}$ must be larger than $\ell^{-\tilde{c}-2}$ for some n , as otherwise we can obtain an oracle $(f, \text{Breaker})$ for which $r_f \in \Omega(n_f^- / \log(n_f^-))$.

The second case is more interesting: there is \tilde{n} for which $q_{\ell,\tilde{n}} > \ell^{-\tilde{c}-2}$.

In this case, we know that with some polynomial probability, $g(\ell, \cdot)$ will only make few queries. We would like to apply the previous machinery, but cannot do so directly: g possibly makes more than $\tilde{n}/d \log(\tilde{n})$ many queries for some oracle f , and possibly queries f on input lengths shorter than \tilde{n} for some oracle f .

Also (and this is the problem we fix first), the previous machinery only allows g to make queries to one fixed input length \tilde{n} , whereas g may query f with many different parameters k for which $n(k) \geq \tilde{n}$.

To solve this, we use the following idea: underlying to $f(k, \cdot)$ could in fact be a *single* one-way function $\tilde{f} : \{0, 1\}^{\tilde{n}} \rightarrow \{0, 1\}^{\tilde{n}}$ for many different values of k , so that $f(k, x) = S_k(\tilde{f}(P_k(x)))$ for some simple to compute projection $P_k : \{0, 1\}^{n(k)} \rightarrow \{0, 1\}^{\tilde{n}}$ and some expansion $S_k : \{0, 1\}^{\tilde{n}} \rightarrow \{0, 1\}^{n(k)}$.

Thus, we pick uniform random injective functions P_k and uniform injective expansions S_k for each k for which $n(k) \geq \tilde{n}$. We then consider the construction $\tilde{g}_n^{(\tilde{f}, f, P, S)}$. This construction is defined as follows:

The function $\tilde{g}_n^{(\tilde{f}, f, P, S)}$ simulates g , except whenever g calls the oracle f .

In case g calls $f(k, x)$ for some k with $k < \ell^{1/c}$, the answer of $f(k, x)$ is hard-coded into \tilde{g} (because f is already defined on these lengths).

If g calls $f(k, x)$ for some k with $n(k) < \tilde{n}$ and $k \geq \ell^{1/c}$, then \tilde{g} calls $f(k, x)$ as well.

In case g calls $f(k, x)$ for some k with $n(k) \geq \tilde{n}$, and $k \geq \ell^{1/c}$, \tilde{g}_n instead calls $S_k(\tilde{f}(P_k(x)))$.

The function \tilde{g} behaves almost as g when \tilde{f} is chosen as a random permutation. The only exception is in the unlikely case that $P_k(x) = P_k(x')$ for two queries $(k, x) \neq (k, x')$ to f .

The function \tilde{g} solves the last problem above, so that we get closer to apply the previous machinery. However, \tilde{g} still can make more than $\tilde{n}/d \log(\tilde{n})$ queries to f or query f on shorter

inputs than \tilde{n} for some f . Thus, we consider the construction \tilde{h} , which is just like \tilde{g} , but has additional restrictions:

Whenever \tilde{g} does more than $\tilde{n}/d \log(\tilde{n})$ calls, \tilde{h} simply stops and outputs (v, \perp) .

Whenever g does a call to $f(k, \cdot)$ with $n(k) < \tilde{n}$ and $k \geq k_0$, \tilde{h} stops and outputs (v, \perp) .

We note that \tilde{h} does not need to call the oracle f at any time anymore.

As long as $d \rightarrow \infty$ for $\ell \rightarrow \infty$, the results from the previous sections will guarantee that breaking the pseudouniformity of \tilde{h} does not help inverting \tilde{f} . The main difficulty is that \tilde{h} may behave very differently from \tilde{g} . However, we can note that

$$\Pr_{\tilde{f}, P, S, v, f} [\tilde{h}^{\tilde{f}, P, S}(v) = \tilde{g}^{\tilde{f}, P, S}(v)] \geq q_{\ell, \tilde{n}} - \frac{\tilde{n}^2}{2\tilde{n}} \quad (71)$$

because as long as no two queries to $P_k(\cdot)$ collide in the evaluation of \tilde{g} , each query will be answered with a uniform random answer, and so g and \tilde{g} will behave exactly the same.

