Limits on the Stretch of Non-Adaptive Constructions of Pseudo-Random Generators

Josh Bronson[1], Ali Juma[2], and Periklis A. Papakonstantinou[3] ⋆

[1] HP TippingPoint
josh.t.bronson@hp.com
[2] University of Toronto
ajuma@cs.toronto.edu
[3] ITCS, Tsinghua University
papakons@tsinghua.edu.cn

Abstract. The standard approach for constructing a large-stretch pseudo-random generator given a one-way permutation or given a smaller-stretch pseudo-random generator involves repeatedly composing the given primitive with itself. In this paper, we consider whether this approach is necessary, that is, whether there are constructions that do not involve composition. More formally, we consider black-box constructions of pseudo-random generators from pseudo-random generators of smaller stretch or from one-way permutations, where the constructions make only non-adaptive queries to the given object. We consider three classes of such constructions, and for each class, we give a black-box impossibility result that demonstrates a contrast between the stretch that can be achieved by adaptive and non-adaptive black-box constructions.

We first consider constructions that make constantly-many non-adaptive queries to a given pseudo-random generator, where the seed length of the construction is at most $O(\log n)$ bits longer than the length n of each oracle query. We show that such constructions cannot achieve stretch that is even a single bit greater than the stretch of the given pseudo-random generator.

We then consider constructions with arbitrarily long seeds, but where oracle queries are collectively chosen in a manner that depends only on a portion of the seed whose length is at most $O(\log n)$ bits longer than the length n of each query. We show that such constructions making constantly-many non-adaptive queries cannot achieve stretch that is $\omega(\log n)$ bits greater than the stretch of the given pseudo-random generator.

Finally, we consider a class of constructions motivated by streaming computation. Specifically, we consider constructions where the computation of each individual output bit depends only on the seed and on the response to a single query to a one-way permutation. We allow the seed to have a public portion that is arbitrarily long but must always be included in the output, and a non-public portion that is at most $O(\log n)$

⋆ P.P. is supported in part by the NSF China Grant 61050110452, 60553001, 61073174, 61033001 and the National Basic Research Program of China Grant 2007CB807900, 2007CB807901.

bits longer than the length n of each oracle query. We show that such constructions whose queries are chosen non-adaptively based only on the non-public portion of the seed cannot achieve linear stretch.

1 Introduction

It is well known that if there exist pseudo-random generators obtaining even one bit of stretch, then for every polynomial $p(n)$, there exist pseudo-random generators obtaining $p(n)$ bits of stretch. The usual approach for constructing a pseudo-random generator of large stretch from a pseudo-random generator of smaller stretch involves composing the smaller-stretch generator with itself repeatedly. Similarly, the usual approach for constructing a pseudo-random generators of large stretch from a one-way permutation involves composing the one-way permutation with itself repeatedly.

In this paper, we consider whether there exist such constructions that do *not* involve composition. To formalize this requirement about composition, we consider constructions that only have oracle access to the given object (a smaller-stretch pseudo-random generator or a one-way permutation) and query this oracle *non-adaptively*. We refer to such constructions as *non-adaptive (oracle) constructions.*

> *Given oracle access to a pseudo-random generator or a one-way permutation is it possible to construct, via non-adaptive oracle queries, a pseudo-random generator of large stretch?*

We give a number of black-box impossibility results for non-adaptive oracle constructions of pseudo-random generators. Some of these arguments are rather technically involved. Roughly speaking, we answer in the negative whether we can obtain, with only a constant number of queries to a pseudo-random generator, a pseudo-random generator of much larger stretch, where *answers to these non-adaptive queries are combined arbitrarily.* The challenge is to deal with this arbitrary computation phase.

Non-adaptive constructions are conceptually related to *streaming cryptography*; that is, computing private-key primitives with a device that uses small space and accesses the seed a small number of times. One of the three non-adaptive settings we consider in this paper is motivated by questions in streaming cryptography.

Our results. Observe that if pseudo-random generators exist, then there exist trivial non-adaptive oracle constructions of large-stretch pseudo-random generators: such constructions can simply ignore their oracle and directly compute a large-stretch pseudo-random generator. Since we are interested in constructions that use their oracle in a non-trivial way, we focus on constructions whose pseudo-randomness is proven using a *black-box reduction* [8] to the the security (pseudo-randomness or one-wayness) of their oracle.

We consider three classes of such constructions, and give bounds on the stretch that can be obtained by each class. For each class, our results demonstrate a contrast between the stretch that can be achieved by adaptive and non-adaptive constructions. We show that, in some sense, whatever was already known regarding algorithms for non-adaptive constructions is the best we can hope for. While we are primarily interested in constructions that are polynomial-time computable, our bounds hold even for computationally-unbounded constructions (where the number of oracle queries is still bounded).

– *Class 1: Constructions with short seeds*
  Suppose we have a pseudo-random generator $f : \{0,1\}^n \to \{0,1\}^{n+s(n)}$ and we wish to obtain a pseudo-random generator with larger stretch, say stretch $2 \cdot s(n)$. We can easily define such a generator $G^f : \{0,1\}^n \to \{0,1\}^{n+2\cdot s(n)}$ as follows: on input $x \in \{0,1\}^n$, G^f computes $y_0||y_1 = f(x)$ (where $|y_0| = s(n)$ and $|y_1| = n$), and outputs $y_0||f(y_1)$. G^f can be formalized as a fully black-box construction making two *adaptive* oracle queries, each of the same length as G's seed x, to an oracle mapping n bits to $n + s(n)$ bits. This idea can easily be extended to obtain, for every $k \in \mathbb{N}$, a fully black-box construction making k adaptive oracle queries and achieving stretch $k \cdot s(n)$.

  We show that fully black-box constructions making constantly-many queries, each of the same length as their seed length n, *must* make *adaptive* queries even to achieve stretch $s(n) + 1$, that is, even to achieve a one-bit increase in stretch. We show that this also holds for constructions whose seed length is at most $O(\log n)$ bits longer than the length n of each oracle query.

– *Class 2: Constructions with long seeds*
  What about constructions whose seed length is significantly longer than the length of each oracle query? Can we also show that such constructions must make adaptive oracle queries in order to achieve greater stretch than their oracle? In fact, a very simple way for such a construction to make non-adaptive oracle queries, yet achieve greater stretch than its oracle, involves splitting up its seed into two or more portions, and using each portion as an oracle query. For example, if $f : \{0,1\}^n \to \{0,1\}^{n+1}$ is pseudo-random, then the generator $G^f : \{0,1\}^{2n} \to \{0,1\}^{2n+2}$ defined for all $x_1, x_2 \in \{0,1\}^n$ as $G^f(x_1||x_2) = f(x_1)||f(x_2)$ is also pseudo-random. Observe that when this construction is given an input chosen uniformly at random, the oracle queries x_1 and x_2 are chosen independently (and uniformly at random); this property is crucial for the construction's security.

