Electronic Colloquium on Computational Complexity, Report No. 5 (2012)

A remark on one-wayness versus pseudorandomness

Periklis A. Papakonstantinou Guang Yang

Institute for Theoretical Computer Science
Tsinghua University

Abstract

Every pseudorandom generator is in particular a one-way function. If we only consider part of the output of the pseudorandom generator is this still one-way? Here is a general setting formalizing this question. Suppose $G : \{0,1\}^n \to \{0,1\}^{\ell(n)}$ is a pseudorandom generator with stretch $\ell(n) > n$. Let $M_R \in \{0,1\}^{m(n)\times\ell(n)}$ be a linear operator computable in polynomial time given randomness R. Consider the function

$$F(x, R) = \big(M_R G(x), R\big)$$

We obtain the following results.

- There exists a pseudorandom generator such that for every constant $\mu < 1$ and for an arbitrary polynomial time computable $M_R \in \{0,1\}^{(1-\mu)n\times\ell(n)}$, F is not one-way.

  Furthermore, our construction yields a tradeoff between the hardness of the pseudorandom generator and the output length $m(n)$. For example, given $\alpha = \alpha(n)$ and a 2^{cn}-hard pseudorandom generator we construct a $2^{\alpha cn}$-hard pseudorandom generator such that F is not one-way, where $m(n) \le \beta n$ and $\alpha + \beta = 1 - o(1)$.
- We show this tradeoff to be tight for 1-1 pseudorandom generators. That is, for any G which is a $2^{\alpha n}$-hard 1-1 pseudorandom generator, if $\alpha + \beta = 1 + \epsilon$ then there is $M_R \in \{0,1\}^{\beta n\times\ell(n)}$ such that F is a $\Omega(2^{\epsilon n})$-hard one-way function.

1 Introduction

A one-way function is a function easy to compute but hard to invert. A pseudorandom generator is an efficient deterministic algorithm that stretches a short random seed to a longer one which is hard to distinguish from random. They are both fundamental primitives in private-key cryptography.

We tend to believe that one-wayness is a weaker notion than pseudorandomness. One reason is that every pseudorandom generator is in particular a one-way function, but the other direction fails dramatically. In this paper we consider the effect on the one-wayness of a pseudorandom generator when "hashing" its output. A natural way to formalize this is to consider the application of an efficiently sampleable linear operator, which also captures (but a minor twist) universal families of hash functions and certain randomness extractors. Formally, let $G : \{0,1\}^n \to \{0,1\}^{\ell(n)}$, $\ell(n) > n$ be a pseudorandom generator, and fix an arbitrary polynomial time algorithm that on input R it outputs a matrix $M_R \in \{0,1\}^{m(n)\times\ell(n)}$. Consider the following "hashing method":

$$F^G(x, R) = \big(M_R G(x), R\big)$$

We study the effect of the size of $m(n)$ on the one-wayness of F^G. In fact, all of our results hold for affine $\mathbf{F}(\mathbf{x}, \mathbf{R}) = \big(\mathbf{M_R G}(\mathbf{x}) + \mathbf{b_R}, \mathbf{R}\big)$ as well.

ISSN 1433-8092

1.1 Previous work and motivation

Studying relations among basic cryptographic primitives is fundamental for cryptography. Since the seminal work of Håstad-Impagliazzo-Levin-Luby [HILL89], the first to construct a pseudorandom generator from any one-way function, there is a line of excellent works (e.g. [HRV10, HHR06a, HHR06b]) improving its efficiency. Questions regarding the other direction have so far been neglected[1].

Instead of asking whether one-wayness is preserved when hashing the output of every pseudorandom generator, we can ask the weaker question of whether there exists a pseudorandom generator that has this property. If we have such a pseudorandom generator at hand (also enjoying an additional mild property), then via an adaptation of the work of Applebaum-Ishai-Kushilevitch [AIK04, AIK05] we can implement cryptographic primitives in a streaming fashion. *Streaming Cryptography* [BJP11], not to be confused with stream ciphers, concerns the computation of cryptographic primitives with a device that has small working memory, e.g. logarithmic or sub-linear, and it makes a small number of passes, e.g. poly-logarithimic, over its input. Our results rule out a certain class of constructions in Streaming Cryptography.

1.2 Our results

We have obtained both negative and positive results. We show that there exists a pseudorandom generator where if we apply a length-shrinking, even by a constant factor, linear operator on its output then this *is not* a one-way function. Our construction (Theorem 1) yields a tradeoff between the hardness of this generator and the shrinkage factor. Theorem 1 is also, in particular, about universal families of hash functions, but with a minor detail regarding the zero input vector (this does not affect the results). In Theorem 2 we show that this construction is optimal, in the sense that if instead we use any generator which is a little harder, or if the shrinkage factor is a little bigger, then the resulting function *is* one-way.

Theorem 1. *Suppose G is a pseudorandom generator with hardness $s_G(\cdot)$. Then, there is a pseudorandom generator $G^* : \{0,1\}^n \to \{0,1\}^{\ell(n)}$ for an arbitrary polynomial $\ell(n)$, such that $F^{G^*}(x, R) = (M_R G^*(x), R)$ is not one-way, where $M_R \in \mathbb{F}_2^{m(n)\times\ell(n)}$ is a linear operator sampled in polynomial time using randomness R, $m(n) \le (1-\mu)n$ for any constant $\mu > 0$. Moreover, G^* preserves the injectivity of G and its hardness is at least $s_G(\mu n - n^\delta)$ for every $\delta > 0$.*

The "moreover" part makes the theorem stronger. Also, preserving injectivity in this theorem finds application in explaining a subtle issue regarding the optimal output length of hash functions in the first step of [HILL89] construction (see Section 4 in [HILL89], or p.138 in [Gol01]).