We are now interested in the probability that BreakOW inverts a random image of \tilde{g} . To apply the previous machinery, we want to instantiate BreakOW *using* \tilde{h} . Thus, we consider the probability

$$p_{\ell, \tilde{n}} := \Pr_{\tilde{f}, P, S, v, f} [\text{SafeToAnswer}_{\tilde{h}}(\tilde{h}^{\tilde{f}, P, S}(v), \text{QueryY}(\tilde{h}, \tilde{f}, v)) \wedge \tilde{h}^{\tilde{f}, P, S}(v) = \tilde{g}^{\tilde{f}, P, S}(v)] \quad (72)$$

Here, SafeToAnswer is instantiated using \tilde{h} instead of g .¹²

We will then show that we can do a similar case distinction $p_{\ell, \tilde{n}}$ as we did in the previous sections on $p(g)$. This will allow us to build oracles $(f(k, \cdot), \text{Breaker}(\ell, \cdot))$ where $\text{Breaker}(\ell, \cdot)$ breaks the construction on this length.

After this, we set $\text{Breaker}(\ell', \cdot) := \perp$ for $\ell < \ell' < \ell^{c^2}$, which ensures that there is no problem because different lengths are interfering with each other. We then go to the next iteration for which $\text{Breaker}(\ell, \cdot)$ is not yet defined.

Building the oracles We now describe a randomized procedure which builds oracles f and Breaker by building a sequence of extending prefixes (as in the proof of Lemma 25). After this, we prove that the oracle arising from this sequence has the required properties with probability 1.

Algorithm GenerateOracles

Fix Breaker and f up to some length using Lemma 25, then ensure that

$f(k, \cdot)$ is defined up to security parameter k_0 for some k_0 , and that

Breaker(ℓ, \cdot) is defined up to k_0^c .

$d := 1$

$\ell :=$ smallest ℓ for which Breaker(ℓ, \cdot) has not yet been defined

do forever

 // We define Breaker(ℓ, \cdot) in this iteration

if $\forall n \in \{\ell^{1/\tilde{c}}, \dots, \ell^{\tilde{c}}\}$: $q_{\ell, n} \leq \ell^{-(\tilde{c}+2)}$ **then**

¹²Strictly speaking, to instantiate SafeToAnswer we should give it a function \tilde{h} which only uses the oracle \tilde{f} , but not oracles P and S . For that purpose, one can think of P and S as being hardcoded into \tilde{h} .

```

Breaker( $\ell, \cdot$ ) :=  $\perp$  //Breaker will not help on this length
if  $\exists k \in \mathbb{N} : k^c = \ell$  then
     $f(k, \cdot) \leftarrow \Pi_{n(k)}$  //A random permutation on  $n(k)$  bits
else
    let  $\tilde{n}$  be such that  $q_{\ell, \tilde{n}} > \ell^{-(\tilde{c}+2)}$ 
    if  $p_{\ell, \tilde{n}} \leq \frac{1}{2} \ell^{-(\tilde{c}+2)}$  then
        try at most  $\ell^{\tilde{c}+3}$  times
            BuildPUBreaker( $\ell, \tilde{n}$ )
        stop if Breaker( $\ell, \cdot$ ) has distinguishing advantage at least  $\ell^{-(\tilde{c}+3)}$ ,
            otherwise roll back the changes and try the loop again
    else
        try at most  $\ell^{\tilde{c}+3}$  times
            BuildOWBreaker( $\ell, \tilde{n}$ )
        stop if Breaker( $\ell, \cdot$ ) inverts  $g$  with probability at least  $\ell^{-(\tilde{c}+3)}$ ,
            otherwise roll back the changes and try the loop again
    fi
     $d := d + 1$ 
     $\ell := \ell + 1$ 