  What about constructions where oracle queries cannot be chosen independently and uniformly at random? Specifically, what if we consider constructions where we place no restriction on the seed length, but insist that oracle queries are collectively chosen in a manner that depends only on a portion of the seed that is not too much longer than the length of each oracle query (making it impossible to simply split up the seed into multiple queries)? While this setting may seem unnatural at first, it *is* possible in this setting to obtain a construction that makes constantly-many non-adaptive oracle queries to a pseudo-random generator and achieves more stretch than its oracle; indeed, even a single query suffices. For example, if $f : \{0,1\}^n \to \{0,1\}^{n+s(n)}$ is pseudo-random, then by the Goldreich-Levin theorem [6] we have that for all functions $m(n) \in O(\log n)$, the number generator $G^f : \{0,1\}^{n \cdot m(n)+n} \to \{0,1\}^{n \cdot m(n)+n+s(n)+m(n)}$ defined for all $r_1, r_2, \ldots, r_{m(n)}, x \in \{0,1\}^n$ as

$$G^f\left(r_1||r_2||\ldots||r_{m(n)}||x\right) = r_1||r_2||\ldots||r_{m(n)}||f(x)||\langle r_1, x\rangle||\langle r_2, x\rangle||\ldots||\langle r_{m(n)}, x\rangle$$

is pseudo-random; the stretch of G^f is $m(n)$ bits greater than the stretch of f. Also observe that the query made by $G^{(\cdot)}$ depends only on a portion of the seed of $G^{(\cdot)}$ whose length is the same as the length of the query (indeed, the query is identical to this portion of the seed). Using this Goldreich-Levin-based approach, it is easy to see that *adaptive* black-box constructions whose input length is much longer than the length n of each oracle query can obtain stretch $k \cdot s(n) + O(\log n)$ by making k queries to an oracle of stretch $s(n)$, even when the portion of the seed that is used to choose oracle queries has length n.

We show that fully black-box constructions $G^{(\cdot)}$ making constantly-many queries of length n to a pseudo-random generator $f : \{0,1\}^n \to \{0,1\}^{n+s(n)}$, such that only the rightmost $n + O(\log n)$ bits of the seed of $G^{(\cdot)}$ are used to choose oracle queries, *must* make *adaptive* queries in order to achieve stretch $s(n) + \omega(\log n)$. That is, such constructions making *constantly*-many non-adaptive queries cannot achieve greater stretch than the stretch provided by Goldreich-Levin with just a *single* query. This holds no matter how long a seed is used by the construction $G^{(\cdot)}$.

– *Class 3: Goldreich-Levin-like constructions*
The final class of constructions we consider is motivated by the streaming computation of pseudo-random generators. What is the relationship between non-adaptivity and streaming? In what sense could one prove a black-box lower bound that rules out streaming constructions of pseudo-random generator G of linear stretch using a one-way permutation π? A black-box lower-bound stated for a streaming device has to reference the many details of the model. We wish to state a similar thing in a setting that abstracts out a common property of streaming algorithms extended to have oracle access to a one-way permutation. Can a streaming algorithm be adaptive (even when we do not account for the space occupied by the oracle tape), in the sense that a query depends on many bits of previous queries? Given that a random input is incompressible, and the fact that we lack space (so as to store) and passes over the input (so as to recompute), it is plausible to consider non-adaptivity as a clean setting for studying black-box streaming constructions.

We consider a class of constructions where the seed has a public portion that is always included in the output, the choice of each oracle query does not depend on the public portion of the seed, and the computation of each individual output bit depends only on the seed and on the response to a *single* oracle query. We refer to such constructions making non-adaptive oracle queries as *bitwise-nonadaptive* constructions. It is not hard to see that such constructions making polynomially-many *adaptive* queries to a one-way permutation $\pi : \{0,1\}^n \rightarrow \{0,1\}^n$ can achieve arbitrary polynomial stretch; the idea is to repeatedly compose π with itself, outputting a hardcore bit of π on each composition. For example, using the Goldreich-Levin hardcore bit [6], a standard way of constructing a pseudo-random generator G of polynomial stretch $p(n)$ is the following: On input $r, x \in \{0,1\}^n$,

$$G^{\pi}(r||x) = r||\langle r, x\rangle||\langle r, \pi(x)\rangle||\langle r, \pi^2(x)\rangle||\ldots||\langle r, \pi^{p(n)+n}(x)\rangle$$

where $\pi^i := \underbrace{\pi \circ \pi \circ \ldots \circ \pi}_{i \text{ times}}$, and $\langle \alpha, \beta\rangle$ denotes the standard inner product of α and β. Observe that the leftmost n bits of the seed of G are public in the sense that they are included in the output. Also observe that each of the remaining output bits of G is computed using only a single output of π along with the input bits of G. Finally, observe that the queries made to π do not depend on the public input bits of G, and the number of non-public input bits is no greater than the length n of each oracle query. It is natural to ask whether the *adaptive* use of π in a construction of this form is necessary. This is particularly interesting if we wish to compute G in a streaming setting where we have small workspace, we are allowed to produce the output bit-by-bit, and we are allowed to re-read the input once per output bit.
We show that fully black-box bitwise-nonadaptive constructions $G^{(\cdot)}$ making queries of length n to a one-way permutation, such that the non-public portion of the seed of $G^{(\cdot)}$ is of length at most $n + O(\log n)$, cannot achieve linear stretch. This holds no matter the length of the public portion of the seed of $G^{(\cdot)}$.

We conclude this paper with some remarks and observations about streaming models for cryptography. Our treatment of streaming models mostly serves the purpose of proposing some new research directions.

Related work. Black-box reductions were formalized by Impagliazzo and Rudich [8], who observed that most proofs of security in cryptography are of this form. Impagliazzo and Rudich also gave the first black-box impossibility results. In their most general form, such results show that for particular security properties P_1 and P_2, it is impossible to give a black-box construction of P_1 from P_2. The same approach can also be applied to particular classes of black-box constructions, such as those making some restricted number of oracle queries or those that query their oracle non-adaptively. A large number of impossibility results have been given using this framework. The results most closely related to the problem we are considering are those of Gennaro *et al* [5], Viola [13], Lu [10], and Miles and Viola [11].

Gennnaro *et al* [5] consider black-box constructions of pseudo-random generators from one-way permutations. They show that such constructions cannot achieve $\omega(\log n)$ bits of stretch per oracle query of length n, even when queries are chosen adaptively. Their result can be extended in a straightforward way to show that for the second class of constructions we consider (and also for a more general class where queries are allowed to depend on the entire seed), for every $k \in \mathbb{N}$, constructions making k oracle queries to a pseudo-random generator of stretch $s(n)$ cannot achieve stretch $k \cdot s(n) + \omega(\log n)$, even when these queries are chosen adaptively. By contrast, recall that we show that for this class of constructions, for every $k \in \mathbb{N}$, constructions making k *non-adaptive* oracle queries to a pseudo-random generator of stretch $s(n)$ cannot achieve stretch $s(n) + \omega(\log n)$.