Here is a variant of Theorem 1 restricted to projections (i.e. when we just sample from the output of the pseudorandom generator) of size $O(\frac{n}{\log(n)})$.

Lemma 1. *If M_R in Theorem 1 is restricted to random projections with $m(n) = O(\frac{n}{\log(n)})$, then there exists (some other) G^* such that F^{G^*} is invertible in non-uniform NC^2.*

On the other hand, we prove that when hashing a 2^{cn}-hard pseudorandom generator to a little more than $(1-c)n$ bits then its one-wayness is preserved.

[1] This is not surprising, since a pseudorandom generator is in particular a one-way function.

Theorem 2. *Suppose $f : \{0,1\}^n \to \{0,1\}^{\ell(n)}$ is a 2^{cn}-hard 1-1 pseudorandom generator. Let $F := F^f(x,h) = \big(h(f(x)), h\big)$, where $h : \{0,1\}^{\ell(n)} \to \{0,1\}^{m(n)}$ is a hash function from a universal family of hash functions $S_{\ell(n)}^{m(n)}$. If $m(n) \geq (1-c+\epsilon)n$ for constant $\epsilon \in (0, \frac{c}{5})$, then F is one-way with hardness $2^{\epsilon n}$.*

In fact, the above theorem holds true if instead of a pseudorandom generator we consider f to be an injective one-way function.

1.3 Outline

In Section 2, we introduce notation, definitions, and basic facts. In Section 3, we construct G^* from a pseudorandom generator G such that F^{G^*} is not one-way when hashing down its output by a constant factor. In Section 4 we show that for every 1-1 pseudorandom generator f with hardness 2^{cn} and $m(n) \geq (1-c+\epsilon)n$, F^f preserves the one-wayness and has hardness at least $2^{\epsilon n}$. We conclude in Section 5 with some further research directions.

2 Preliminary

2.1 Notation and definitions

Probability notation. For probability distributions X, Y, we denote by $X \sim Y$ that X and Y are identically distributed. $x \leftarrow X$ denotes that x is sampled from X, and $x \in_R S$ denotes that x is sampled uniformly from S. U_n denotes the uniform distribution over $\{0,1\}^n$. The *statistical distance* between two distributions X and Y is defined as $\Delta(X,Y) = \frac{1}{2}\sum_z |\Pr[X=z] - \Pr[Y=z]|$.

Universal families of hash functions. Let S_n^m denote a set of functions from $\{0,1\}^n$ to $\{0,1\}^m$. Let H_n^m be a random variable uniformly distributed over S_n^m. S_n^m is called a *universal family of hash functions* if the following conditions hold:

- S_n^m is a pairwise independent family of mappings: for every $x \neq y \in \{0,1\}^n$, $H_n^m(x)$ and $H_n^m(y)$ are independent distributions and both identically distributed to U_m.
- S_n^m has a succinct representation: for every $h \in S_n^m$, the description of h is $\mathsf{poly}(n,m)$.
- S_n^m can be efficiently evaluated: there is a polynomial time algorithm $\mathcal{H}$ such that for every $h \in S_n^m, x \in \{0,1\}^n$, $\mathcal{H}(h,x) = h(x)$.

Specifically, $h(x) = M \cdot x + b$ is a universal family of hash functions when the matrix M and vector b are uniformly distributed. In fact, $h(x) = M \cdot x$ satisfies all above conditions except that $H_n^m(x)$ is not uniformly distributed at the point $x = \mathbf{0}$.

Cryptographic primitives. Here are the definitions of one-way functions, pseudorandom generators, and k-wise independent distributions. The definitions are for uniform adversaries, however our results hold in the non-uniform setting as well (c.f. [Gol01, Vad11]).

A *one-way function* $f : \{0,1\}^* \to \{0,1\}^*$ is a polynomial time computable function where no probabilistic polynomial time algorithm A inverts f with non-negligible probability; i.e. for every k and every probabilistic polynomial time algorithm A, we have $\Pr_{x \leftarrow U_n}[A(f(x), 1^n) \in f^{-1}(f(x))] < n^{-k}$, for sufficiently large n.

Furthermore, we say that f has *hardness* $s(n)$ if for every sufficiently large input of length n, f cannot be inverted with probability $\geq \frac{1}{s(n)}$ by any adversary A which runs in time $\leq s(n)$. Therefore, f is a one-way function if f has super-polynomial hardness $s(n)$.

A *pseudorandom generator* G is a polynomial time computable function which stretches every input x to an output of length $|G(x)| = \ell(|x|) > |x| = n$, such that every probabilistic polynomial time algorithm D cannot distinguish between $U_{|G(x)|}$ and $G(U_{|x|})$; i.e. for every k and D, $|\Pr[D(G(U_n), 1^n) = 1] - \Pr[D(U_{|G(U_n)|}, 1^n) = 1]| < n^{-k}$ when n is sufficiently large. We call ℓ the *stretch of* G. Similar to one-way functions we define an $s(n)$-hard pseudorandom generator.

We subscript a string $\sigma \in \{0,1\}^n$ with $R \subseteq \{1, \dots, n\}$, and we write σ_R, to denote the substring of σ keeping exactly the bits indexed by R. In this notation, a function h is called k-wise independent if for every $K \subseteq \{1, \dots, n\}$ where $|K| = k$ we have that $h(U_n)_K \sim U_k$.

