Electronic Colloquium on Computational Complexity, Report No. 65 (2003)

Compressibility Lower Bounds In Oracle Settings

Hoeteck Wee
University of California, Berkeley
hoeteck@cs.berkeley.edu

June 18, 2003

Abstract

A source is compressible if we can efficiently compute short descriptions of strings in the support and efficiently recover the strings from the descriptions. In this paper, we present a technique for proving lower bounds on compressibility in an oracle setting, which yields the following results:

1. We exhibit oracles relative to which there exists samplable sources over $\{0,1\}^n$ of low pseudoentropy (say $n/2$) that cannot be compressed to length less than $n - \omega(\log n)$ by polynomial size circuits. This matches the upper bounds in [GS91, TVZ03], and provides an oracle separation between compressibility and pseudoentropy, thereby partially addressing an open problem posed in [Imp99].
2. In the random oracle model, we show that there exists incompressible functions as defined in [DLN96] where any substantial compression of the output of the function must reveal something about the seed.

1 Introduction

A systematic study of data compression from a computational stand-point was initiated in [GS91] and extended more recently in [TVZ03]. In both papers and our present work, the focus is on compression that can be achieved using efficient[1] compression and decompression algorithms. With unbounded computational power, we know from information theory [CT91] that the entropy $H(X)$ of a source X is both an upper and lower bound on the size of the compression (to within an additive $\log n$ term). If we are limited to efficient algorithms, then a source of pseudoentropy k cannot be compressed to length $k - \omega(\log n)$, where a source is said to have pseudoentropy at least k if it is computationally indistinguishable from some distribution having entropy at least k.

Is pseudoentropy indeed the right lower bound on the size of the compression for samplable sources? This was posed by Impagliazzo as an open problem [Imp99] in a talk (the case of general sources was partially addressed in [GS91]). Our work in this paper was motivated by this problem, and a key technical contribution of this paper is that the answer is no in an oracle setting. More specifically, we exhibit an oracle under which there samplable distributions over $\{0,1\}^n$ of very low entropy and pseudoentropy (say $n/2$) that cannot be compressed to less than $n - \omega(\log n)$ bits.

1.1 Previous Work

We know[2] that the output of a pseudorandom generator in $\{0,1\}^n$ cannot be efficiently compressed to length less than $n - O(1)$; otherwise, the compression and decompression algorithms will constitute a distinguisher

[1]The notion of efficiency in [GS91] and [TVZ03] are algorithms that can be implemented using probabilistic polynomial-time machines, whereas in this paper, we are concerned with polynomial size circuits. As we are concerned with lower bounds, this yields stronger results.

[2]In [GS91], this observation is attributed to L. Levin.

ISSN 1433-8092

from the uniform distribution over $\{0,1\}^n$. [GS91, TVZ03] explicitly impose the constraint that the source is not pseudorandom by requiring that there is an efficient membership test for its support, and in this setting, present a universal compression algorithm that compresses any source with entropy at most $n - O(\log n)$ to a source with expected length $n - \theta(\log n)$.

Goldberg and Sipser [GS91] present an oracle relative to which the $n-\theta(\log n)$ bound cannot be improved. Under this oracle, there exists a source over $\{0,1\}^n$ with an efficient membership test and entropy much less than $n/2$ but cannot be compressed by any probabilistic polynomial time machine with more than $O(\log n)$ savings in length. The source they constructed comprises exactly one string s_n of length n for each n, namely that which is Kolmogorov random, and the oracle is a membership test for such strings.

Dwork, Lotspiech and Naor [DLN96] present another approach to bypassing the impossibility of compressing pseudorandom sources which is motivated by a cryptographic application. There, the compression algorithm is given the seed to the pseudorandom output but is required to output a compressed string that reveals no information about the seed. The Blum-Micali-Yao generator based on a one-way permutation g is not incompressible with this definition, because the output of the generator can be efficiently recovered from $g(s)$ and its hardcore bit, and yet this information does not reveal the seed s.

1.2 Our Contributions

1.2.1 Separating Pseudoentropy and Compressibility

We prove a stronger version of the lower bound in Goldberg and Sipser [GS91] by presenting an oracle relative to which there exists samplable sources over $\{0,1\}^n$ of low pseudoentropy that cannot be compressed to length less than $n - \omega(\log n)$ by polynomial size circuits. Note that the source used in [GS91] is not samplable (otherwise, we can compress each string s of length n to n and decompress using the sampling algorithm), and can be optimally compressed by non-uniform circuits. Furthermore, our bounds hold for an average-case setting rather than a worst-case setting, and we also provide a separation result for a meaningful range of pseudoentropy. It also follows from our results that there is no black-box reduction from compression and decompression algorithms (with output length close to pseudoentropy) to sampling and membership tests.

1.2.2 An Incompressible Function

We apply the same techniques to prove the existence of incompressible functions as defined in [DLN96] in the random oracle model. An incompressible function is one in which any substantial compression of its output must reveal something about its input.

2 Preliminaries

We begin by reviewing some definitions and observations, most of which have previously appeared in [CT91, GS91, TVZ03].

2.1 Basic definitions

Definition 2.1 *A distribution X on $\{0,1\}^n$ has* s-pseudoentropy k *if there is a distribution D on $\{0,1\}^n$ of entropy*[3] *k such that every circuit of size s distinguishes X from D with* $\mathrm{neg}(s)$. *We say X on $\{0,1\}^n$ has* pseudoentropy k *if X has s-pseudoentropy at least k for all $s = poly(n)$.*

It is clear that entropy is a lower bound for s-pseudoentropy.