Viola [13] considers black-box constructions of pseudo-random generators from one-way *functions* where oracle queries are non-adaptive but chosen in a computationally unbounded way, while the output of the construction is computed from the query responses by an AC^0 (polynomial-size and constant-depth) circuit. He shows that such constructions cannot achieve linear stretch. The class of constructions considered by Viola is, in general, incomparable to the classes we consider. His class is more general in terms of the numbers of queries allowed and the way that queries are chosen: he places no bounds on the number of queries, allows the queries to be chosen arbitrarily based on the seed (while we require queries to be chosen in a computable manner), and places no restrictions on the length of the queries relative to the length of the seed. On the other hand, his class is more restrictive in terms of the computational power allowed after the query responses are received: he only allows AC^0 computation, while we allow unbounded computation.

Lu [10] considers the same class of constructions as Viola, except that Lu allows the output to be computed from the query responses by a subexponential-size constant-depth circuit (rather than an AC^0 circuit). He shows that such constructions cannot achieve linear stretch.

Miles and Viola [11] consider black-box constructions of pseudo-random generators from pseudo-random generators of 1-bit stretch, where the oracle queries are non-adaptive but chosen in a computationally unbounded way, while the output of the construction consists simply of query response bits; that is, these constructions are not allowed to perform any computation on query responses. They show that such constructions cannot achieve linear stretch. Like the constructions considered by Viola [13] and Lu [10], the class of constructions considered by Miles and Viola is, in general, incomparable to the classes we consider: the constructions they consider are more general in the manner in which queries are chosen (they place no restrictions on the length of queries relative to the length of the seed), but much more restrictive in terms of the computational power allowed after query responses are received.

In the positive direction, Haitner *et al* [7] give the first *non-adaptive* black-box construction of a pseudo-random generator from a one-way *function*. Their construction achieves sublinear stretch. They also give a non-adaptive black-box construction achieving linear stretch, but this requires an *exponentially-hard* one-way function. In both of these constructions, the oracle queries are collectively chosen based on a portion of the seed that is significantly longer than the length of each oracle query. By contrast, recall that all of our impossibility results are for constructions where the oracle queries are collectively chosen based on a portion of the seed that is no more than logarithmically-many bits longer than the length of each oracle query.

Organization. Section 2 contains definitions and preliminaries. The impossibility results for constructions with short seeds and long seeds are discussed in Sections 3 and 4 respectively. In Section 5, we state a restriction on the way that constructions choose oracle queries, and under this restriction we extend the results of Sections 3 and 4 to constructions making polynomially-many queries. The impossibility result for Goldreich-Levin-like constructions is found in Section 6. Section 7 contains our remarks on streaming models in cryptography.

2 Preliminaries

Notation. We use "PPT" to denote "probabilistic polynomial time". We denote by $\langle a \rangle_n$ the n-bit binary string representation of $a \in \mathbb{N}$, padded with leading zeros when necessary. If the desired representation length is clear from the context, we write $\langle a \rangle$ instead of $\langle a \rangle_n$. If $a \geq 2^n$, then $\langle a \rangle_n$ denotes the n least significant bits of the binary representation of a. We denote by $x||y$ the concatenation of strings x and y.

2.1 Pseudo-random generators and one-way functions

A length-increasing function $G : \{0,1\}^{\ell_1(n)} \to \{0,1\}^{\ell_2(n)}$ is a *pseudo-random generator* if for every PPT adversary M, we have

$$\left| \Pr_{x \leftarrow \{0,1\}^{\ell_1(n)}} \left[M\left(G\left(x\right)\right) = 1 \right] - \Pr_{z \leftarrow \{0,1\}^{\ell_2(n)}} \left[M\left(z\right) = 1 \right] \right| \leq 1/n^c$$

for all c and sufficiently large n.

A function $f : \{0,1\}^{\ell_1(n)} \to \{0,1\}^{\ell_2(n)}$ is *one-way* if for every PPT adversary M, we have $\Pr_{x \leftarrow \ell_1(n)} \left[f\left(M\left(f\left(x\right)\right)\right) = f\left(x\right) \right] \leq 1/n^c$ for all c and sufficiently large n.

2.2 Non-adaptive constructions

Our impossibility results are for constructions that use their oracle in a non-adaptive manner.

Definition 1. (Non-adaptive oracle machine) *Let $M^{(\cdot)}$ be a deterministic oracle Turing machine. We say that $M^{(\cdot)}$ is a* non-adaptive oracle machine *if the oracle queries made by $M^{(\cdot)}$ are determined by only the input to $M^{(\cdot)}$, and, in particular, do not depend on the responses to previous queries.*

We will sometimes need to refer to the *querying function* of a non-adaptive oracle machine.

Definition 2. (Querying function) *Let $\ell_1(n)$, $\ell_2(n)$, and $p(n)$ be polynomials, and let $M^{(\cdot)} : \{0,1\}^{\ell_1(n)} \to \{0,1\}^{\ell_2(n)}$ be a non-adaptive oracle machine that makes $p(n)$ oracle queries, each of length n. The* querying function *of $M^{(\cdot)}$, denoted Q_M, is the function $Q_M : \{0,1\}^{\ell_1(n)} \times \{0,1\}^{\log p(n)} \to \{0,1\}^n$ such that for all $x \in \{0,1\}^{\ell_1(n)}$ and $0 \le i < p(n)$, the i-th oracle query made by $M^{(\cdot)}(x)$ is $Q_M(x, \langle i \rangle)$. When $p(n) \equiv 1$, the second argument to Q_M is omitted.*

If there exists a polynomial $r(n)$ such that the queries made by $M^{(\cdot)}$ depend only on the rightmost $r(n)$ bits of the input of $M^{(\cdot)}$, then the $r(n)$-restricted querying function *of $M^{(\cdot)}$, denoted $Q_M^{r(n)}$, is the function $Q_M^{r(n)} : \{0,1\}^{r(n)} \times \{0,1\}^{\log p(n)} \to \{0,1\}^n$ such that for all $v \in \{0,1\}^{\ell_1(n)-r(n)}$, $w \in \{0,1\}^{r(n)}$, and $0 \le i < p(n)$, the i-th oracle query made by $M^{(\cdot)}(v||w)$ is $Q_M^{r(n)}(w, \langle i \rangle)$.*

2.3 Black-box reductions

Reingold, Trevisan, and Vadhan [12] give a classification of black-box security reductions. Our impossibility results apply to what Reingold *et al* call *fully-black box* reductions. We avoid defining such reductions in their full generality and instead focus on security reductions for constructions of pseudo-random number generators from pseudo-random generators of smaller stretch.