Circuit classes. We denote by NC^2 the functions computed by *non-uniform* families of poly-size boolean circuits with *multiple outputs*, where the gates are of constant fan-in and the depth of the circuit is $O(\log^2 n)$ for input length n.

2.2 Basic facts and lemmas

The following is a well-known fact (see e.g. [Gol01]).

Lemma 2. *Let G be a pseudorandom generator. Then, G is a one-way function.*

The following lemma states that a uniform randomly chosen matrix has a good chance of being row independent. In fact, more general results hold for $n \times n$ matrices (see e.g. [BKW97, Muk84]). The proof of the following lemma is an easy exercise and is given for completeness in the appendix.

Lemma 3. *Uniformly at random pick a $p \times q$ matrix N over $\mathbb{F}_2$; i.e. $N \in_R \mathbb{F}_2^{p \times q}$. Then, N has full row-rank with probability at least $1 - 2^{p-q}$.*

A deep result due to Mulmuley [Mul87] (which derandomizes [BvzGH82]) states that Gauss elimination for linear systems over $\mathbb{F}_2$ can be done in uniform NC^2. Later on, when we apply this lemma, we introduce non-uniformity for a different reason.

Lemma 4 ([Mul87])**.** *Gauss elimination can be done in uniform NC^2.*

3 Length-shrinking linear operators destroy one-wayness: a shrinkage-hardness tradeoff

We prove Theorem 1. That is, given a pseudorandom generator G of hardness $s_G(n)$ we construct a pseudorandom generator G^* of *almost* the same hardness $s_{G^*}(n) = s_G\big((\mu - o(1))n\big)$ for some constant μ, such that an application of any efficiently sampled linear operator, which outputs $(1-\mu)n$ bits, on the output of G^* does not preserve one-wayness.

First we introduce the construction of G^*. It is easy to see that it preserves pseudorandomness and injectivity; i.e. if G is 1-1 then G^* is also 1-1.

Construction 1. *Construct G^* as*

$$G^*(x_1, x_2, x_3) = (\hat{G}(x_1) + (P_G(x_3) \cdot x_2), x_2, x_3) \tag{1}$$

$|x_1| = n_1$, $|x_2| = n_2$, $|x_3| = n_3$, $n_1 + n_2 + n_3 = n$. $\hat{G}(x_1) = G^{(z)}(x_1)|_{\{1,2,\cdots,\ell'(n)\}}$ where $G^{(z)}$ denotes z iterated compositions of G with itself for sufficient large z, such that $|G^{(z)}(x_1)| \geq \ell'(n) = \ell(n) - n_2 - n_3$. $P_G(x_3)$ is an $\ell'(n) \times n_2$ matrix generated by iteratively applying pseudorandom generator G and random seed x_3. All operations are over $\mathbb{F}_2$.

By definition of $\hat{G}$, $|\hat{G}(x_1)| = \ell'(n)$. That is, $|G^*(x_1, x_2, x_3)| = \ell'(n) + n_2 + n_3 = \ell(n)$. Since we XOR $\hat{G}(x_1)$ with $P_G(x_3) \cdot x_2$, then $s_G(n_1)$ lower bounds the hardness of $G^*(x)$. We can choose n_3 to be an arbitrarily small polynomial in n. The parameters n_1 and n_2 determine a tradeoff between the hardness of the pseudorandom generator G^* and the shrinking length. This tradeoff is not a minor issue. If we were to choose arbitrarily close to 1 the constants in the hardness and in the shrinking length then a modification of [HILL89] would have shown that exponentially hard pseudorandom generators, unconditionally, do not exist (this is not an immediate argument).

The following lemma is the main ingredient of the proof of Theorem 1.

Lemma 5. *Suppose $F^{G^*}(x, R) = (M_R G^*(x), R)$ and $G^*(x_1, x_2, x_3)$ be as in Construction 1. Let $M_R \in \{0,1\}^{m(n) \times \ell(n)}$, $m(n) < n_2$, be computable in polynomial time given R. Then, there exists a probabilistic polynomial time algorithm A such that*

$$\Pr_{y,R}[F^{G^*}(A(y, R)) = (y, R)] > 1 - 2^{-(n_2 - m(n))} - \mathsf{poly}(\frac{1}{s_G(n_3)})$$

Proof. Recall that $G^*(x_1, x_2, x_3) = \big(\hat{G}(x_1) + (P_G(x_3) \cdot x_2), x_2, x_3\big)$, where $x = (x_1, x_2, x_3)$ and x_1, x_2, x_3 has length n_1, n_2, n_3 respectively. Then,

$$F^{G^*}(x, R) = \big(M_R G^*(x), R\big) = \big(M_R(\hat{G}(x_1) + (P_G(x_3) \cdot x_2), x_2, x_3), R\big)$$