[3] *min-entropy is more commonly used in the definition of pseudoentropy. However, we are interested in proving an upper bound on the pseudoentropy in this setting, so using Shannon entropy (which is always at least the min-entropy) in the definition constitutes a stronger result.*

Lemma 2.2 *Let X_n be a flat source with membership oracle and $H(X_n) \leq k$. Then, X_n has s-pseudoentropy at most $k + \mathrm{neg}(s)$, for $s = \Omega(n)$.*

Proof: Consider the test A that accepts an input $x \in \{0,1\}^n$ iff the membership oracle accepts x. Note that A accepts X_n with probability 1. Let D be a source on $\{0,1\}^n$ of maximum entropy such that no polynomial size circuit can distinguish X_n from D with non-negligible advantage (in s). In particular, A distinguishes X_n from D with advantage at most $\epsilon = \mathrm{neg}(s)$. Then, at least $1 - \epsilon$ fraction of $\mathrm{Sup}(D)$ lies in $\mathrm{Sup}(X_n)$. Hence,

$$H(D) \leq (1-\epsilon)k + \epsilon \log(\frac{2^n}{\epsilon}) = k + \epsilon(n-k) + \epsilon \log(\frac{1}{\epsilon}) = k + \mathrm{neg}(s)$$

It follows that X_n has pseudoentropy at most $k + \mathrm{neg}(s)$. ■

2.2 Basics of Compression

Definition 2.3 *[TVZ03] For functions[4] $\mathsf{Enc} : \Sigma^* \to \Sigma^*$ and $\mathsf{Dec} : \Sigma^* \to \Sigma^*$, we say $(\mathsf{Enc}, \mathsf{Dec})$ compresses source X to length m if*

1. *For all $x \in \mathrm{Sup}(X)$, $\mathsf{Dec}(\mathsf{Enc}(x)) = x$, and*
2. *$\mathrm{E}[\,|\mathsf{Enc}(X)|\,] \leq m$.*

Definition 2.4 *[TVZ03] We say source X is* compressible *to length m if there exists functions Enc and Dec such that $(\mathsf{Enc}, \mathsf{Dec})$ compresses X to length m.*

Proposition 2.5 *[TVZ03] A source X_n is not compressible to length less than $H(X_n) - \lceil \log(n+2) \rceil$.*

For efficient compression and decompression algorithms, we have the following upper bound that can be achieved either using arithmetic coding [GS91] or condensers [TVZ03]:

Proposition 2.6 *[GS91, TVZ03] Any source over $\{0,1\}^n$ with membership oracle can be compressed to length $n - \theta(\log n)$ in polynomial time if $H(X_n) < n - (3+\delta)\log n$.*

The following lemma establishes (in some sense) pseudoentropy as the computational analogue of entropy as a measure of compressibility.

Lemma 2.7 *Let X_n be a source with $2s$-pseudoentropy k. Then, X_n cannot be compressed to length less than $k - \lceil \log(n+2) \rceil - O(1)$ by any circuits $(\mathsf{Enc}, \mathsf{Dec})$ of size s.*

Proof: (sketch) Let D be a source of entropy k such that X_n is computationally indistinguishable from D, and suppose on the contrary that we have circuits $(\mathsf{Enc}, \mathsf{Dec})$ of size s that compresses X_n to length less than $k - \lceil \log(n+2) \rceil - O(1)$. It follows from the computational indistinguishability property that $\Pr[\mathsf{Dec}(\mathsf{Enc}(D)) = D] \geq \Pr[\mathsf{Dec}(\mathsf{Enc}(X_n)) = X_n] - \mathrm{neg}(s)$, and that $\mathrm{E}[\,|\mathsf{Enc}(D)|\,] \leq \mathrm{E}[\,|\mathsf{Enc}(X_n)|\,] + \mathrm{neg}(s)$. From Prop 2.9 below, we may modify $(\mathsf{Enc}, \mathsf{Dec})$ to yield circuits of size $2s + O(1)$ that compress D to length $k - \lceil \log(n+2) \rceil - O(1)$, a contradiction to Prop 2.5. ■

[4] *The functions Enc and Dec are also referred to as the encoding and decoding functions respectively, hence the choice of notation.*

2.2.1 Average-Case Results (from compression somewhere to compression everywhere)

Consider the following weaker definition of compressibility, where we only require that we obtain short outputs only on some fraction of the input:

Definition 2.8 *For functions* $\mathsf{Enc} : \Sigma^* \to \Sigma^*$ *and* $\mathsf{Dec} : \Sigma^* \to \Sigma^*$*, we say* $(\mathsf{Enc}, \mathsf{Dec})$ *α-somewhere compresses source X to length m if there exists $W \subseteq \mathrm{Sup}(X)$ of density at least α (that is, $\Pr_{x \leftarrow X}[x \in W] \geq \alpha$) satisfying*

1. *For all $x \in W$, $\mathsf{Dec}(\mathsf{Enc}(x)) = x$, and*
2. *$\mathrm{E}[|\mathsf{Enc}(X |_W)|] \leq m$, where $X |_W$ is the distribution on strings x drawn according to X conditioned upon $x \in W$.*

*Furthermore, we say source X is α-*somewhere compressible *to length m if there exists functions* Enc *and* Dec *such that* $(\mathsf{Enc}, \mathsf{Dec})$ *α-somewhere compresses X to length m.*

Proposition 2.9 *Let X be a source over $\{0,1\}^n$, and suppose* $(\mathsf{Enc}, \mathsf{Dec})$ *α-somewhere compresses source X to length m. Then, X is compressible to length $m' = \alpha m + (1-\alpha)n + 1$ by functions of similar computational complexity to* $(\mathsf{Enc}, \mathsf{Dec})$.

Proof: Consider $(\mathsf{Enc}', \mathsf{Dec}')$ given by:

$$\mathsf{Enc}'(x) = \begin{cases} 0\mathsf{Enc}(x) & \text{if } |\mathsf{Enc}(x)| < n \text{ and } \mathsf{Dec}(\mathsf{Enc}(x)) = x \\ 1x & \text{otherwise} \end{cases}$$

The result follows readily. Note that the transformation does not require an explicit specification of W. ■

In particular, if there exists circuits $(\mathsf{Enc}, \mathsf{Dec})$ of size s that α-compresses X to length $n - \omega(\log n)$ for some constant $0 < \alpha < 1$, then X is compressible to length $n - \omega(\log n)$ by circuits of size $2s + O(1)$.

2.2.2 Non-Uniform Compression

Consider $(\mathsf{Enc}, \mathsf{Dec})$ functions that may be implemented by a family of circuits $\{(\mathsf{Enc}_n, \mathsf{Dec}_n)\}$. In order to have a meaningful definition of compression in the non-uniform setting, we make use of the observation (made in [TVZ03]) that for a source X_n with support in $\{0,1\}^n$, we may assume and also stipulate that $|\mathsf{Enc}_n(x)| \leq n+1$ for all $x \in \mathrm{Sup}(X_n)$. Therefore, Dec_n may be seen as taking inputs of length $\lceil \log(n+1) \rceil + n + 1$, where the first $\lceil \log(n+1) \rceil$ bits (prefixed with zeroes) specify the length of the "actual" input x.