Definition 3. (Fully black-box reduction [8]) *Let $G^{(\cdot)} : \{0,1\}^{\ell_1(n)} \to \{0,1\}^{\ell_2(n)}$ be a number generator whose construction has access to an oracle for a length-increasing function mapping $\ell'_1(n)$ bits to $\ell'_2(n)$ bits. There is a* fully black-box *reduction of the pseudo-randomness of $G^{(\cdot)}$ to the pseudo-randomness of its oracle if there exists a PPT oracle machine $M^{(\cdot,\cdot)}$ such that for every function $f : \{0,1\}^{\ell'_1(n)} \to \{0,1\}^{\ell'_2(n)}$ and every function $A : \{0,1\}^{\ell_2(n)} \to \{0,1\}$, if A breaks the pseudo-randomness of G^f then $M^{(f,A)}$ breaks the pseudo-randomness of f.*

Definition 3 can be modified in a straightforward way for constructions of pseudo-random number generators from other primitives, such as from one-way permutations.

An oracle construction whose security is proven using a black-box reduction is called a *black-box construction.*

3 Constructions with short seeds

In this section, we consider constructions whose seed length is not more than $O(\log n)$ bits longer than the length n of each oracle query. Recall that such constructions making k adaptive queries to a given pseudo-random generator can achieve stretch that is k times the stretch of the given generator. We show that such constructions making constantly-many *non-adaptive* queries cannot achieve stretch that is *even a single bit* longer than the stretch of the given generator.

Theorem 1. *Let $k \in \mathbb{N}$, and let $\ell_1(n)$ and $\ell_2(n)$ be polynomials such that $\ell_1(n) \le n+O(\log n)$ and $\ell_2(n) > n$. Let $G^{(\cdot)} : \{0,1\}^{\ell_1(n)} \to \{0,1\}^{\ell_1(n)+(\ell_2(n)-n)+1}$ be a non-adaptive oracle construction of a number generator, making k queries of length n to an oracle mapping n bits to $\ell_2(n)$ bits. Then there is no fully black-box reduction of the pseudo-randomness of $G^{(\cdot)}$ to the pseudo-randomness of its oracle.*

The approach we use to prove Theorem 1 does not seem to extend to the case of polynomially-many (or even $\omega(1)$-many) queries. However, a similar approach does work for polynomially-many queries when we place a restriction on the many-oneness of the number generator's querying function. We state this restriction in Section 5.

We give an overview of the proof of Theorem 1 in Section 3.1, and we give the proof details in the full version of this paper.

3.1 Proof overview for Theorem 1

A simpler case We first consider the simpler case of constructions making just a single query, where the query made is required to be the same as the construction's input. That is, we consider constructions $G^{(\cdot)} : \{0,1\}^n \to \{0,1\}^{\ell_2(n)+1}$ such that on every input $x \in \{0,1\}^n$, G makes query x to an oracle mapping n bits to $\ell_2(n)$ bits. Fix such a construction $G^{(\cdot)}$. We need to show the existence of functions $f : \{0,1\}^n \to \{0,1\}^{\ell_2(n)}$ and $A : \{0,1\}^{\ell_2(n)} \to \{0,1\}$ such that A breaks the pseudo-randomness of G^f but f is pseudo-random even with respect to adversaries that have oracle access to f and A. Following the approach for proving black-box impossibility results initiated by Impagliazzo and Rudich [8], we actually define a *joint distribution* $(\mathcal{F}, \mathcal{A})$ over pairs of functions, such that with probability one over $(f, A) \leftarrow (\mathcal{F}, \mathcal{A})$, A breaks the pseudo-randomness of G^f but f is pseudo-random even with respect to adversaries that have oracle access to f and A.

Consider how we might define such a joint distribution $(\mathcal{F}, \mathcal{A})$. The most obvious approach is to let $(\mathcal{F}, \mathcal{A})$ be the distribution defined by the following procedure for sampling a tuple $(f, A) \leftarrow (\mathcal{F}, \mathcal{A})$: randomly select f from the (infinite) set of all functions that, for each $n \in \mathbb{N}$, map n bits to $\ell_2(n)$ bits; let A be the function such that for every $z \in \{0,1\}^{\ell_2(n)+1}$, $A(z) = 1$ if and only if there exists an $s \in \{0,1\}^n$ such that $G^f(s) = z$. Following this approach, we have that with probability one over $(f, A) \leftarrow (\mathcal{F}, \mathcal{A})$, A breaks the pseudo-randomness of G^f but f is pseudo-random with respect to adversaries that have oracle access to f alone. However, it is *not* necessarily the case that f is pseudo-random with respect to adversaries that have oracle access to f *and* A. For example, suppose construction G is such that for every $x \in \{0,1\}^{n-1}$ and every $b \in \{0,1\}$, $G^f(x||b) = f(x||b)||b$. In this case, it is easy to use A to break f: on input $y \in \{0,1\}^{\ell_2(n)}$, output 1 if and only if either $A(y||0) = 1$ or $A(y||1) = 1$.

To overcome this problem, we add some "noise" to A. We need to be careful that we add enough noise to A so that it is no longer useful for breaking f, but we do not add so much noise that A no longer breaks G^f. Our basic aproach is to modify A so that instead of only accepting $G^f(s)$ for all $s \in \{0,1\}^n$, A accepts $G^{f_i(s)}$ for all s, all i, and some appropriate collection of functions $\{f_0, f_1, f_2, \dots\}$ where $f_0 = f$. How should this collection of functions be defined? Since we want to make sure that A still breaks G^f, and since we have that A accepts $G^f(s)$ with probability 1 over $s \leftarrow \{0,1\}^n$, we need to ensure that A accepts randomly chosen strings with probability non-negligibly less than 1. For this, it suffices to ensure that (# of n-bit strings s)*(# of functions f_i) is at most, say, half the number of strings of length $\ell_2(n)+1$. At the same time, to prevent A from helping to break f, we would like it to be the case that, intuitively, A treats strings that are *not* in the image of f on an equal footing with strings that *are* in the image of f. One way to accomplish these objectives, which we follow, is to randomly select a permutation π on $\{0,1\}^{\ell_2(n)}$, define $f(x) = \pi(0^{\ell_2(n)-n}||x)$ for all $x \in \{0,1\}^n$, and define A to accept $G^{\pi(y||\cdot)}(s)$ for every $y \in \{0,1\}^{\ell_2(n)-n}$ and every $s \in \{0,1\}^n$. We formalize this as a joint distribution $(\mathcal{F}, \mathcal{A}, \Pi)$ over tuples (f, A, π) that are sampled in the manner just described.