```

```

procedure BuildPUBreaker( $\ell, \tilde{n}$ )
     $r := \tilde{n}/d \log(\tilde{n})$ 
    Pick  $\mathcal{Y} \subseteq \{0, 1\}^{\tilde{n}}$ ,  $|\mathcal{Y}| = 2^{\tilde{n}/100r}$  u.a.r.
    for each  $k \in \{\ell^{1/c}, \dots, \ell^c\}$  with  $n(k) \geq \tilde{n}$  do
        pick a regular function  $P_k : \{0, 1\}^{n(k)} \rightarrow \{0, 1\}^{\tilde{n}}$  u.a.r.
        pick an injective function  $S_k : \{0, 1\}^{\tilde{n}} \rightarrow \{0, 1\}^{n(k)}$  u.a.r.
    // At this point,  $\tilde{h}$  and  $\mathcal{Y}$  are defined
    Obtain  $W$  (using  $\tilde{h}$  as underlying function) as in Lemma 17.
    Breaker( $\ell, \cdot$ ) := BreakPU( $W$ )
    Pick a regular function  $\tilde{f} : \{0, 1\}^{\tilde{n}} \rightarrow \mathcal{Y}$  u.a.r.
    for each  $k \in \{\ell^{1/c}, \dots, \ell^c\}$  do
        if  $n(k) \geq \tilde{n}$  then
             $f(k, \cdot) := S_k \circ \tilde{f} \circ P_k$ 
        else
             $f(k, \cdot) \leftarrow \Pi_{n(k)}$ 
    for  $\ell' \in \{\ell + 1, \dots, \ell^{c^2}\}$  do
        Breaker( $\ell', \cdot$ ) :=  $\perp$ 
     $\ell := \ell^{c^2} + 1$ 

```

```

procedure BuildOWBreaker( $\ell, \tilde{n}$ )
     $r := \tilde{n}/d \log(\tilde{n})$ 
    for each  $k \in \{\ell^{1/c}, \dots, \ell^c\}$  with  $n(k) \geq \tilde{n}$  do
        pick a regular function  $P_k : \{0, 1\}^{n(k)} \rightarrow \{0, 1\}^{\tilde{n}}$  u.a.r.
        pick an injective function  $S_k : \{0, 1\}^{\tilde{n}} \rightarrow \{0, 1\}^{n(k)}$  u.a.r.
    // At this point,  $\tilde{h}$  is defined
    Breaker( $\ell, \cdot$ ) := BreakOW $_{\tilde{h}}^{(\tilde{f})}(\cdot)$ 
    Pick a permutation  $\tilde{f} : \{0, 1\}^{\tilde{n}} \rightarrow \{0, 1\}^{\tilde{n}}$  u.a.r.
    for each  $k \in \{\ell^{1/c}, \dots, \ell^c\}$  do

```

```

if  $n(k) \geq \tilde{n}$  then
     $f(k, \cdot) := S_k \circ \tilde{f} \circ P_k$ 
else
     $f(k, \cdot) \leftarrow \Pi_{n(k)}$ 
for  $\ell' \in \{\ell + 1, \dots, \ell^{c^2}\}$  do
    Breaker( $\ell', \cdot$ ) :=  $\perp$ 
 $\ell := \ell^{c^2+1}$ 

```

Clearly, GenerateOracles defines an infinite sequence of prefixes $(f^{(i)}, \text{Breaker}^{(i)})$, and as before we can extend that to a single oracle $(f, \text{Breaker})$. Analogously, we can extend events which are defined on prefixes to an infinite sequence of events.

We next explain how these procedures make their random choices. For this we assume that for each $k \in \mathbb{N}$, a permutation $\tilde{f}(k, \cdot)$ is picked. Whenever GenerateOracles executes the assignment $f(k, \cdot) \leftarrow \Pi_{n(k)}$ (in the part where it defines $\text{Breaker}(\ell, \cdot) := \perp$, it assigns $f(k, x) := \tilde{f}(k, x)$). We can imagine these permutations to be picked before GenerateOracles is executed (in that way, we can talk about future assignments).