Proposition 2.10 *(Levin) If non-uniform one-way functions exist, then there exist polynomial-time samplable sources X_n of entropy at most n^ϵ that cannot be compressed to length $n - O(1)$ by any polynomial size circuits* $(\mathsf{Enc}, \mathsf{Dec})$.

Proof: (sketch) If non-uniform one-way functions exist, then there exists a pseudorandom generator $G : \{0,1\}^{n^\epsilon} \to \{0,1\}^n$ that is secure against polynomial size circuits. Take $X_n = G(U_{n^\epsilon})$. ■

3 An Incompressible Samplable Source with Low Pseudoentropy

Let us fix k, n and study for which d there exists flat sources in $\{0,1\}^n$ of entropy k that is not compressible by circuits of size s to length $n - d$. We may think of k, d, s as functions of n. In addition, let $N = 2^n, K = 2^k, D = 2^d$.

Let $\mathcal{F}$ be the set of injective functions $f : \{0,1\}^k \to \{0,1\}^n$. For each such $f \in \mathcal{F}$, we have a flat source $f(U_k)$ in $\{0,1\}^n$, and we define an sampling oracle $\mathcal{O}_f^S$, a membership oracle $\mathcal{O}_f^M$ and an oracle $\mathcal{O}_f$ that combines both sampling and membership functionalities as follows:

$$\begin{aligned} \mathcal{O}_f^S(x) &= f(x) \\ \mathcal{O}_f^M(b,x) &= \begin{cases} 1 & x \in f(\{0,1\}^n) \\ 0 & x \notin f(\{0,1\}^n) \end{cases} \\ \mathcal{O}_f(b,x) &= \begin{cases} \mathcal{O}_f^S(x) & \text{if } b = 0 \\ \mathcal{O}_f^M(x) & \text{if } b = 1 \end{cases} \end{aligned}$$

In the rest of the section, whenever we refer to oracle circuits (and in particular oracle circuits $(\mathsf{Enc}, \mathsf{Dec})$ for some source $f(U_k))$, we always mean oracle access to $\mathcal{O}_f$, where the specific function f will be clear from context.

3.1 Main Result

Theorem 3.1 *For any k satisfying $6 \log s + O(1) < k < n$, there exists (injective) functions $f : \{0,1\}^k \to \{0,1\}^n$ such that given oracle access to $\mathcal{O}_f$,*

1. *$f(U_k)$ is samplable and has entropy k and s-pseudoentropy $k + \text{neg}(n)$.*
2. *$f(U_k)$ cannot be compressed to length less than $n - 2\log s - \log n - O(1)$ by oracle circuits of size s. In addition, $f(U_k)$ cannot be α-somewhere compressed to length less than $n - \theta(\log s)$ by oracle circuits of size s, for any constant $0 < \alpha < 1$.*

The following corollary follows readily:

Corollary 3.2 *For any k satisfying $\omega(\log n) < k < n$, there exists an oracle relative to which there exists a samplable source X with entropy k and pseudoentropy $k + \text{neg}(n)$, but cannot be compressed to length $n - \omega(\log n)$ by polynomial size circuits.*

3.1.1 A Remark on Optimality of the Compressibility Lower Bound

Note that this lower bound in Corollary 3.2 matches the upper bound in Prop 2.6.

[Tre02] also pointed out a simpler construction of compression and decompression functions that compresses the source $f(U_k)$ to length $n - \log s$ for $k < n/2 - \log s/2$, which matches the lower bound in Theorem 3.1 up to constant multiples of $\log s)$. Take a randomly chosen universal hash function $h : \{0,1\}^n \to \{0,1\}^{n-\log s}$, which can be shown injective on $f(\{0,1\}^k)$ with constant probability. Fix one such function h_n, and encode $x \in \{0,1\}^n$ as $h(x)$ and decode y by enumerating over the pre-images of $h^{-1}(y)$ and taking the unique pre-image that is accepted by the membership oracle. This yields compression and decompression circuits of size $O(ns)$ that compresses $f(U_k)$ to length $n - \log s$.

3.2 Main Idea

Let $\mathsf{compressible}$ be the set of functions $f \in \mathcal{F}$ for which there exists oracle circuits $(\mathsf{Enc}, \mathsf{Dec})$ of size s such that given oracle access to $\mathcal{O}_f$ compresses $f(U_k)$ to length $n - d$. For each such f and the corresponding $(\mathsf{Enc}, \mathsf{Dec})$ circuits, we define

$$\begin{aligned} \mathsf{invert}_f &= \{x \mid \text{on input } \mathsf{Enc}(f(x)),\ \mathsf{Dec} \text{ queries } \mathcal{O}_f^S \text{ on } x\} \\ \mathsf{forge}_f &= \{x \mid \text{on input } \mathsf{Enc}(f(x)),\ \mathsf{Dec} \text{ does not query } \mathcal{O}_f^S \text{ on } x\} \end{aligned}$$

Clearly, invert_f and forge_f form a partition of $\{0,1\}^k$. In addition, we define

$$\begin{aligned}\mathsf{invertible} &= \{f \in \mathsf{compressible} : |\mathsf{invert}_f| > \frac{1}{n}2^k\} \\ \mathsf{forgeable} &= \{f \in \mathsf{compressible} : |\mathsf{forge}_f| \geq (1-\frac{1}{n})2^k\}\end{aligned}$$

Observe that $\mathsf{compressible} = \mathsf{invertible} \cup \mathsf{forgeable}$[5]. For functions f in $\mathsf{invertible}$, there exists small circuits that invert f on invert_f by running Enc and then monitoring the oracle queries that Dec makes to $\mathcal{O}_f^S$. Hence, $\mathsf{invertible}$ is small because a random function is non-uniformly one-way with high probability [GT00]. Similarly, for functions f in $\mathsf{forgeable}$, the circuit Dec computes ("forges") f on $x \in \mathsf{forge}_f$ without querying $\mathcal{O}_f^S$ on x. This cannot happen too often unless $\mathsf{Enc}(f(x))$ is "long".