It is easy to show that with probability one over $(f, A, \pi) \leftarrow (\mathcal{F}, \mathcal{A}, \Pi)$, A does indeed break G^f. It is much more difficult to show that with probability one over $(f, A, \pi) \leftarrow (\mathcal{F}, \mathcal{A}, \Pi)$, f is pseudo-random ever with respect to PPT adversaries that have oracle access to f and A. We argue that it suffices to show that for every PPT oracle machine $D^{(\cdot,\cdot)}$, the probability over $(f, A, \pi) \leftarrow (\mathcal{F}, \mathcal{A}, \Pi)$ and $s \leftarrow \{0,1\}^n$ that $D^{(f,A)}(f(s))$ makes oracle query s to f is negligible. Now, instead of only showing this for every PPT oracle machine $D^{(\cdot,\cdot)}$, we find it more convenient to show this for every computationally unbounded probabilistic oracle machine $D^{(\cdot,\cdot)}$ that makes at most polynomially-many oracle queries. How might we do so? We would like to argue that A does not help D to find s since a computationally unbounded D can try to compute A by itself. More formally, we would like to show that given D, we can build a D' that, given input $f(s)$ and given oracle access only to f, simulates D on input $f(s)$, answers f-queries of D using the given oracle, and "makes up" answers to the A-queries of D in a manner that ensures that the probability that the simulation of D makes query s is very close to the probability that $D^{(f,A)}(f(s))$ makes oracle query s. Of course, D' does not "know" π, so it is not immediately clear how it should answer the A-queries of the simulation of D. If D' simply randomly chooses its own permutation π' and answers A-queries using π' in place of the unknown π, the simulation of D may "notice" this sleight of hand. For example, since D is given $f(s)$ as input, it might (depending on the definition of G) be able to compute the value of $G^f(s)$, and hence make query $G^f(s)$ to A; if this query does not produce response 1, D will "know" that queries are not being responded to properly.

We address this by showing that D' can still compute "most" of A on its own, and that the "rest" of A is not helpful for finding s. Specifically, we split A into two functions, A_1 and A_2, that together can be used to compute A. Function A_1 outputs 1 only on input $G^f(s)$. For every $(\ell_2(n)+1)$-bit string z, $A_2(z) = 1$ if and only if $z \neq G^f(s)$ and $A(z) = 1$. We then argue that querying A_1 provides very little help for finding s. Let X be the set of all strings $x \in \{0,1\}^n$ such that

$G^f(x) = G^f(s)$. Roughly speaking, if X is large, then A_1 gives no information about s beyond the fact that $s \in X$. On the other hand, if X is small, then we argue it is unlikely that an adversary making polynomially-many queries to A_1 will receive a non-zero response to any of its queries (in other words, it is unlikely that query $G^f(s)$ will be made). It remains to argue that D' can compute A_2 on its own. We show that if D' randomly selects a permutation π', computes an A'_2 based on π' (rather than π), uses this A'_2 along with the given A_1 to answer the A-queries of the simulation of D, and answers the f-queries of the simulation of D based on $\pi'(0^{\ell_2(n)-n}||\cdot)$ (rather than using the given oracle f), then it is unlikely that the simulation of D will make a query that "exposes" the fact that its oracle queries are not being answered by f and A.

The general case We extend the above argument to constructions $G^{(\cdot)} : \{0,1\}^{\ell_1(n)} \to \{0,1\}^{\ell_1(n)+(\ell_2(n)-n)+1}$ making constantly-many non-adaptive queries, where the length $\ell_1(n)$ of the construction's input is allowed to be $O(\log n)$ bits longer than the length n of each oracle query. The high-level idea is the same: we define a joint distribution $(\mathcal{F}, \mathcal{A}, \Pi)$ by specifying a procedure for sampling a tuple $(f, A, \pi) \leftarrow (\mathcal{F}, \mathcal{A}, \Pi)$, and the way we sample π and f is (almost) the same as before. But now we change the way A behaves. Our goal is to follow the same style of argument as before. To accomplish this, we would still like it to be the case that when we "split up" A into functions A_1 and A_2, there is still at most one string accepted by A_1 (this helps us ensure that A_1 does not provide too much information about s). Recall that before, when D' was run on an input $f(s)$, the unique string accepted by A_1 was $G^f(s)$. This made sense because in the previous setting, the only input on which $G^{(\cdot)}$ made oracle query s was s itself. But in the current setting, for each $s \in \{0,1\}^n$, there may be many inputs $x \in \{0,1\}^{\ell_1(n)}$ on which $G^{(\cdot)}$ makes oracle query s. We would like to modify the definition of A so that rather than accepting $G^{\pi(y||\cdot)}(x)$ for every $y \in \{0,1\}^{\ell_2(n)-n}$ and *every* $x \in \{0,1\}^{\ell_1(n)}$, A accepts $G^{\pi(y||\cdot)}(x)$ for every $y \in \{0,1\}^{\ell_2(n)-n}$ and x in some subset $\mathit{Good}(n) \subseteq \{0,1\}^{\ell_1(n)}$ such that for every $s \in \{0,1\}^n$, there is at most one $x \in \mathit{Good}(n)$ such that $G^{(\cdot)}$ on input x makes query s. But we cannot do exactly this (and still have that A breaks G^f), since, for example, there might be some string t that $G^{(\cdot)}$ queries no matter what its input is.

Instead, we need to proceed very carefully, partitioning the set of strings t of length n into those that are queried by $G^{(\cdot)}$ for "many" of its inputs $x \in \{0,1\}^{\ell_1(n)}$, and those queried by $G^{(\cdot)}$ for "at most a few" of its inputs $x \in \{0,1\}^{\ell_1(n)}$. We call the former set $Fixed(n)$ and the latter set $NotFixed(n)$. We then define a set $\mathit{Good}(n) \subseteq \{0,1\}^{\ell_1(n)}$ of inputs to $G^{(\cdot)}$ such that for no pair of distinct inputs from $\mathit{Good}(n)$ does $G^{(\cdot)}$ make the same query $t \in NotFixed(n)$. That is, each $t \in NotFixed(n)$ is queried by $G^{(\cdot)}$ for at most one of its inputs $x \in \mathit{Good}(n)$. The challenge, of course, is ensuring that that the set $\mathit{Good}(n)$ defined this way is "large enough".

We define A to accept $G^{\pi(y||\cdot)}(x)$ for every $y \in \{0,1\}^{\ell_2(n)-n}$ and every $x \in \mathit{Good}(n)$. Now we can "split up" A into A_1 and A_2 in a manner similar to what we did before: on input $f(s)$ to D', where $s \in NotFixed(n)$, if there exists a string $x \in Good(n)$ such that $G^{(\cdot)}(x)$ makes query s (note that there can be at most one such string x by definition of $Good(n)$), then A_1 only accepts $G^f(x)$, and if there is no such string x then A_1 does not accept any strings; as before, we define A_2 to accept the remaining strings accepted by A. We then argue as before about the (lack of) usefulness of A_1 and A_2 for helping to find s. Finally, we argue that our definition of $Fixed(n)$ ensures that this set will be of negligible size, and hence it does not hurt to ignore the case $s \in Fixed(n)$ (since this case will occur with negligible probability).