Also, for each ℓ and each $k \in \{\ell^{1/c}, \dots, \ell^c\}$, we pick ℓ^{c+3} choices for S_k, P_k . Also, for each possible \tilde{n} we pick ℓ^{c+3} choices for \tilde{f} . Then, in the i th iteration, we simply assume that the i th such choice is used. As with \tilde{f} , this is useful in order to argue about future assignments.

Lemma 26. *Consider an execution of the algorithm GenerateOracles. For each ℓ , let N_ℓ be the event that the else clause of algorithm GenerateOracles is executed on iteration ℓ .*

Then, with probability 1, infinitely many events N_ℓ occur.

Proof. We let d_ℓ be the random variable which takes the value of d in the ℓ th iteration of GenerateOracles.

For each ℓ and each $n \in \{1, \dots, \ell^c\}$ we now define an event $B_{\ell,n}$. For this event, we stop the normal execution of GenerateOracles at loop ℓ , and instead extend f using \tilde{f} exclusively (i.e., fill everything with the random permutations we picked before). We then let $B_{\ell,n}$ be the event which occurs if $r_f \leq n/(d_\ell \log(n))$ and $n_f^- = n$ in this extension. Clearly $\Pr[B_{\ell,n} \wedge \neg N_\ell] \leq \Pr[B_{\ell,n} | \neg N_\ell] \leq \ell^{-(\tilde{c}+2)}$, because in case we extend with random permutations, $q_{\ell,n}$ is defined exactly as the probability that the event $B_{\ell,n}$ occurs.

Thus, $\sum_{\ell, n \leq \ell^c} \Pr[B_{\ell,n} \wedge \neg N_\ell] < \infty$, and so by the Borel-Cantelli lemma, $(B_{\ell,n} \wedge \neg N_\ell)$ happens for infinitely many ℓ with probability 0.

Now, suppose that in some execution only finitely many events $B_{\ell,n}$ happen. Then we found an oracle for which $r_f \in \Omega(n_f^- / \log(n_f^-))$, because in this case we *do* extended only using \tilde{f} starting from some fixed length.

Therefore, in all executions infinitely many events $B_{\ell,n}$ happen, and so the event N_ℓ must happen for infinitely many ℓ with probability 1. \square

Lemma 27. *Suppose that $q_{\ell, \tilde{n}} > \ell^{-(\tilde{c}+2)}$, $p_{\ell, \tilde{n}} \leq \frac{1}{2} \ell^{-(\tilde{c}+2)}$, $d > 2c$, and ℓ is larger than some constant in an execution of the loop in algorithm GenerateOracles.*

Then, with probability at least $\frac{1}{8} \ell^{-(\tilde{c}+2)}$, after a single call to BuildPUBreaker(ℓ, \tilde{n}), the oracle BreakPU(ℓ, \cdot) has advantage at least $\frac{1}{8} \ell^{-(\tilde{c}+2)}$ in distinguishing $g(\ell, v)$ from a uniform random string.

Proof. We first notice that $\Pr_{w \leftarrow \{0,1\}^{m(\ell)}}[\text{BreakPU}(\ell, w) = 1]$ is negligible: Lemma 17 gives a set W of size at most $2^{m(\ell) - \frac{\tilde{n}}{100}}$ and $\tilde{n} \geq \ell^{-\tilde{c}}$.

We next show that

$$\Pr_{v, \tilde{f}', f, P, S} [\tilde{g}^{\tilde{f}', f, P, S}(v) \in W] \geq \frac{1}{3} \ell^{(\tilde{c}+2)}, \quad (73)$$

where \tilde{f}' is chosen as a random function $\tilde{f}' : \{0, 1\}^{\tilde{n}} \rightarrow \mathcal{Y}$, for \mathcal{Y} of size $2^{\tilde{n}/100r}$ chosen uniformly at random. From (73) we get the lemma by applying Markov's inequality.