To formalize this intuition, we use techniques from [GT00] based on a reconstruction paradigm to prove upper bounds for $|\mathsf{invertible}|$ and $|\mathsf{forgeable}|$ by arguing that functions in $\mathsf{invertible}$ and $\mathsf{forgeable}$ have short descriptions. This yields the desired upper bound on $|\mathsf{compressible}|$, from which theorem 3.1 follows.

3.3 Proof of theorem 3.1

3.3.1 $\mathsf{invertible}$ is small

Lemma 3.3 *Take any $f \in \mathsf{invertible}$, and let $(\mathsf{Enc}, \mathsf{Dec})$ be oracle circuits of size s that compress $f(U_k)$ to length $n-d$, and also satisfy $|\mathsf{invert}_f| > \frac{1}{n}2^k$. Then, there exists an oracle circuit $\mathcal{A}$ of size $s' = 2s + sn$ such that*

$$\Pr_{x \in U_k}[\mathcal{A}^{\mathcal{O}_f}(f(x)) = x] > \frac{1}{n}$$

Proof: Consider the following circuit $\mathcal{A}$ that on input $y \in \{0,1\}^n$:

1. Compute $z = \mathsf{Enc}(y)$.
2. Simulate Dec on input z and monitor the queries Dec makes to $\mathcal{O}_f^S$. When the simulation is completed with output $\mathsf{Dec}(z)$, output the query x to $\mathcal{O}_f^S$ where the answer is $\mathsf{Dec}(z)$. If there is no such query, output 0.

It is easy to see that for all $x \in \mathsf{invert}_f$, $\mathcal{A}(f(x)) = x$, from which the result follows. ■

Lemma 3.4 *Take any $f \in \mathsf{invertible}$, and let $\mathcal{A}$ be the circuit constructed in lemma 3.3. Then, f can be described using*

$$\log\binom{N}{b} + \log\binom{K}{b} + \log\left(\binom{N-b}{K-b}(K-b)!\right)$$

bits, given $\mathcal{A}$, where $b = \frac{K}{s'n}$.

Proof: Recall that for all $x \in \mathsf{invert}_f$, $\mathcal{A}(f(x)) = x$. WLOG, assume that for all such x, $\mathcal{A}$ makes distinct queries to $\mathcal{O}_f^S$, and always queries $\mathcal{O}_f^S$ on x before it outputs x. We claim that there exists a subset T of $f(\mathsf{invert}_f)$ of size b, such that we can describe f given:

$$f^{-1}(T), T, f\mid_{\{0,1\}^k - f^{-1}(T)}$$

[5]Note that this is not a partition; it could be the case that for a fixed f can be compressed with two different pairs of circuits, and in one case, it falls into $\mathsf{invertible}$ and the other into $\mathsf{forgeable}$.

Greedy-Construct-T

1. Initially, T is empty and all elements of $f(\mathsf{invert}_f)$ are candidates for being an element of T. Remove the lexicographically smallest element $y = f(x)$ in $f(\mathsf{invert}_f)$, and put y in T.

2. Simulate $\mathcal{A}$ on y, and halt the simulation immediately after $\mathcal{A}$ queries O_f^S on x. Let $x_1, \dots, x_q$ be the queries $\mathcal{A}$ makes to $\mathcal{O}_f^S$, where $x_q = x$ and $q \le s'$.

3. Remove $f(x_1), \dots, f(x_{q-1})$ from $f(\mathsf{invert}_f)$ (note that some of these values may have already been removed in previous iterations). Then, all the elements $x_1, \dots, x_{q-1}$ that were not already added to T will never be added to T.

4. Remove the lexicographically smallest of the remaining elements in $f(\mathsf{invert}_f)$, say $y = f(x)$, put y in T, and return to step (2).

Recover-F

1. To reconstruct the values of f^{-1} on T, start with a look-up table for f on values in $\{0,1\}^k - W$, and go through the strings in T in lexicographic order.

2. Pick the lexicographically smallest element y from T and simulate $\mathcal{A}$ on y. Halt immediately after $\mathcal{A}$ makes a query x to $\mathcal{O}_f^S$, for which the answer is not in the look-up table for f.

3. We are given T and $f\mid_{\{0,1\}^k - f^{-1}(T)}$, so we know all of $f(\{0,1\}^k)$ and can therefore answer all queries to O_f^M.

4. Consider any query x' $\mathcal{A}$ makes to $\mathcal{O}_f^S$ that precedes the last query x. By construction, either $x' \notin f^{-1}(T)$, or $f(x')$ precedes $f(x)$ lexicographically in T (in this case, we will have added $(x', f(x'))$ to the look-up table in a previous iteration, as done in step (5)). In either case, the look-up table has the answer, so we can simply retrieve the answer.

5. Once the simulation halts, we know the value $x = f^{-1}(y)$. Add (x, y) to the look-up table for f.

6. Remove y from T, and return to step (2), choosing the lexicographically smallest of the remaining elements in T.

In each step of Greedy-Construct-T, we add one element to T and remove at most s' elements from $f(\mathsf{invert}_f)$. Since $f(\mathsf{invert}_f)$ has initially K/n elements, in the end T has at least $K/s'n$ elements. ■

Lemma 3.5 *If $k > 6 \log s + O(1)$,*

$$|\mathsf{invertible}| < 2^{-(s+1)} \binom{N}{K} K!$$

Proof: We can describe an oracle circuit of size s' using $s'(n+k)\log s'$ bits, so any function $f \in \mathsf{invertible}$ can be described using

$$\log \binom{N}{b} + \log \binom{K}{b} + \log \left(\binom{N-b}{K-b} (K-b)! \right) + s'(n+k) \log s'$$

bits. It follows that

$$\begin{aligned}\frac{|\mathsf{invertible}|}{\binom{N}{K}K!} &\le \frac{\binom{N}{b}\binom{K}{b}\binom{N-b}{K-b}(K-b)!2^{s'(n+k)\log s'}}{\binom{N}{K}\cdot K!} \\ &= \frac{\binom{K}{b}}{b!}\cdot 2^{s'(n+k)\log s'} < \left(\frac{e^2K}{b^2}\right)^b \cdot 2^{K/s^2} \\ &\le \left(\frac{2e^2s^4}{K}\right)^{K/s^2} < 2^{-(s+1)}\end{aligned}$$