4 Constructions with long seeds

In Section 3, we saw that black-box constructions $G^{(\cdot)}$ making constantly-many non-adaptive oracle queries, where the seed length of $G^{(\cdot)}$ is not too much longer than the length of each oracle query, cannot achieve even a single bit more stretch than their oracle. In this section, we consider constructions whose seed length is allowed to be much longer than the length of each oracle query, but where the oracle queries are collectively chosen in a manner that depends only on a portion of the seed whose length is not more than $O(\log n)$ bits longer than the length n of each oracle query. Recall that such constructions making even a single query to a given pseudo-random generator can achieve stretch that is $O(\log n)$ bits longer than the stretch of the given generator [6]. Further, recall that such constructions making k adaptive queries can achieve stretch that is $O(\log n)$ bits longer than k times the stretch of the given generator. We show that such constructions making constantly-many non-adaptive queries cannot achieve stretch that is $\omega(\log n)$ bits longer than the stretch of the given generator.

Theorem 2. *Let $k \in \mathbb{N}$, $c \in \mathbb{R}^+$, and $m(n) \in \omega(\log n)$. Let $\ell_0(n)$, $\ell_1(n)$, and $\ell_2(n)$ be polynomials such that $\ell_1(n) \leq n + c\log n$. Let $G^{(\cdot)} : \{0,1\}^{\ell_0(n)+\ell_1(n)} \to \{0,1\}^{\ell_0(n)+\ell_1(n)+(\ell_2(n)-n)+m(n)}$ be a non-adaptive oracle construction of a number generator that makes k queries of length n to a number generator mapping n bits to $\ell_2(n)$ bits, such that for all $r \in \{0,1\}^{\ell_0(n)}$ and $x \in \{0,1\}^{\ell_1(n)}$, the queries made by $G^{(\cdot)}$ on input $(r||x)$ depend only on x. Then there is no fully black-box reduction of the pseudo-randomness of $G^{(\cdot)}$ to the pseudo-randomness of its oracle.*

As is the case for Theorem 1, the approach we use to prove Theorem 2 does not seem to extend to the case of polynomially-many (or even $\omega(1)$-many) queries. However, a similar approach does work for polynomially-many queries when we place a restriction on the many-oneness of the number generator's querying function. We state this restriction in Section 5.

We give an overview of the proof of Theorem 2 in Section 4.1, and we give the proof details in the full version of this paper.

4.1 Proof overview for Theorem 2

As in the proof of Theorem 1, it suffices to define a joint distribution $(\mathcal{F}, \mathcal{A})$ over pairs of functions, such that with probability one over $(f, A) \leftarrow (\mathcal{F}, \mathcal{A})$, A breaks the pseudo-randomness of G^f but f is pseudo-random even with respect to adversaries that have oracle access to f and A. Unlike the previous proof, we actually define distributions $\mathcal{F}$ and $\mathcal{A}$ that are independent – in fact, we define $\mathcal{A}$ to be a degenerate distribution that assigns all probability to a fixed function A. We define a set $Good(n) \subseteq \{0,1\}^{\ell_1(n)}$ in a careful manner very similar to the proof of Theorem 1, but taking into account the fact that the queries of $G^{(\cdot)}$ depend only on the rightmost $\ell_1(n)$ bits of its seed. The goal is to ensure that $Good(n)$ is sufficiently large and has the property that for every string $x \in Good(n)$, every $r \in \{0,1\}^{\ell_0(n)}$, and every $f \in \mathcal{F}$, A accepts $G^f(r||x)$. Simultaneously, we need to ensure that the total number of strings accepted by A is sufficiently smaller than $2^{\ell_0(n)+\ell_1(n)+(\ell_2(n)-n)+m(n)}$ and that $f \leftarrow \mathcal{F}$ is pseudo-random with probability one even with respect to adversaries that have oracle access to f and A.

If we define $\mathcal{F}$ in a very straightforward way (e.g. as the uniform distribution over all 1-1 functions), the total number of strings that A will need to accept (in order to accept $G^f(r||x)$ for every $f \in \mathcal{F}$, every r, and every $x \in Good(n)$) could be too large. The problem is that when deciding whether to accept a given input, A is existentially quantifying over over a set that is (much) larger than the set of its possible inputs. We need to minimize the number of different $f \in \mathcal{F}$ (while, of course, still ensuring that $f \leftarrow \mathcal{F}$ is pseudo-random with probability one even with respect to adversaries that have oracle access to f and A). At the same time, we need to add some structure to the $f \in \mathcal{F}$ to, intuitively, reduce the amount of new information contained in the responses to the oracle queries made by G^f when run on each $r||x$ where $x \in Good(n)$. The idea is that rather than existentially quantifying over every r, every $x \in Good(n)$, and every $f \in \mathcal{F}$ when deciding whether to accept a particular input z, A will instead existentially quantify over every r, every $x \in Good(n)$, and every possible value for the (small amount of) new information (that is, the information not already determined by x) contained in the responses to oracle queries made by $G^{(\cdot)}$ when run on input $r||x$.

Similarly to the proof of Theorem 1, our procedure for constructing the set $Good(n)$ ensures that for every distinct $x, x' \in Good(n)$, each query q made by G, when run on an input whose rightmost bits are x, is either in some small set $Fixed(n)$ or is distinct from every query q' made by G when run on every input whose rightmost bits are x'. This allows us to follow a two-step approach to defining $\mathcal{F}$. We first define a permutation h on $\{0,1\}^n$ that, for each $x \in Good(n)$, maps the queries $q \notin Fixed(n)$ made by G, when run on an input whose rightmost bits are x, to strings that differ in at most a small number of bits, and, in particular, have a common $(m(n)/2)$-bit suffix. Roughly speaking, sampling $f \leftarrow \mathcal{F}$ proceeds as follows. We randomly select a function $f' : \{0,1\}^n \leftarrow \{0,1\}^{\ell_2(n)}$ that is the identity on its first $n - m(n)/2$ input bits, and is 1-1 on its last $m(n)/2$ input bits, mapping them to $\ell_2(n) - n + m(n)/2$ output bits. We then define $f = f' \circ h$. The actual definition of $\mathcal{F}$ that we use in the proof also ensures that for every $q \in \mathit{Fixed}(n)$, the value $f(q)$ is independent of the choice $f \leftarrow \mathcal{F}$ (that is, $f_1(q) = f_2(q)$ for all $f_1, f_2 \in \mathcal{F}$).

Intuitively, this approach ensures that $f \leftarrow \mathcal{F}$ has "just enough" randomness. At the same time, this approach ensures that for every r and every $x \in \mathit{Good}(n)$, the responses to oracle queries made by $G^f(r||x)$ collectively contain at most $\ell_2(n) - n + m(n)/2$ bits of information that depend on the choice $f \leftarrow \mathcal{F}$.

We remark that it is crucial for this proof that $2^{m(n)/2}$ is super-polynomial. It is for this reason that we cannot adapt the current proof in order to obtain a significantly simpler proof of Theorem 1; in Theorem 1, the corresponding value of $m(n)$ (the additional stretch achieved by $G^{(\cdot)}$) is exactly 1.