Therefore for the goal $F^{G^*}(x, R) = (y, R)$, it suffices to find an x such that

$$M_R\big(\hat{G}(x_1) + (P_G(x_3) \cdot x_2), x_2, x_3\big) = y \tag{2}$$

We analyze further the structure of the above matrix equation. Without loss of generality, we may assume that M_R is already in reduced row echelon form, after applying Gauss elimination, and it has full row-rank (easy to guarantee by deleting all zero rows). To match the form of the column vector $\big(\hat{G}(x_1) + (P_G(x_3) \cdot x_2), x_2, x_3\big)$, we partition M_R into $M_R = (M_1 | M_2 | M_3)$ where the sub-matrices M_1, M_2, M_3 have $\ell'(n), n_2$ and n_3 columns respectively. Then

$$M_R = \begin{pmatrix} M_1 & M_2 & M_3 \end{pmatrix} = \begin{pmatrix} M_1' & M_2'' & M_3''' \\ 0 & M_2' & M_3'' \\ 0 & 0 & M_3' \end{pmatrix}$$

where M_1', M_2' and M_3' have full row-rank. Note that depending on M_R, it is possible that M_2', M_3' and M_3'' are empty (i.e. size 0, instead of having 0-entries). Equation (2) can be written as

$$\left\{\begin{array}{rrrl} M_1'\big(\hat{G}(x_1) + P_G(x_3)x_2\big) + & M_2''x_2 + & M_3'''x_3 & = y_1 \\ & M_2'x_2 + & M_3''x_3 & = y_2 \\ & & M_3'x_3 & = y_3 \end{array}\right. \tag{3}$$

Rewriting it as a linear system in x_2,

$$\left\{\begin{array}{rrcll} \big(M_1'P_G(x_3) + & M_2''\big) & x_2 & = y_1 & + M_3'''x_3 + M_1'\hat{G}(x_1) \\ & M_2' & x_2 & = y_2 & + M_3''x_3 \\ & & \mathbf{0} & = y_3 & + M_3'x_3 \end{array}\right. \tag{4}$$

Now the problem reduces to finding a solution x to (4). We present an adversary A which finds a solution to the above system.

A : INVERTING F^{G^*} (on input (y, R)):

Compute M_R with input R;
Do Gauss elimination on the left of $(M_R|y)$;
Delete zero-rows and return "No answer" if detecting a row $(0, 0, 0, \cdots, 0, 1)$;
Compute $M_1', M_2', M_2'', M_3', M_3'', M_3'''$;
Set x_1 to a fixed value u, say n_1 zeros;
Uniformly at random pick v from $\{x_3 \big| M_3' x_3 = y_3\} \subseteq \{0,1\}^{n_3}$ ($v \leftarrow U_{n_3}$ if M_3' is empty);
Compute $P_G(v)$ and $\hat{G}(u)$;
Consider: $\begin{pmatrix} M_1' P_G(v) + M_2'' \\ M_2' \end{pmatrix} x_2 = \begin{pmatrix} y_1 + M_1' \hat{G}(u) + M_3''' v \\ y_2 + M_3'' v \end{pmatrix}$;
Solve x_2 and output $(x, R) = ((u, x_2, v), R)$. Output "Fail" if there is no solution.

It is easy to verify that A runs in polynomial time and the output is a pre-image of (y, R). Now, we analyze the probability that A succeeds. It suffices to calculate the probability that A outputs "Fail", which is upper bounded by the probability that $\mathcal{M} = \begin{pmatrix} M_1' P_G(v) + M_2'' \\ M_2' \end{pmatrix}$ does not have full row-rank. Let $\mathcal{M}' = \begin{pmatrix} M_1' \cdot U_{\ell'(n) \times n_2} + M_2'' \\ M_2' \end{pmatrix}$. Since M_1', M_2' have full row-rank, $\mathcal{M}' \sim \begin{pmatrix} U_{r_1 \times n_2} \\ M_2' \end{pmatrix}$ does not have full row-rank with probability at most $\sum_{1 \le i \le r_1} \frac{2^{r_2+i-1}}{2^{n_2}} < \frac{2^{r_1+r_2}}{2^{n_2}} = 2^{-(n_2-r_1-r_2)}$ by Lemma 3, where r_1, r_2 is the number of rows in M_1', M_2' respectively. Moreover, the gap between $\Pr[\mathcal{M}$ has full row-rank$]$ and $\Pr[\mathcal{M}'$ has full row-rank$]$ is bounded by $\mathsf{poly}(\frac{1}{s_G(n_3)})$, since otherwise there exists a polynomial time distinguisher for $P_G(v)$ and $U_{\ell'(n) \times n_2}$ with advantage at least $\mathsf{poly}(\frac{1}{s_G(n_3)})$. Therefore, we have

$$\begin{aligned} \Pr[\mathcal{M} \text{ has full row-rank}] &\ge \Pr[\mathcal{M}' \text{ has full row-rank}] - \mathsf{poly}(\frac{1}{s_G(n_3)}) \\ &\ge 1 - 2^{-(n_2-r_1-r_2)} - \mathsf{poly}(\frac{1}{s_G(n_3)}). \end{aligned}$$

Since M_R has $m(n)$ rows in total, which implies $r_1 + r_2 \le m(n)$,

$$\Pr_y[A \text{ succeeds}] \ge \Pr[\mathcal{M} \text{ has full row-rank}] \ge 1 - 2^{-(n_2 - m(n))} - \mathsf{poly}(\frac{1}{s_G(n_3)})$$

Thus completes our proof of Lemma 5. $\square$

Corollary 1. *If $m(n) \le n_2 - \omega(\log(n))$ and $n_3 = n^{\Omega(1)}$, then $F^{G^*}(x, R) = \big(M_R G^*(x), R\big)$ is not (even weakly) one-way.*

Let $n_1 = \mu n - n^\delta$, $n_2 = (1-\mu)n + \log^2(n)$, and $n_3 = n - n_1 - n_2 = n^\delta - \log^2(n)$ in Construction 1 and $m(n) = n_2 - \log^2(n) = (1-\mu)n$. Applying Lemma 5 and Corollary 1, we conclude the proof of Theorem 1. Therefore in general, hashing down the output of a pseudorandom generator by a constant factor does not preserve its one-wayness, even when the pseudorandom generator is exponential hard.