■

3.3.2 forgeable is small

Lemma 3.6 *Take any $f \in$ forgeable, and let $(\mathsf{Enc}, \mathsf{Dec})$ be oracle circuits of size s that compress $f(U_k)$ to length $n-d$, and also satisfy $|\mathsf{forge}_f| \ge (1-\frac{1}{n})2^k$. Then, f can be described using*

$$a\left(\frac{n}{n-1}\cdot(n-d)+\log n\right)+\log\binom{K}{a}+\log\left(\binom{N-a}{K-a}(K-a)!\right)+a\log s$$

bits, given Dec, *where $a = (1-\frac{1}{n})K/s$.*

Proof: WLOG, assume that Dec makes distinct queries to O_f^S and distinct queries to $\mathcal{O}_f^M$. Now, recall that for all $x \in \mathsf{forge}_f$, Dec on input $\mathsf{Enc}(f(x))$ never queries $\mathcal{O}_f^S$ on x. We may also assume that for all such x, Dec on input $\mathsf{Enc}(f(x))$ always queries $\mathcal{O}_f^M$ on $f(x)$ before it outputs $f(x)$. Note that $f(x)$ may not necessarily be the last query Dec makes to $\mathcal{O}_f^M$.

We claim that there exists a subset W of forge_f of size a, such that we can describe f given:

$$\mathsf{Enc}(f(W)), W, f\mid_{\{0,1\}^k - W}$$

in addition to $\{a_z \in [s] \mid z \in \mathsf{Enc}(f(W))\}$ of membership advice strings and where $\mathsf{Enc}(f(W))$ is represented as an ordered set where the ordering is that induced by the lexicographic ordering on W. Furthermore, W satisfies

$$\mathrm{E}_{x\in W}[\,|\mathsf{Enc}(f(x))|\,] \le \frac{n}{n-1}\cdot(n-d)$$

Therefore, we can describe the ordered set $\mathsf{Enc}(f(W))$ using $a\left(\frac{n}{n-1}\cdot(n-d)+\log n+1\right)$ by concatenating the values of $|\mathsf{Enc}(f(w))|\mathsf{Enc}(f(w))$ as w runs through W in lexicographic ordering. Note that we should write $|\mathsf{Enc}(f(w))| \in \{0,1\}^{\lceil\log(n+2)\rceil}$ in binary with leading 0's, so that $|\mathsf{Enc}(f(w))|\mathsf{Enc}(f(w))$ yields a prefix-free encoding of $f(w)$.

GREEDY-CONSTRUCT-W

1. Initially, W is empty and all elements of forge_f are candidates for being an element of W. Remove the lexicographically smallest[6] element $z = \mathsf{Enc}(f(x))$ in $\mathsf{Enc}(f(\mathsf{forge}_f))$, and put x in W.

2. Simulate Dec on z, and halt the simulation immediately after Dec queries $\mathcal{O}_f^M$ on $f(x)$. Let $x_1, \ldots, x_q$ be the queries Dec makes to $\mathcal{O}_f^S$; and let $y_1' = f(x_1'), \ldots, y_m' = f(x_m')$ and $y_1, \ldots, y_r$ be the queries Dec makes to $\mathcal{O}_f^M$, where $\mathcal{O}_f^M$ answers yes to $y_1', \ldots, y_m'$, and no to $y_1, \ldots, y_r$. In particular, $x = x_m'$ and $f(x) = y_m'$. Set $a_z = m + r$, that is, the a_z-th query that Dec makes to $\mathcal{O}_f^M$ is $f(x)$.

[6]In the lexicographic ordering on $\mathsf{Enc}(f(\{0,1\}^k))$, shorter strings always have precedence over longer strings.

3. Remove $\mathsf{Enc}(f(x_1)), \ldots, \mathsf{Enc}(f(x_q))$ and $\mathsf{Enc}(f(x'_1)), \ldots, \mathsf{Enc}(f(x'_{m-1}))$ from $\mathsf{Enc}(f(\mathsf{forge}_f))$ (note that some of these values may have already been removed in previous iterations). Then, all the elements $x_1, \ldots, x_q$ and $x'_1, \ldots, x'_{m-1}$ that were not already added to W will never be added to W.

4. Remove the lexicographically smallest of the remaining elements in $\mathsf{Enc}(f(\mathsf{forge}_f))$, say $z = \mathsf{Enc}(f(x))$, put x in W, and return to step (2).

RECOVER-F

1. To reconstruct the values of f on W, start with a look-up table for f on values in $\{0,1\}^k - W$, and go through the strings in $\mathsf{Enc}(f(W))$ in lexicographic order.

2. Pick the lexicographically smallest element $z = \mathsf{Enc}(f(x))$ from $\mathsf{Enc}(f(W))$ and simulate Dec on z. Halt immediately after Dec makes the a_z-th query to $\mathcal{O}_f^M$, which by construction is the value $f(x)$.

3. By construction, whenever Dec makes a query x' to $\mathcal{O}_f^S$, either $x' \notin W$, or $\mathsf{Enc}(f(x'))$ precedes z lexicographically in $\mathsf{Enc}(f(W))$ (in this case, we will have added $(x', f(x'))$ to the look-up table in a previous iteration, as done in step (5)). In either case, the look-up table has the answer, so we can simply retrieve the answer.

4. Consider any of the first $a_z - 1$ queries that Dec makes to $\mathcal{O}_f^M$, say y'. If $y' \in f(\{0,1\}^k)$, say $y' = f(x')$, then by construction, either $x' \notin W$, or $\mathsf{Enc}(f(x'))$ precedes z lexicographically in $\mathsf{Enc}(f(W))$. In either case, the look-up table has the entry (x', y'). If $y \notin f(\{0,1\}^k)$, we will not find y' in the look-up table. Therefore, we can answer the query in the simulation by responding with a "yes" if y' in the look-up table, and "no" otherwise.