5 Moving beyond constantly-many queries

In this section we consider extending Theorem 1 and Theorem 2 to the case of polynomially-many queries. We are able to do this for a restricted class of constructions. We begin by defining the restriction we need to place on the querying function of the construction.

Definition 4. (Many-oneness bounded almost everywhere) *Let $\ell(n)$ and $q(n)$ be polynomials, and let $f : \{0,1\}^{\ell(n)} \to \{0,1\}^n$ be a function. f has* many-oneness bounded by $q(n)$ almost everywhere *if for all c and sufficiently large n, there are fewer than $2^n/n^c$ strings $y \in \{0,1\}^n$ such that $|f^{-1}(y)| > q(n)$.*

Theorem 3. *Let $p(n)$, $q(n)$, $\ell_1(n)$, and $\ell_2(n)$ be polynomials such that $\ell_1(n) \leq n + O(\log n)$ and $\ell_2(n) > n$. Let $G^{(\cdot)} : \{0,1\}^{\ell_1(n)} \to \{0,1\}^{\ell_1(n)+(\ell_2(n)-n)+1}$ be a non-adaptive oracle construction of a number generator, making $p(n)$ queries of length n to an oracle mapping n bits to $\ell_2(n)$ bits, such that the querying function of $G^{(\cdot)}$ has many-oneness bounded by $q(n)$ almost everywhere. Then there is no fully black-box reduction of the pseudo-randomness of $G^{(\cdot)}$ to the pseudo-randomness of its oracle.*

Theorem 4. *Let $c \in \mathbb{R}^+$ and $m(n) \in \omega(\log n)$. Let $p(n)$, $q(n)$, $\ell_0(n)$, $\ell_1(n)$, and $\ell_2(n)$ be polynomials such that $\ell_1(n) \leq n + c \log n$. Let $G^{(\cdot)} : \{0,1\}^{\ell_0(n)+\ell_1(n)} \to \{0,1\}^{\ell_0(n)+\ell_1(n)+(\ell_2(n)-n)+m(n)}$ be a non-adaptive oracle construction of a number generator that makes $p(n)$ queries of length n to a number generator mapping n bits to $\ell_2(n)$ bits, such that $G^{(\cdot)}$ has an $\ell_1(n)$-restricted querying function whose many-oneness is bounded by $q(n)$ almost everywhere. Then there is no fully black-box reduction of the pseudo-randomness of $G^{(\cdot)}$ to the pseudo-randomness of its oracle.*

The proofs of Theorem 3 and Theorem 4 follow the same basic structure as the proofs of Theorem 1 and Theorem 2, respectively, but the procedure used to define the set $\mathit{Good}(n)$ in each proof is simpler as a result of the restriction on the many-oneness of the querying function. For both Theorem 3 and Theorem 4, the procedure begins by defining $\mathit{Fixed}(n) \subseteq \{0,1\}^n$ to be the set of strings in the image of the querying function Q_G whose many-oneness is *not* bounded by $q(n)$. Then, since the remaining strings in the image of Q_G have bounded many-oneness, it is easy to define a large set $\mathit{Good}(n) \subseteq \{0,1\}^{\ell_1(n)}$ such that for all distinct $x, x' \in \mathit{Good}(n)$ and all $0 \leq i, j < p(n)$, either $Q_G(x, \langle i \rangle) \in \mathit{Fixed}(n)$ or $Q_G(x, \langle i \rangle) \neq Q_G(x', \langle j \rangle)$. The idea is to proceed as follows: initially, every $x \in \{0,1\}^{\ell_1(n)}$ is a candidate for inclusion in $\mathit{Good}(n)$; while there are candidates remaining, select an arbitrary candidate x, add it to $\mathit{Good}(n)$, and remove from consideration as candidates all x' such that for some $0 \leq i, j < p(n)$, we have $Q_G(x, \langle i \rangle) \notin \mathit{Fixed}(n)$ and $Q_G(x, \langle i \rangle) = Q_G(x', \langle j \rangle)$. For every x added to $\mathit{Good}(n)$ by this procedure, at most $p(n)(q(n)-1)$ are removed from consideration, and hence at the end of this procedure $\mathit{Good}(n)$ has size at least $2^{\ell_1(n)}/(p(n)(q(n)-1)+1)$. Further details about these proofs are omitted for the sake of conciseness.

6 Goldreich-Levin-like constructions

In this section, we consider constructions where the seed has a public portion that is always included in the output, such that the oracle queries are chosen *non-adaptively* based only on the non-public portion of the seed. We further require that the computation of each individual output bit depends only on the seed and on the response to a single oracle query. We begin by formalizing this class of constructions.

Definition 5. (Bitwise-nonadaptive construction) *Let $\ell_0(n)$, $\ell_1(n)$, and $\ell_2(n)$ be polynomials, and let $G^{(\cdot)} : \{0,1\}^{\ell_0(n)+\ell_1(n)} \to \{0,1\}^{\ell_0(n)+\ell_2(n)}$ be a non-adaptive oracle machine. We say that $G^{(\cdot)}$ is* bitwise-nonadaptive *if there exist uniformly-computable functions*

$$Q_G = \left\{ Q_{G,n} : \{0,1\}^{\ell_1(n)} \times \{0,1\}^{\log \ell_2(n)} \to \{0,1\}^n \right\}$$

and

$$B = \left\{ B_n : \{0,1\}^{\ell_0(n)} \times \{0,1\}^{\ell_1(n)} \times \{0,1\}^n \times \{0,1\}^{\log \ell_2(n)} \to \{0,1\} \right\}$$

such that for all n, all $r \in \{0,1\}^{\ell_0(n)}$, all $x \in \{0,1\}^{\ell_1(n)}$, and all permutations $\pi : \{0,1\}^n \to \{0,1\}^n$, we have $G^\pi(r||x) = r||b_0||b_1||\ldots||b_{\ell_2(n)-1}$ where $b_i = B_n(r, x, \langle i \rangle, \pi(Q_{G,n}(x, \langle i \rangle)))$ for $0 \leq i \leq \ell_2(n) - 1$.

Observe that the Goldreich-Levin-based pseudo-random generator $G^\pi(r||x) = r||\pi(x)||\langle r, x \rangle$ is bitwise-nonadaptive.

We show that fully black-box bitwise-nonadaptive constructions $G^{(\cdot)}$ making queries to a one-way permutation, such that the non-public portion of the seed of $G^{(\cdot)}$ is no more that $O(\log n)$ bits longer than the length n of each oracle query, cannot achieve linear stretch.