Regarding the roles of n_1, n_2, n_3 in above argument, we first notice that n_3 is the least important one since we only need $s_G(n_3)$ super-polynomial. In most common cases of interest $s_G(\cdot)$ is monotonically increasing (hence, s_G^{-1} is well defined), it suffices to set $n_3 = s_G^{-1}(n^{\omega(1)})$ which could be as small as $\log^{O(1)}(n)$ when s_G is exponential. Meanwhile, the difference between $m(n)$ and n_2 is also negligible. Therefore it turns out $n_1 + m(n) = n - o(n)$. Recalling that G^* has hardness $s_G(n_1)$, thus there is a tradeoff between the hardness of G^* and the output length of M_R. Letting $n_1 = \alpha n$ and $m(n) = \beta n$, we get $\alpha + \beta = 1 - o(1)$ as stated in the abstract.

Special case of random projections. When M_R is a projection of length $O(\frac{n}{\log n})$ we construct a simpler pseudorandom generator G^* where F^{G^*} can even be inverted in NC^2. For this we combine the "strong pseudorandom" (cryptographic) object G with a "weak pseudorandom" object, a k-wise independent generator. Specifically, let $G^*(x_1, x_2) = (\hat{G}(x_1) + Hx_2)$ where H realizes a k-wise generator with $k = \Theta(\frac{n}{\log(n)})$. See Proposition 6.5 in [ABI86] and Chap. 7.6 in [MS77] for details.

Lemma 6. *Let $m(n) \leq k$, where k is as above. Then, $F^{G^*}(x, R) = \big(M_R G^*(x), R\big)$ can be inverted in NC^2.*

The adversary is a modification of A which appears in the proof of Lemma 5. In particular, in Step 4, only M_1' matters since other matrices are 0-sized; in Step 6,7,8, $P_G(v)$ is replaced by H and the linear system in Step 8 becomes $M_1' H x_2 = y_1 + \hat{G}(u)$. Although $\hat{G}$ is polynomial time computable, we can non-uniformly hardwire the value of $\hat{G}$ on a constant one for each input length. Since u can be fixed, then by Lemma 4 we have that $M_1' H$ is invertible in NC^2.

4 Tightness of the construction

Even if we assume that a pseudorandom generator of hardness $2^{0.99n}$ exists, Theorem 1 says that then there is a generator of hardness $2^{0.99\alpha n}$ such that when applying a linear map on its output shrinking it down to βn many bits then this is not one-way, for $\alpha + \beta = 1 - o(1)$. We show that this tradeoff between α and β is tight, i.e. when $\alpha + \beta = 1 + \epsilon$ and a 1-1 generator f has hardness $2^{\alpha n}$, then F^f forms a $2^{\epsilon n}$-hard one-way function.

Below, we prove of Theorem 2. For this we apply the following well-known lemma, but in a non-uniform setting.

Lemma 7 ([Gol01] Lemma 3.5.1, or e.g. [HILL89, Sip83, GL89])**.** *Let $m < \ell$ be integers, S_ℓ^m be a universal family of hash functions, and b,δ be two reals such that $m \leq b \leq \ell$ and $\delta \geq 2^{-\frac{b-m}{2}}$. Suppose that X_ℓ is a random variable distributed over $\{0,1\}^\ell$ such that for every x, it holds $\Pr[X_n = x] \leq 2^{-b}$. Then for every $\xi \in \{0,1\}^m$ and for all but at most $2^{-(b-m)}\delta^{-2}$ fraction of the h's in S_ℓ^m, it holds that*

$$\Pr_{X_\ell}[h(X_\ell) = \xi] \in (1 \pm \delta) 2^{-m}$$

Proof of Theorem 2. We present the proof for a non-uniform adversary, simpler to present but already a rather involved argument. Fix one efficient construction of sampling from a universal family of hash functions (e.g. choose one from [Vad11]). Now F is well-defined for a given f.

Assume that F is not a $2^{\epsilon n}$-hard one-way function. Let A be a probabilistic algorithm which runs in time $T_A = O(2^{\epsilon n})$ and inverts F with probability $p_A(n)$, i.e.

$$\Pr_{x \leftarrow U_n, h \leftarrow_R S_{\ell(n)}^{m(n)}}[A(h(f(x)), h) \in F^{-1}(h(f(x)), h)] = p_A(n) > \frac{1}{2^{\epsilon n}}$$

We show that f is not 2^{cn}-hard with oracle access to A. That is, we construct a non-uniform adversary A_f that given $y \leftarrow f(U_n)$, A_f computes x' such that $f(x') = y$ in time $O(2^{cn})$ and with probability at least $\Omega(2^{-cn})$.

Here is the description of A_f: suppose the non-uniform advice is $h_0 \in S_{\ell(n)}^{m(n)}$, where h_0 depends on n, and $y \leftarrow f(U_n)$ is the input. A_f first computes $(h_0(y), h_0)$. Then apply A on $(h_0(y), h_0)$ to compute x' such that $h_0(f(x')) = h_0(y)$.