To see (73), we use that

$$\Pr_{v, \tilde{f}', f, P, S} [\tilde{g}^{\tilde{f}', f, P, S}(v) \in W] \geq \Pr_{v, \tilde{f}, f, P, S} [\tilde{g}^{\tilde{f}, f, P, S}(v) \in W] - \frac{r^2}{|\mathcal{Y}|},$$

where $\tilde{f} \in \mathcal{P}_{\tilde{n}}$ is a uniform random permutation on \tilde{n} bits: this follows as in the proof of Lemma 17.

We now see that

$$\begin{aligned} & \Pr_{v, \tilde{f}, f, P, S} [\tilde{g}^{\tilde{f}, f, P, S}(v) \in W] \\ & \geq \Pr_{v, \tilde{f}, f, P, S} [\tilde{h}^{\tilde{f}, P, S}(v) \in W \wedge \tilde{h}^{\tilde{f}, P, S}(v) = \tilde{g}^{\tilde{f}, f, P, S}(v)] \\ & \geq \Pr_{v, \tilde{f}, f, P, S} \left[(\neg \text{SafeToAnswer}_{\tilde{h}}(\tilde{h}^{\tilde{f}, P, S}(v), \text{QueryY}(\tilde{h}, \tilde{f}, v))) \wedge \tilde{h}^{\tilde{f}, P, S}(v) = \tilde{g}^{\tilde{f}, f, P, S}(v) \right], \end{aligned}$$

due to the definition of W in the proof of Lemma 17.

Using (71), we see that this last probability is at least $q_{\ell, \tilde{n}} - p_{\ell, \tilde{n}} - \frac{\tilde{n}^2}{2^n}$, which gives (73), and therefore the lemma. \square

Lemma 28. *Suppose that $p_{\ell, \tilde{n}} \geq \frac{1}{2} \ell^{-(\tilde{c}+2)}$. Then, with probability at least $\frac{1}{4} \ell^{-(\tilde{c}+2)}$, after a call to $\text{BuildOWBreaker}(\ell, \tilde{n})$, the oracle BreakOW will invert $g(\ell, v)$ with probability at least $\frac{1}{4} \ell^{-(\tilde{c}+2)}$.*

Proof. Consider, for fixed (\tilde{f}, f, P, S) the probability that

$$p' := \Pr_v [\text{SafeToAnswer}_{\tilde{h}}(\tilde{h}^{\tilde{f}, P, S}(v), \text{QueryY}(\tilde{h}, \tilde{f}, v)) \wedge \tilde{h}^{\tilde{f}, P, S}(v) = \tilde{g}^{\tilde{f}, f, P, S}(v)]. \quad (74)$$

We know that $p_{\ell, \tilde{n}} = \mathbb{E}_{\tilde{f}, P, S, f} [p'] \geq \frac{1}{2} \ell^{-(\tilde{c}+2)}$. Thus, with probability $\frac{1}{4} \ell^{-(\tilde{c}+2)}$, p' is at least $\frac{1}{4} \ell^{-(\tilde{c}+2)}$. Now, after BreakOW fixed \tilde{f}, f, P, S , in case $p' \geq \frac{1}{4 \ell^{\tilde{c}+2}}$, it is clear that BreakOW will invert g with this probability (because for any w which is chosen as $w = g(v)$, \tilde{h} has no preimages of w which g does not have, and BreakOW will at least find the preimage v for \tilde{h}). \square

Lemma 29. *With probability 1, the probability that $A(k, \cdot)$ inverts $f(k, \cdot)$ is a negligible function in k .*

Proof. Let $B_{k, \alpha}$ be the event that $A(k, \cdot)$ inverts $f(k, \cdot)$ with probability at least $k^{-\alpha}$. We show that for any $\alpha \in \mathbb{N}$, with probability 1, finitely many events $B_{k, \alpha}$ happen. By the Borel-Cantelli lemma it is enough to show that $\sum_k \Pr[B_{k, \alpha}] < \infty$ for any α . For this, it is clearly enough to show that $\Pr[B_{k, \alpha}]$ is a negligible function in k for any α .