5. Once the simulation halts, we know the value $f(x)$. In addition, we can use the ordering on $\mathsf{Enc}(f(W))$ to figure out x, and add $(x, f(x))$ to the look-up for f. More specifically, if $f(x)$ is the ith element of $\mathsf{Enc}(f(W))$ (in the induced ordering), then x is the ith element if W (in lexicographic ordering).

6. Remove z from $\mathsf{Enc}(f(W))$, and return to step (2), choosing the lexicographically smallest of the remaining elements in $\mathsf{Enc}(f(W))$.

In each step of GREEDY-CONSTRUCT-W, we add one element z to W and remove at most s elements from $\mathsf{Enc}(f(\mathsf{forge}_f))$. Since $\mathsf{Enc}(f(\mathsf{forge}_f))$ has initially $(1 - \frac{1}{n})K$ elements, in the end W has at least $(1 - \frac{1}{n})K/s$ elements. Note that the s elements we remove succeed (or equal) z in lexicographic order, and must have length at least $|z|$. It follows

$$\mathrm{E}_{x \in W}[\,|\mathsf{Enc}(f(x))|\,] \leq \mathrm{E}_{x \in \mathsf{forge}_f}[\,|\mathsf{Enc}(f(x))|\,] \leq \frac{n}{n-1} \cdot (n-d)$$

■

Lemma 3.7 *If $k > 6 \log s + O(1)$ and $d > 2 \log s + \log n + O(1)$,*

$$|\mathsf{forgeable}| < 2^{-(s+1)} \binom{N}{K} K!$$

Proof: Again, we can describe Dec using $s(n+k)\log s$ bits, so any function $f \in \mathsf{forgeable}$ can be described using

$$a\left(\frac{n}{n-1} \cdot (n-d) + \log n\right) + \log \binom{K}{a} + \log\left(\binom{N-a}{K-a}(K-a)!\right) + a \log s + s(n+k)\log s$$

bits. It follows that

$$\begin{aligned}
\frac{|\mathsf{forgeable}|}{\binom{N}{K}K!} &\leq \frac{(Nn/D)^a (N/D)^{a/(n-1)} \binom{K}{a}\binom{N-a}{K-a}(K-a)! s^a 2^{s(n+k)\log s}}{\binom{N}{K}\cdot K!} \\
&< \frac{(4Nn/D)^a \binom{K}{a} s^a}{\binom{N}{a} a!} \cdot 2^{s(n+k)\log s} \\
&< \frac{\left(\frac{4Nn}{D}\right)^a \left(\frac{Ke}{a}\right)^a s^a}{\left(\frac{N}{a}\right)^a \left(\frac{a}{e}\right)^a} \cdot 2^{s(n+k)\log s} = \left(\frac{4Ke^2 ns}{Da}\right)^a \cdot 2^{s(n+k)\log s} \\
&< \left(\frac{8e^2 s^2 n}{D}\right)^{K/2s} \cdot 2^{K/2s} < 2^{-(s+1)}
\end{aligned}$$

■

3.3.3 Rest of the proof

From lemmas 3.5 and 3.7, we have (for the parameters stated in the theorem)

$$|\mathsf{compressible}| < 2^{-s}\binom{N}{K}K!$$

The result follows readily from this and lemma 2.2.

4 An Incompressible Function From Cryptography

4.1 On Incompressible Functions

4.1.1 Motivation

Dwork, Lotspiech and Naor [DLN96] defined an incompressible length-increasing function $f : \{0,1\}^k \to \{0,1\}^n$ (with $k = o(n)$) to be one where in order for one party Alice to communicate $f(x)$ to Bob in $o(|f(x)|)$ bits, Alice must reveal x, in the sense that Bob can effectively compute x from the message Alice transmits. This definition is motivated by its application to Digital Signets, a scheme for protecting digital content from illegal redistribution by an authorized user. Here, some digital content is distributed in an encrypted form, say using a one-time pad with the string $f(x)$, which has length comparable to that of the content, where the seed x to f is typically some sensitive piece of information about the authorized user, such as her credit card number. Typically, the content being distributed requires large bandwidth for redistribution and therefore it is infeasible to redistribute $f(x)$ uncompressed. On the other hand, the user will not want to reveal x either.

4.1.2 A Formal Definition

Definition 4.1 *Given a length-increasing function* $f : \{0,1\}^k \to \{0,1\}^n$, *and functions* $\mathsf{Enc} : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^*$ *and* $\mathsf{Dec} : \{0,1\}^* \to \{0,1\}^n$, *we say* $(\mathsf{Enc}, \mathsf{Dec})$ *compresses* f *to length* m *if*

1. *For all* $x \in \{0,1\}^k$, $\mathsf{Dec}(\mathsf{Enc}(x, f(x))) = x$, *and*
2. $\mathrm{E}[|\mathsf{Enc}(U_k, f(U_k))|] \leq m$.

Furthermore, we say (Enc, Dec) securely compresses f *to length* m *if the following additional condition is satisfied: for all polynomial size circuits* A,

$$\Pr_{x\in\{0,1\}^k}[A(\mathsf{Enc}(x,f(x)))=x]=\mathrm{neg}(n)$$

Definition 4.2 *We say the function* $f:\{0,1\}^k\to\{0,1\}^n$ *is* incompressible *if there exists no polynomial size circuits* (Enc, Dec) *that securely compresses* f *to length* $o(n)$.

The key distinction between the notion of compressibility here and that in section 2.2 is that Enc has access to a short description of $f(x)$ here, namely that of x. Therefore, $\mathsf{Enc}(x,f(x))=x$ and $\mathsf{Dec}(x)=x$ efficiently compress f to length k; however, it does not securely compress f. In addition, even if we take f to be some pseudorandom generator secure against polynomial size circuits, $(U_k,f(U_k))$ is not pseudorandom against polynomial size circuits, so it is no longer necessarily the case that f cannot be efficiently compressed to length $n-O(1)$.