Therefore, A_f runs in time $O(T_A) = O(2^{\epsilon n}) = O(2^{cn})$. In what follows we denote by $x' = x'(h(y), h)$ the output of A on input $(h(y), h)$. Now, we calculate the probability that A_f outputs x'. We will determine later how to find h_0, in fact why h_0 exists.

$$\Pr_{y \leftarrow f(U_n)}[A_f \text{ inverts } f \text{ on } y] = \Pr_{y \leftarrow f(U_n)}[x' = A(h_0(y), h_0), f(x') = y] = \Pr_{x \leftarrow U_n}[f(x') = f(x)] \tag{5}$$

where in the last equation we omit writing how x' is derived and mentioning its dependence.

$$\begin{aligned}
&\Pr_{x \leftarrow U_n}[f(x') = f(x)] \\
=& \sum_{z \in h_0(f(\{0,1\}^n))} \Pr_{x \leftarrow U_n}[h_0(f(x)) = z] \Pr_{x \leftarrow U_n}[f(x') = f(x) \big| h_0(f(x)) = z] \\
=& \sum_{z \in h_0(f(\{0,1\}^n))} \Pr_{x \leftarrow U_n}[h_0(f(x)) = z] \Pr_{x \in_R (h_0 \circ f)^{-1}(z)}[x = x' = x'(z, h_0)]
\end{aligned}$$

$f(x') = f(x)$ is equivalent to $x' = x$ since f is 1-1. From this point on, $x'(z, h_0)$ is uniquely defined and constant from z and h_0. Therefore we can take it out of the probability.

$$\begin{aligned}
=& \sum_{z \in h_0(f(\{0,1\}^n))} \frac{|(h_0 \circ f)^{-1}(z)|}{2^n} \cdot \big(\frac{1}{|(h_0 \circ f)^{-1}(z)|} \cdot I[h_0(f(x'(z, h_0))) = z]\big) \\
=& \frac{1}{2^n} \sum_{z \in h_0(f(\{0,1\}^n))} I[h_0(f(x')) = z] = \frac{1}{2^n} \sum_{z \in \{0,1\}^m} I[h_0(f(x')) = z]
\end{aligned} \tag{6}$$

where $I[h_0(f(x')) = z]$ is the indicator of the event "$h_0(f(x')) = z$ for $x' = A(z, h_0)$". Note that the sum $\sum_{z \in \{0,1\}^m} I[h_0(f(x')) = z]$ corresponds to the number of z's that A inverts (z, h_0).

However, when fixing h_0, the probability "A succeeds" is

$$\Pr_{x \leftarrow U_n}[A \text{ inverts } (h_0(f(x)), h_0)] = \sum_{z \in \{0,1\}^m} \Pr_{x \leftarrow U_n}[h_0(f(x)) = z] \cdot I[h_0(f(x')) = z] \tag{7}$$

Notice that (7) is the probability of "A succeeds on $\big(h_0(f(U_n)), h_0\big)$", while (6) counts the number of z's that A inverts $\big(z, h_0\big)$. There two are related in the following sense. Remember that hashing down a weak random source smooths the distribution, hence $h_0(f(U_n))$ seems close to U_m. In this sense, we make an estimation with error upper bounded by their statistical distance.

$$
\begin{aligned}
&\Big| \Pr_{x \leftarrow U_n}[A \text{ inverts } (h_0(f(x)), h_0)] - \frac{1}{2^m} \sum_{z \in \{0,1\}^m} I[h_0(f(x')) = z] \Big| \\
=& \Big| \sum_{z \in \{0,1\}^m} \Pr_{x \leftarrow U_n}[h_0(f(x)) = z] \cdot I[h_0(f(x')) = z] - \sum_{z \in \{0,1\}^m} \frac{1}{2^m} I[h_0(f(x')) = z] \Big| \\
\leq & \sum_{z \in \{0,1\}^m} \Big| \Pr_{x \leftarrow U_n}[h_0(f(x)) = z] - \frac{1}{2^m} \Big| \cdot I[h_0(f(x')) = z] \\
=& 2\Delta\big(h_0(f(U_n)), U_m\big)
\end{aligned} \tag{8}
$$

Plugging (8) into (6), it immediately leads to the lower bound

$$
\Pr_{x \leftarrow U_n}[f(x') = f(x)] \geq 2^{m-n}\Big(\Pr_{x \leftarrow U_n}[A \text{ inverts } (h_0(f(x)), h_0)] - 2\Delta\big(h_0(f(U_n)), U_m\big)\Big) \tag{9}
$$

Now, our goal is to show that there exists a choice for h_0 in (9) giving the $\Omega(\frac{1}{2^{cn}})$ lower bound.

Claim 1. *There is a (good) $h_0 \in S_{\ell(n)}^{m(n)}$ such that*

- *Property 1:* $\Delta\big(h_0(f(U_n)), U_m\big) < 2 \cdot 2^{\frac{1+\epsilon n-(n-m)}{3}}$;
- *Property 2:* $\Pr_{x \leftarrow U_n}[h_0(f(x')) = h_0(f(x))] \geq 2^{-(1+\epsilon n)}$.