To show this, we distinguish cases. First, consider the case that $f(k, \cdot)$ is picked as a random permutation in GenerateOracles , i.e., the case where where $q_{\ell, n} \leq \ell^{-(\tilde{c}+2)}$ for all n and $k^c = \ell$.

All oracles $\text{Breaker}(\ell, \cdot)$ which $A(k, \cdot)$ can possibly access are fixed before $f(k, \cdot)$ is chosen, and so we can ignore them. The same holds for all oracles $f(k', \cdot)$ for $k' \leq k$.

However, A can also access $f(k', \cdot)$ for $k < k' \leq k^c$. These are picked later, and the distribution can depend on $f(k, \cdot)$.

Luckily, there is only a polynomial number of possibilities how the functions $f(k', \cdot)$ for $k < k' \leq k^c$ will be chosen in the end. To see that, note that we can specify how all of these $f(k', \cdot)$ are chosen by specifying

- the integer $\ell \in \{k^c, \dots, k^{c^2}\}$ for which the algorithm GenerateOracles uses the else clause, in case there is one (note that there is at most one)
- the integer \tilde{n} which GenerateOracles uses in this case
- whether GenerateOracles uses BuildPUBreaker or BuildOWBreaker,
- and which of the at most $\ell^{\tilde{c}+3}$ iterations is used in the end.

Once we have specified these numbers, we see that we know which of the choices for S_k, P_k, \tilde{f} , and so on are used to pick $f(k', \cdot)$ for all these k' .

We can now simply check whether $A(k, \cdot)$ inverts $f(k, \cdot)$ with probability $k^{-\alpha}$ for *any* of these random choices. Since this probability is negligible, we apply the union bound and get the result in this case.

The same argument works in case $f(k, \cdot)$ is picked from $\Pi_{n(k)}$ in either BuildOWBreaker or BuildPUBreaker (because $n(k) < \tilde{n}$).

Thus, consider the last case where $f(k, \cdot)$ is set to $S_k \circ \tilde{f} \circ P_k$ in either BuildOWBreaker or BuildPUBreaker. Then, for any intertion we consider the breaker which tries to invert \tilde{f} by first inverting S_k , then running $A(k, \cdot)$, and then applying P_k on the result. The probability that this algorithm inverts \tilde{f} in any of the at most ℓ^c iterations of BuildPUBreaker or BuildOWBreaker is negligible (by the previous sections), and so we get the result in this case as well. \square

Finishing the proof We can now finish the proof of Theorem 7.

First, we see that with probability 1 the oracles $(f, \text{Breaker})$ generated are such that Breaker either infinitely often breaks the one-wayness or the pseudouniformity of g : first, due to Lemma 26 we see that we will infinitely often attempt to construct Breaker in one of the two ways, and by either Lemma 27 or Lemma 28 we see that the probability that this only works finitely many times is 0 (again using Borel-Cantelli). By Lemma 29 we see that f will be one-way for A , which proves the result.

9 Acknowledgements

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

References

- [BJP11] Josh Bronson, Ali Juma, and Periklis A. Papakonstantinou. Limits on the stretch of non-adaptive constructions of pseudo-random generators. In *TCC 2011*, pages 504–521, 2011.
- [BM84] Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM Journal on Computing*, 13(4):850–864, 1984.
- [DHR08] Nenad Dedic, Danny Harnik, and Leonid Reyzin. Saving private randomness in one-way functions and pseudorandom generators. In *TCC*, pages 607–625, 2008.
- [GGKT05] Rosario Gennaro, Yael Gertner, Jonathan Katz, and Luca Trevisan. Bounds on the efficiency of generic cryptographic constructions. *SIAM J. Comput.*, 35(1):217–246, 2005.
- [GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *Journal of the ACM*, 33(4):792–807, 1986.