4.2 Existence of Incompressible Functions in the Random Oracle Model

Theorem 4.3 *Let* $\mathcal{O}$ *be a random oracle that maps* $\{0,1\}^k$ *to* $\{0,1\}^k$. *Then, for every integer* $c>1$ *and* $n=k^c$, *the function* $f:\{0,1\}^k\to\{0,1\}^n$:

$$f:x\mapsto\mathcal{O}(x+1)\circ\mathcal{O}(x+2)\circ\cdots\circ\mathcal{O}(x+n/k)$$

is incompressible (per definition 4.2) with probability $1-\mathrm{neg}(k)$ *(over the choices made by the random oracle).*

Let $\mathcal{H}$ be the set of functions $h:\{0,1\}^k\mapsto\{0,1\}^k$, so we may view $\mathcal{O}$ as being a uniformly chosen element of $\mathcal{H}$. We write $\mathcal{O}_h$ when $\mathcal{O}$ is chosen to be $h\in\mathcal{H}$, and we define

$$f_h:x\mapsto h(x+1)\circ h(x+2)\circ\cdots\circ h(x+\ell)$$

where $\ell=n/k$. In addition, for each $h\in\mathcal{H}$, and each (Enc, Dec) that compresses h_f, we define

$$\begin{aligned}\mathsf{invert}_{h,(\mathsf{Enc},\mathsf{Dec})}&=\{x\mid\text{on input }\mathsf{Enc}(x,f_h(x)),\ \mathsf{Dec}\text{ queries }\mathcal{O}_h\text{ on at least one of }x+1,\ldots,x+\ell\}\\ \mathsf{forge}_{h,(\mathsf{Enc},\mathsf{Dec})}&=\{x\mid\text{on input }\mathsf{Enc}(x,f_h(x)),\ \mathsf{Dec}\text{ does not query }\mathcal{O}_h\text{ on any of }x+1,\ldots,x+\ell\}\end{aligned}$$

Clearly, $\mathsf{invert}_{h,(\mathsf{Enc},\mathsf{Dec})}$ and $\mathsf{forge}_{h,(\mathsf{Enc},\mathsf{Dec})}$ form a partition of $\{0,1\}^k$. The following lemma relates the cardinality of $\mathsf{invert}_{h,(\mathsf{Enc},\mathsf{Dec})}$ to whether (Enc, Dec) securely compresses h_f:

Lemma 4.4 *Suppose* (Enc, Dec) *compresses* h_f *to length* $n/2$ *and satisfy* $|\mathsf{invert}_{h,(\mathsf{Enc},\mathsf{Dec})}|>2^k/n$. *Then, there exists a circuit* $\mathcal{A}$ *of size poly*(s) *such that*

$$\Pr_{x\in\{0,1\}^k}[\mathcal{A}(\mathsf{Enc}(x,f_h(x)))=x]>\frac{1}{n}$$

In particular, if $s=poly(n)$, *then* (Enc, Dec) *does not securely compress* f_h *to length* $n/2$.

Proof: Consider the following circuit $\mathcal{A}$ that on input y: Simulate Dec on input y and and monitor the queries Dec makes to h. Suppose Dec outputs $w_1\circ\ldots\circ w_\ell$ and queries h at $q_1,\ldots,q_t$. For $i=1,2,\ldots,t$, if $h(q_i)=w_j$, then output q_i-j. Otherwise, output 0.

It is easy to see that for all $x\in\mathsf{invert}_f$, $\mathcal{A}(\mathsf{Enc}(x,f_h(x)))=x$, from which the result follows. ■

Now, consider $\mathsf{s-compressible}$, the set of functions $h\in\mathcal{H}$ for which there exists oracle circuits (Enc, Dec) of size s that compresses h_f to length $n/2$ and satisfy $|\mathsf{invert}_{h,(\mathsf{Enc},\mathsf{Dec})}|\le 2^k/n$. It follows from lemma 4.4

that $\mathsf{s-compressible}$ contains all functions $h \in \mathcal{H}$ for which there exists oracle circuits $(\mathsf{Enc}, \mathsf{Dec})$ of size s that securely compresses h_f to length $n/2$. Hence, to prove theorem 4.3, it suffices to establish a (strong) upper bound on $|\mathsf{s-compressible}|$.

Lemma 4.5 *Take any $h \in \mathsf{s-compressible}$, and let $(\mathsf{Enc}, \mathsf{Dec})$ be oracle circuits of size s that compress h_f to length $n/2$, and also satisfy $|\,\mathsf{invert}_{h,(\mathsf{Enc},\mathsf{Dec})}| \leq 2^k/n$. Then, h can be described using*

$$a\left(\frac{n}{n-1} \cdot \frac{n}{2} + \log n + 1\right) + \log \binom{K}{a} + (K - a\ell)k$$

bits, given Dec, where $a = \left(1 - \frac{1}{n}\right) \frac{K}{s+\ell}$.

Proof: The proof is very similar to that for lemma 3.6, possibly much simpler too. First, recall that for all $x \in \mathsf{forge}_{h,(\mathsf{Enc},\mathsf{Dec})}$, Dec on input $\mathsf{Enc}(x, f_h(x))$ never queries $\mathcal{O}_h$ on x. WLOG, assume that Dec makes distinct queries to h on all of these x.

We claim that there exists a subset W of $\mathsf{forge}_{h,(\mathsf{Enc},\mathsf{Dec})}$ of size a, such that we can describe h given:

$$\mathsf{Enc}(W, f_h(W)), W, h\,|_{\{0,1\}^k - \bigcup_{i=1}^{\ell}(W+i)}$$

where $\mathsf{Enc}(W, f_h(W)) = \{\mathsf{Enc}(w, f_h(w)) \mid w \in W\}$ is represented as an ordered set where the ordering is that induced by the lexicographic ordering on W. Moreover, W satisfies

$$\mathrm{E}_{x \in W}[\,|\mathsf{Enc}(x, f_h(x))|\,] \leq \frac{n}{n-1} \cdot \frac{n}{2}$$

Therefore, we can describe the ordered set $\mathsf{Enc}(f(W))$ using $a\left(\frac{n}{n-1} \cdot \frac{n}{2} + \log n + 1\right)$ by concatenating the values of $|\mathsf{Enc}(w, f(w))|\mathsf{Enc}(w, f(w))$ as w runs through W in lexicographic ordering.

GREEDY-CONSTRUCT-W

1. Initially, W is empty and all elements of $\mathsf{forge}_{h,(\mathsf{Enc},\mathsf{Dec})}$ are candidates for being an element of W. Remove the lexicographically smallest element $z = \mathsf{Enc}(x, f_h(x))$ in $\mathsf{Enc}(W, f_h(W))$, and put x in W.