For Property 1, it suffices for concluding the the proof to have $\delta = 2^{\frac{1+\epsilon n-(n-m)}{3}}$ and

$$
\Pr_{\xi \leftarrow U_m}[\Pr[h_0(f(U_n)) = \xi] \notin (1 \pm \delta) \cdot 2^{-m}] < 2^{1+\epsilon n-(n-m)} \delta^{-2}
$$

Let $\delta = 2^{\frac{1+\epsilon n-(n-m)}{3}}$, $b = n, m = m(n), \ell = \ell(n)$ and $X = f(U_n)$ as in Lemma 7. Since $m \leq b \leq \ell(n)$ and f is injective (so that $\Pr_X[X = z] \leq \frac{1}{2^n}$ for every z), we have that for every $\xi \in \{0,1\}^m$ and for all but at most $2^{-(n-m)}\delta^{-2}$ fraction of the h's in $S_{\ell(n)}^{m(n)}$, it holds $\Pr[h(f(U_n)) = \xi] \in (1 \pm \delta) \cdot 2^{-m}$. Let $\mathcal{B}(h, \xi)$ denote the event $\Pr[h(f(U_n)) = \xi] \notin (1 \pm \delta) \cdot 2^{-m}$, then taking probability over ξ and h,

$$
\begin{aligned}
& \Pr_{\xi \leftarrow U_m, h \leftarrow S_{\ell(n)}^{m(n)}}[\mathcal{B}(h, \xi)] \leq 2^{-(n-m)}\delta^{-2} \\
\Longrightarrow & \Pr_{h \leftarrow S_{\ell(n)}^{m(n)}}\big[\Pr_{\xi \leftarrow U_m}[\mathcal{B}(h, \xi)] \geq 2^{1+\epsilon n-(n-m)}\delta^{-2}\big] \leq \frac{1}{2^{1+\epsilon n}}
\end{aligned} \tag{10}
$$

Thus, $\Pr_{\xi \leftarrow U_m}[\Pr[h(f(U_n)) = \xi] \notin (1 \pm \delta) \cdot 2^{-m}] < 2^{1+\epsilon n-(n-m)}\delta^{-2}$ holds for at least $1 - \frac{1}{2^{1+\epsilon n}}$ fraction of the h's in $S_{\ell(n)}^{m(n)}$. In particular, Property 1 is satisfied by that many h's.

For Property 2, we lower bound the probability that A performs not so bad for a randomly chosen h, i.e. $\Pr_{h \leftarrow S_{\ell(n)}^{m(n)}}[\Pr_{x \leftarrow U_n}[h(f(x')) = h(f(x))] \geq \frac{1}{2^{1+\epsilon n}}]$. Let $\mathcal{E}_h$ denote the event that

$\Pr_{x \leftarrow U_n}[h(f(x')) = h(f(x))] \geq 2^{-1-\epsilon n}$, we have

$$
\begin{aligned}
2^{-\epsilon n} \leq p_A(n) = & \Pr_{h \leftarrow S_{\ell(n)}^{m(n)}, x \leftarrow U_n}[h(f(x')) = h(f(x))] \\
= & \Pr_{h \leftarrow S_{\ell(n)}^{m(n)}}[\mathcal{E}_h] \Pr_{x \leftarrow U_n}[h(f(x')) = h(f(x)) \big| \mathcal{E}_h] + \Pr_{h \leftarrow S_{\ell(n)}^{m(n)}}[\neg \mathcal{E}_h] \Pr_{x \leftarrow U_n}[h(f(x')) = h(f(x)) \big| \neg \mathcal{E}_h] \\
\leq & \Pr_{h \leftarrow S_{\ell(n)}^{m(n)}}[\mathcal{E}_h] \cdot 1 + \Pr_{h \leftarrow S_{\ell(n)}^{m(n)}}[\neg \mathcal{E}_h] \cdot 2^{-1-\epsilon n} = (1 - 2^{-1-\epsilon n}) \Pr_{h \leftarrow S_{\ell(n)}^{m(n)}}[\mathcal{E}_h] + 2^{-1-\epsilon n} \\
\implies & \Pr[\mathcal{E}_h] > 2^{-1-\epsilon n}
\end{aligned}
$$

Hence, we lower bound the probability of h having Property 2 as follows

$$
\Pr_{h \leftarrow S_{\ell(n)}^{m(n)}}[\underbrace{\Pr_{x \leftarrow U_n}[h(f(x')) = h(f(x))] \geq 2^{-1-\epsilon n}}_{\mathcal{E}_h}] > 2^{-1-\epsilon n}
$$

The following calculation shows that an h_0 as required exists.

$$
\Pr_{h \leftarrow S_{\ell(n)}^{m(n)}}[h \text{ satisfies both Property 1 and 2}] > (1 - \frac{1}{2^{1+\epsilon n}}) + 2^{-1-\epsilon n} - 1 = 0
$$

Using this h_0 in (9), and recalling that $m = m(n) = (1 - c + \epsilon)n$, we obtain

$$
\begin{aligned}
& \Pr_{y \leftarrow f(U_n)}[A_f \text{ inverts } f \text{ on } y] = \Pr_{x \leftarrow U_n}[f(x') = f(x)] \\
\geq & 2^{m-n}\big(2^{-(1+\epsilon n)} - 2(2 \cdot 2^{\frac{1+\epsilon n-(n-m)}{3}})\big) = 2^{-1-cn} - 2^{(7+(5\epsilon-4c)n)/3} = \Omega(2^{-cn})
\end{aligned}
$$

Note that the running time of A_f is bounded by $O(2^{cn})$. In conclusion, $F(x,h) = \big(h(f(x)), h\big)$ is one-way, and its hardness is at least $2^{\epsilon n}$. □

5 Conclusions and open questions

We have showed that "hashing" the output of a pseudorandom generator to a constant fraction of its input length, in general, destroys its one-wayness. We prove this in the form of a tradeoff between cryptographic hardness and output length of the hash. We also show that this tradeoff is tight.

The question asked in this paper is of independent interest. It is further motivated by Streaming Cryptography in logarithmic space (see e.g. [KGY89, BJP11]). In particular, our main result precludes the possibility of basing Streaming Cryptography in this specific way on arbitrary pseudorandom generators.