- [GKL93] Oded Goldreich, Hugo Krawczyk, and Michael Luby. On the existence of pseudorandom generators. *SIAM Journal on Computing*, 22(6):1163–1175, 1993.
- [GL89] Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In *Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing*, pages 25–32, 1989.
- [HH09] Iftach Haitner and Thomas Holenstein. On the (im)possibility of key dependent encryption. In *TCC 2009*, pages 202–219, 2009.
- [HHR06a] Iftach Haitner, Danny Harnik, and Omer Reingold. Efficient pseudorandom generators from exponentially hard one-way functions. In *ICALP (2)*, pages 228–239, 2006.
- [HHR06b] Iftach Haitner, Danny Harnik, and Omer Reingold. On the power of the randomized iterate. In Cynthia Dwork, editor, *Advances in Cryptology — CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*, 2006.
- [HHR07] Iftach Haitner, Jonathan J. Hoch, Omer Reingold, and Gil 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*, pages 669–679, 2007.
- [HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.
- [Hol06] Thomas Holenstein. Pseudorandom generators from one-way functions: A simple construction for any hardness. In *TCC 2006*, pages 443–461, 2006.
- [Hol11] Thomas Holenstein. Some concentration bounds. Manuscript, 2011.
- [HRV10] Iftach Haitner, Omer Reingold, and Salil 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.
- [IK10] Russell Impagliazzo and Valentine Kabanets. Constructive proofs of concentration bounds. In *APPROX-RANDOM*, pages 617–631, 2010.
- [IR89] Russell Impagliazzo and Steven Rudich. Limits on the provable consequences of one-way permutations. In *Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing*, pages 44–61, 1989.
- [KSS11] Jeff Kahn, Michael Saks, and Clifford D. Smyth. The dual BKR inequality and Rudich’s conjecture. *Combinatorics, Probability & Computing*, 20(2):257–266, 2011.
- [KV00] Jeong Han Kim and Van H. Vu. Concentration of multivariate polynomials and its applications. *Combinatorica*, 20(3):417–434, 2000.
- [Lev87] Leonid A. Levin. One-way functions and pseudorandom generators. *Combinatorica*, 7(4):357–363, 1987.
- [Lu06] Chi-Jen Lu. On the complexity of parallel hardness amplification for one-way functions. In *TCC 2006*, pages 462–481, 2006.
- [MM11] Takahiro Matsuda and Kanta Matsuura. On black-box separations among injective one-way functions. In *TCC 2011*, pages 597–614, 2011.
- [MV11] Eric Miles and Emanuele Viola. On the complexity of non-adaptively increasing the stretch of pseudorandom generators. In *TCC 2011*, pages 522–539, 2011.
- [Rao08] Anup Rao. Parallel repetition in projection games and a concentration bound. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, pages 1–10, 2008.
- [RTV04] Omer Reingold, Luca Trevisan, and Salil P. Vadhan. Notions of reducibility between cryptographic primitives. In Moni Naor, editor, *TCC 2004*, volume 2951 of *Lecture Notes in Computer Science*, pages 1–20, 2004.
- [Rud88] Steven Rudich. *Limits on the Provable Consequences of One-way Functions*. PhD thesis, Berkeley, CA, 1988.
- [Sim98] Daniel R. Simon. Finding collisions on a one-way street: Can secure hash functions be based on general assumptions? In Kaisa Nyberg, editor, *Advances in Cryptology — EUROCRYPT ’98*, volume 1403 of *Lecture Notes in Computer Science*, pages 334–345, 1998.
- [SSS95] Jeanette P. Schmidt, Alan Siegel, and Aravind Srinivasan. Chernoff-hoeffding bounds for applications with limited independence. *SIAM J. Discrete Math.*, 8(2):223–250, 1995.
- [Vio05] Emanuele Viola. On constructing parallel pseudorandom generators from one-way functions. In *IEEE Conference on Computational Complexity*, pages 183–197, 2005.

- [VZ12] Salil Vadhan and Colin Jia Zheng. Characterizing pseudoentropy and simplifying pseudorandom generator constructions. In *Proceedings of the Forty-Fourth Annual ACM Symposium on Theory of Computing*, 2012. To appear.
- [Yao82] Andrew C. Yao. Theory and applications of trapdoor functions (extended abstract). In *The 23rd Annual Symposium on Foundations of Computer Science*, pages 80–91, 1982.