Theorem 5. *Let $\alpha > 1$, and let $\ell_0(n)$, $\ell_1(n)$, and $\ell_2(n)$ be polynomials such that $\ell_1(n) < n + O(\log n)$ and $\ell_2(n) \geq \alpha \cdot \ell_1(n)$. Let $G^{(\cdot)} : \{0,1\}^{\ell_0(n)+\ell_1(n)} \to \{0,1\}^{\ell_0(n)+\ell_2(n)}$ be a bitwise-nonadaptive number generator that makes queries to a permutation on $\{0,1\}^n$. Then there is no fully black-box reduction of the pseudo-randomness of $G^{(\cdot)}$ to the one-wayness of its oracle.*

To prove Theorem 5, we proceed in a manner similar to the proof of Theorem 2, building up a set $Good'(n)$ whose purpose is similar to the set $Good(n)$ in that proof. The fact that each output bit of G depends only a single oracle query simplifies the construction of $Good'(n)$. Specifically, when constructing $Good'(n)$, we can ignore some of the "more difficult to deal with" queries made by $G^{(\cdot)}$, since we can later define adversary A to also ignore these queries simply by ignoring the corresponding output bits. This is what allows us to handle linearly-many queries in the current setting, even though we could only handle constantly-many queries in the proof of Theorem 2.

Proof details are deferred to the full version of this paper.

7 Some remarks on Streaming Cryptography

The study of non-adaptivity in Goldreich-Levin-like constructions (Theorem 5) is motivated by questions related to *Streaming Models for Cryptography.* In some sense, impossibility results for non-adaptive black-box-constructions indicate the impossibility of certain type of black-box streaming constructions. We ask whether there is anything positive that can be said in the streaming setting, perhaps using non-black-box techniques. In this section, we put forward the main questions in streaming models for cryptography. Here is the main motivating question:

> *Starting from generic assumptions, is it possible to construct a one-way function or a pseudo-random generator using $O(\log n)$ space and a small ($1, 2, \ldots,$ constant, polylog) number of passes over the seed?*

Why logarithmic space? Observe that assuming the existence of 2^{n^ϵ}-hard one-way functions (resp. one-way permutations), we can easily construct a one-way function (resp. pseudo-random generator) that uses *poly*-logarithmic space and *reads its input once.* By "2^{n^ϵ}-hard", we mean functions that are hard to invert with probability $\geq 1/2^{n^\epsilon}$ in time $\leq 2^{n^\epsilon}$. Computing such functions in logarithmic space without the ability to recompute (by revisiting the input) seems counter-intuitive. In fact, one can show that unconditionally this cannot be done with any constant number of passes (see the full version of this paper). Are super-constantly-many passes sufficient?

Motivation and related work. Streaming cryptography is motivated both from a theoretical and a practical viewpoint. The practical impact is in settings where on-line or streaming computation of a cryptographic primitive is needed. Theoretical motivation comes from the general theme of computing cryptographic primitives using rudimentary resources. Most relevant to streaming cryptography is the seminal work of Applebaum, Ishai, and Kushilevitz [2, 1, 4, 3], which builds upon the work of Randomizing Polynomials (e.g. [9]), and shows the possibility of *Cryptography in* NC^0: given a "cryptographic function" f, construct a randomized encoding of f, which is a distribution $\{\hat{f}\}$ that (i) preserves the security of f, and (ii) is much simpler to compute than f. This amazing technical achievement brings the combinatorics of cryptographic functions to a simplified setting, and opens the possibility of better understanding cryptographic primitives and non-black-box techniques.

Goals and observations. We wish to be able to state a theorem of the form: if one-way functions exist then one-way functions computable in a streaming manner exist. We believe that this is a difficult thing to show. A potentially more feasible goal would be to show: if 2^{n^ϵ}-hard one-way functions exist then log-space streaming cryptography exists. In fact, by relying on [1, 7], one can easily obtain a *non-black-box* construction of a one-way function computable in $O(\log n)$ space with $\log^{O(1)} n$ passes over the input, assuming that both (i) 2^{n^ϵ}-hard one-way functions exist, and (ii) log-space computable one-way functions exist; see the full version of this paper for the details. The latter assumption refers to functions that are just super-polynomially hard, computable with $n^{O(1)}$ many passes. It seems challenging to do the construction relying only on the existence of 2^{n^ϵ}-hard one-way functions. One can take this further to conjecture that it is possible to prove the following statement in some *constructive* way:

> 2^{n^ϵ}*-hard one-way functions exist* $\iff$ $O(\log n)$ *streaming one-way functions exist* $\iff$ *one-way functions computable in* NC^0 *exist*

This is a rather ambitious research direction. In particular, the right-to-left implication is a hardness amplification of some sort. Our intuition is that streaming computation of functions, functions computable by NC^0 circuits of some restricted form (e.g. of bounded treewidth), and 2^{n^ϵ}-hard one-way functions seem to be related.

References

1. Applebaum, B., Ishai, Y., Kushilevitz, E.: Computationally private randomizing polynomials and their applications. Computational Complexity 15(2), 115–162 (2006), (also CCC'05)
2. Applebaum, B., Ishai, Y., Kushilevitz, E.: Cryptography in NC^0. SIAM J. Comput 36(4), 845–888 (2006), (also FOCS'04)
3. Applebaum, B., Ishai, Y., Kushilevitz, E.: Cryptography with constant input locality. In: CRYPTO'07. pp. 92–110. No. 4622 in Lecture Notes in Computer Science, Springer (2007)
4. Applebaum, B., Ishai, Y., Kushilevitz, E.: On pseudorandom generators with linear stretch in NC^0. Comput. Complexity 17(1), 38–69 (2008), (also RANDOM'06)
5. Gennaro, R., Gertner, Y., Katz, J., Trevisan, L.: Bounds on the efficiency of generic cryptographic constructions. SIAM J. Comput 35(1), 217–246 (2005)

6. Goldreich, O., Levin, L.A.: A hard-core predicate for all one-way functions. In: STOC'89. pp. 25–32. ACM, Berlin (1989)
7. Haitner, I., Reingold, O., Vadhan, S.: Efficiency improvements in constructing pseudorandom generators from one-way functions. In: STOC '10. pp. 437–446. ACM, New York, NY, USA (2010)
8. Impagliazzo, R., Rudich, S.: Limits on the provable consequences of one-way permutations. In: STOC '89 (1989)
9. Ishai, Y., Kushilevitz, E.: Randomizing polynomials: a new representation with applications to round-efficient secure computation. In: Young, D.C. (ed.) FOCS '00. pp. 294–304. IEEE Computer Society (2000)
10. Lu, C.J.: On the complexity of parallel hardness amplification for one-way functions. In: TCC '06. pp. 462–481. LNCS (2006)
11. Miles, E., Viola, E.: On the complexity of increasing the stretch of pseudorandom generators. In: TCC '11: Proceedings of the 8th Theory of Cryptography Conference (2011)
12. Reingold, O., Trevisan, L., Vadhan, S.: Notions of reducibility between cryptographic primitives. In: TCC '04. pp. 1–20. LNCS (2004)
13. Viola, E.: On constructing parallel pseudorandom generators from one-way functions. In: CCC '05. pp. 183–197. IEEE Computer Society (2005)