Another question is whether there exists a pseudorandom generator of reasonable hardness where one-wayness is preserved when hashing its output. This question remains open. We speculate that is a difficult mathematical problem. For example, an interesting direction would be to show that this question is equivalent to constructing 2^{n^ϵ}-hard one-way functions; i.e. a problem essentially about $\Omega(2^{n^\epsilon})$ circuit lower bounds.

Finally, we ask whether starting from generic assumptions there is a possibly different avenue to computing cryptographic primitives in a streaming fashion.

Acknowledgements

We would like to thank John Steinberger, and Andrew Wan for the helpful remarks on a previous draft. We would also like to thank Andrej Bogdanov and Oded Goldreich for the helpful discussions.

References

[ABI86] N. Alon, L. Babai, and A. Itai. A fast and simple randomized parallel algorithm for the maximal independent set problem. *Journal of Algorithms*, 7:567–583, 1986.

[AIK05] B. Applebaum, Y. Ishai, and E. Kushilevitz. Computationally private randomizing polynomials and their applications. *Computational Complexity*, 15(2):115–162, 2006 (also CCC'05).

[AIK04] B. Applebaum, Y. Ishai, and E. Kushilevitz. Cryptography in NC^0. *SIAM Journal on Computing (SICOMP)*, 36(4):845–888, 2006 (also FOCS'04).

[BJP11] J. Bronson, A. Juma, and P. A. Papakonstantinou. Limits on the stretch of non-adaptive constructions of pseudo-random generators. In *Theory of Cryptography Conference (TCC)*, pages 504–521, 2011.

[BKW97] J. Blömer, R. Karp, and E. Welzl. The rank of sparse random matrices over finite fields. *Random Structures Algorithms*, 10(4):407–419, 1997.

[BvzGH82] A. Borodin, J. von zur Gathen, and J. Hopcroft. Fast parallel matrix and GCD computations. *Information and Control*, 52(3):241–256, 1982.

[GL89] O. Goldreich and L. A. Levin. A hard-core predicate for all one-way functions. In *Symposium on Theory Of Computing (STOC)*, pages 25–32, 1989.

[Gol01] O. Goldreich. *Foundations of cryptography.* Cambridge University Press, Cambridge, 2001. Basic tools (Vol. I).

[HHR06a] I. Haitner, D. Harnik, and O. Reingold. Efficient pseudorandom generators from exponentially hard one-way functions. In *International Colloquium on Automata, Languages and Programming(ICALP)*, 2006.

[HHR06b] I. Haitner, D. Harnik, and O. Reingold. On the power of the randomized iterate. In *Advances in Cryptology—CRYPTO 2006*, volume 4117, pages 22–40. 2006.

[HILL89] J. Hastad, R. Impagliazzo, L. A. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing (SICOMP)*, 28(4):1364–1396, 1999 (also STOC'89).

[HRV10] I. Haitner, O. Reingold, and S. Vadhan. Efficiency improvements in constructing pseudorandom generators from one-way functions. In *Symposium on Theory Of Computing (STOC)*, pages 437–446, 2010.

[KGY89] M. Kharitonov, A. V. Goldberg, and M. Yung. Lower bounds for pseudorandom number generators. In *Foundations of Computer Science (FOCS)*, pages 242–247, 1989.

[MS77] F. J. MacWilliams and N. J. A. Sloane. *The Theory of Error-Correcting Codes*. North-Holland, 1977.

[Muk84] A. Mukhopadhyay. On the probability that the determinant of an $n \times n$ matrix over a finite field vanishes. *Discrete Math.*, 51(3):311–315, 1984.

[Mul87] K. Mulmuley. A fast parallel algorithm to compute the rank of a matrix over an arbitrary field. *Combinatorica*, 7(1):101–104, 1987.

[Sip83] M. Sipser. A complexity theoretic approach to randomness. In *Symposium on Theory Of Computing (STOC)*, pages 330–335, 1983.

[Vad11] S. Vadhan. Pseudorandomness. April 2011.

A Appendix

A.1 proof of Lemma 3

Proof. Suppose $N \leftarrow U_{p\times q}$ whose row vectors are $\lambda_1, \lambda_2, \cdots, \lambda_p \leftarrow U_q$.

$$
\begin{aligned}
&\Pr_{N\leftarrow U_{p\times q}}[N \text{ has full row-rank}] \\
=&\Pr_{\lambda_1,\cdots,\lambda_p\leftarrow U_q}[\forall \gamma \in \{0,1\}^p \backslash \{\mathbf{0}\}, \gamma\cdot(\lambda_1,\cdots,\lambda_p) \neq 0] \\
=&\Pr_{\lambda_1,\cdots,\lambda_p\leftarrow U_q}[\bigwedge_{1\le i\le p}(\lambda_i \notin \mathsf{span}(\lambda_1,\cdots,\lambda_{i-1}))] \\
\ge& 1-\sum_{1\le i\le p}\Pr_{\lambda_i}[\lambda_i \in \mathsf{span}(\lambda_1,\cdots,\lambda_{i-1})] = 1-\sum_{1\le i\le p}\frac{|\mathsf{span}(\lambda_1,\cdots,\lambda_{i-1})|}{2^q} \\
\ge& 1-\sum_{1\le i\le p}\frac{2^{i-1}}{2^q} \ge 1-2^{p-q},
\end{aligned}
$$

where $\mathsf{span}(\cdot)$ denotes the linear space spanned by these vectors. □

ECCC ISSN 1433-8092

http://eccc.hpi-web.de