2. Simulate Dec on z, and halt the simulation when Dec is done and outputs $y_1 \ldots y_\ell$. Let $x_1, \ldots, x_q$ be the queries Dec makes to f. We could then use the ordering on $\mathsf{Enc}(W, f_h(W))$ to find out what x is.

3. Remove $\mathsf{Enc}(x_1, f_h(x_1)), \ldots, \mathsf{Enc}(x_q, f_h(x_q))$ and $\mathsf{Enc}(x+1, f_h(x+1)), \ldots, \mathsf{Enc}(x+\ell, f_h(x+\ell))$ from $\mathsf{Enc}(W, f_h(W))$. Then, all the elements $x_1, \ldots, x_q$ and $x+1, \ldots, x+\ell$ that were not already added to W will never be added to W.

4. Remove the lexicographically smallest of the remaining elements in $\mathsf{Enc}(W, f_h(W))$, say $z = \mathsf{Enc}(x, f_h(x))$, put x in W, and return to step (2).

RECOVER-F

1. To reconstruct the values of f on W, start with a look-up table for f on values in $\{0,1\}^k - W'$, and go through the strings in $\mathsf{Enc}(W, f_h(W))$ in lexicographic order.

2. Pick the lexicographically smallest element $z = \mathsf{Enc}(x, f_h(x))$ from $\mathsf{Enc}(W, f_h(W))$ and simulate Dec on z. Halt the simulation when Dec is done and outputs $y_1 \ldots y_\ell$.

3. By construction, whenever Dec makes a query x' to f, we can retrieve the answer from the look-up table.

4. Once the simulation halts, we know $f(x+1), \ldots, f(x+\ell)$ (given by $y_1, \ldots, y_\ell$ respectively). In addition, we can use the ordering $\mathsf{Enc}(W, f_h(W))$ to figure out x. We can then add $(x_1, y_1), \ldots, (x_\ell, y_\ell)$ to the look-up for f.

5. Remove z from $\mathsf{Enc}(W, f_h(W))$, and return to step (2), choosing the lexicographically smallest of the remaining elements in $\mathsf{Enc}(W, f_h(W))$.

In each step of GREEDY-CONSTRUCT-W, we add one element to W and remove at most $s+\ell$ elements from $\mathsf{Enc}(\mathsf{forge}_{h,(\mathsf{Enc},\mathsf{Dec})}, f_h(\mathsf{forge}_{h,(\mathsf{Enc},\mathsf{Dec})}))$. Since $\mathsf{Enc}(\mathsf{forge}_{h,(\mathsf{Enc},\mathsf{Dec})}, f_h(\mathsf{forge}_{h,(\mathsf{Enc},\mathsf{Dec})}))$ has initially $\left(1 - \frac{1}{n}\right) K$ elements, in the end W has at least $\left(1 - \frac{1}{n}\right) \frac{K}{s+\ell}$ elements. ■

Lemma 4.6

$$|\mathsf{s-compressible}| < 2^{-s} K^K$$

Proof: Again, we can describe Dec using $s(n+k)\log s$ bits, so any function $h \in \mathsf{s-compressible}$ can be described using

$$a\left(\frac{n}{n-1} \cdot \frac{n}{2} + \log n + 1\right) + \log \binom{K}{a} + (K - a\ell)k + s(n+k)\log s$$

bits. It follows that

$$\begin{aligned}
\frac{|\mathsf{s-compressible}|}{K^K} &< \left(\frac{2nN^{1/2}N^{1/2(n-1)}Ke}{K^\ell a}\right)^a \cdot 2^{s(n+k)\log s} \\
&\leq \left(\frac{4enN^{1/2}}{aK^{\ell-1}}\right)^a \cdot 2^{s(n+k)\log s} \\
&\leq \left(\frac{8enN^{1/2}(s+\ell)}{K^\ell}\right)^a \cdot 2^{s(n+k)\log s} < 2^{-s}
\end{aligned}$$

■

4.2.1 A Remark on Oracle Implementation

Consider an implementation of the random permutation oracle with say $x \mapsto g^x$ over some field $\mathbb{F}_p$ for which g is a generator. Then the output is clearly not incompressible; $\mathsf{Enc}(x, f(x)) = g^x$ (and the corresponding Dec) compresses f to length $O(\log p)$.

4.2.2 Alternative Constructions

Using the same techniques, we can also prove that the function obtained by replacing $\mathcal{O}$ in 4.3 with a random permutation oracle, or taking a random function from $\{0,1\}^k$ to $\{0,1\}^n$ yields an incompressible function.

5 Discussion

The proofs in this paper depend on a very fundamental manner on the information-theoretic one-wayness of random function and permutation oracles, and as a result, the techniques used are limited to such settings. The problem as to whether we obtain similar results without oracles under standard complexity or cryptographic assumptions, such as the existence of (strongly) one-way functions, remains open.

6 Acknowledgements

Thanks to Luca Trevisan for suggesting both the problem and the approach, and for helpful discussions and support throughout the course of this work. I would also like to thank Cynthia Dwork for pointing out [DLN96] to us, and for reading an earlier draft of this paper.

References

[CT91] T.M. Cover and J.A. Thomas. *Elements of Information Theory.* John Wily & Sons, Inc., 1991.

[DLN96] Cynthia Dwork, Jeffrey Lotspiech and Moni Naor, "Digital Signets: Self-Enforcing Protection of Digital Information", *Proceedings of STOC 1996.*

[GS91] Andrew V. Goldberg and Michael Sipser. Compression and Ranking. *SIAM Journal on Computing*, 20:524-536, 1991.

[GT00] Rosario Gennaro and Luca Trevisan, "Lower Bounds on Efficiency of Generic Cryptographic Constructions", *Proceedings of FOCS 2000.*

[Imp99] Russell Impagliazzo, October 1999. Remarks in Open Problem session at the DIMACS Workshop on Pseudorandomness and Explicit Combinatorial Constructions.

[Tre02] Luca Trevisan, private communication.

[TVZ03] Luca Trevisan, Salil Vadhan and David Zuckerman, "Compression of Samplable Sources", submitted.