On Pseudoentropy versus Compressibility

Hoeteck Wee*
Computer Science Division
University of California, Berkeley
hoeteck@cs.berkeley.edu

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. A source has high pseudo-entropy if it is computationally distinguishable from a source of high entropy. 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 [4, 9], and provides an oracle separation between compressibility and pseudoentropy, thereby partially addressing an open problem posed in [6].*
2. *We also provide a separation between $1/s$-metric-type pseudoentropy and $1/s$-Yao-type pseudoentropy - which are two computational analogues of entropy introduced in [1] - for the class of oracle circuits of size s (s polynomially bounded). This is the first known separation result for metric-type and Yao-type pseudoentropy.*
3. *In the random oracle model, we show that there exists incompressible functions as defined in [3] where any substantial compression of the output of the function must reveal something about the seed. This yields the first known practical realization of incompressible functions, under the assumption that random oracles may be realized using cryptographic hash functions.*

Finally, we show that computational assumptions are needed to separate compressibility and pseudoentropy for samplable sources. In particular, if one-way functions do not exist, then any samplable flat source of entropy k can be compressed by circuits to length $k + O(\log n)$; furthermore, any such source has $1/2$-Yao-type pseudoentropy $k + O(\log n)$.

1. Introduction

A systematic study of data compression from a computational stand-point was initiated in [4] and extended more recently in [9]. 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 [2] 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 [6] in a talk (the case of general sources was partially addressed in [4]). Our work in this paper

* Work supported by NSF grant CCR 9984703 and a US-Israel Binational Science Foundation grant.

[1] The notion of efficiency in [4] and [9] 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.

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 from the uniform distribution over $\{0,1\}^n$. [4, 9] 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 [4] 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 [3] 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 its output can be efficiently recovered from $g(s)$ and its hardcore bit, and yet this information does not reveal the seed s.

In a more recent paper, Barak, Shaltiel and Wigderson [1] studied several notions of computational min-entropy: HILL-type pseudoentropy, which is essentially what is referred to as pseudoentropy herein, metric-type pseudoentropy, which is similar to HILL-type pseudoentropy but with a reversal of quantifiers, and Yao-type pseudoentropy, which is similar to compressibility, except it captures some worst-case behavior instead of an average-case behavior. They proved equivalence of HILL-type and metric-type pseudoentropy for circuits, and of all three types of pseudoentropy for circuits with an NP-oracle. In addition, they proved a separation of HILL-type and metric-type pseudoentropy for bounded-width read-once oblivious branching programs. An open problem posed in the paper is whether Yao-type pseudoentropy is equivalent to metric-type pseudoentropy for polynomial-sized circuits.

1.2. Our Contributions

1.2.1. Separating Pseudoentropy and Compressibility We prove a stronger version of the lower bound in Goldberg and Sipser [4] 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 [4] is not samplable (uniformly, that is; 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. Separating Metric-Type and Yao-Type Pseudoentropy A simple extension of the previous result also yields an oracle separation between metric-type and Yao-type pseudoentropy for polynomial size circuits. In particular, we exhibit an oracle under which there exists a samplable source with $1/s$-metric-type pseudoentropy $O(\log s)$ and $1/s$-Yao-type pseudoentropy $n - O(\log s)$ for oracle circuits of size s.

1.2.3. An Incompressible Function We apply the same techniques to prove the existence of in-

[2] In [4], this observation is attributed to L. Levin.

compressible functions as defined in [3] in the random oracle model. An incompressible function is one in which any substantial compression of its output must reveal something about its input. Furthermore, our proof yields a simple and practical construction of incompressible functions, under the assumption that random oracles may be realized using cryptographic hash functions.

1.2.4. Necessity of One-Way Functions Finally, we prove that we cannot expect an unconditional separation result between pseudoentropy and compressibility or between metric-type and Yao-type pseudoentropy for samplable sources using polynomial-sized circuits. In particular, we show that if one-way functions do not exist, then any samplable flat source of entropy k can be compressed by circuits to length $k+O(\log n)$, and has 1/2-Yao-type pseudoentropy $k+O(\log n)$.

2. Preliminaries

2.1. Basic notations

For a finite set S, we write $x \in_R S$ to say that x is distributed uniformly over the set S. We use U_n to denote the uniform distribution over the set $\{0,1\}^n$, and $\mathrm{neg}(n)$ to denote a function that is of the form $n^{-\omega(1)}$. The support $\mathrm{Sup}(X)$ of a distribution X is the set $\{x | \Pr[X = x] > 0\}$, and $H(X)$ denotes the (Shannon) entropy of X.

2.2. Basic definitions

Here, we review some definitions and observations, most of which have previously appeared in [2, 4, 9].

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 (Shannon) entropy is a lower bound for s-pseudoentropy.

Definition 2.2. *A source X_n over $\{0,1\}^n$ is* samplable *if there is a (uniform) polynomial-time algorithm S such that $S(1^n)$ is distributed according to X_n.*

Definition 2.3. *[9] Let X_n be a flat source over $\{0,1\}^n$, that is, every element in its support* $\mathrm{Sup}(X)$ *occurs with the same probability. We say that X_n is a* source with membership oracle *if there is a (uniform) polynomial-time algorithm T such that T such that $T(z) = 1 \Leftrightarrow z \in \mathrm{Sup}(X_n)$.*

Lemma 2.4. *Let X_n be a flat source over $\{0,1\}^n$ with membership oracle and $H(X_n) \le k$. 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, $H(D) \le H(X_n) + \epsilon n \le k + \mathrm{neg}(s)$. It follows that X_n has pseudoentropy at most $k + \mathrm{neg}(s)$. □

2.3. Basics of Compression

Definition 2.5. *[9] 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. $\mathrm{E}[\,|\mathsf{Enc}(X)|\,] \le m$.

If in addition we have $\mathsf{Enc} : \Sigma^* \to \Sigma^m$ *and in particular, $|\mathsf{Enc}(x)| = m$ for all $x \in \mathrm{Sup}(X)$, we say that* $(\mathsf{Enc}, \mathsf{Dec})$ *compresses source X to length exactly m.*

Definition 2.6. *[9] We say source X is* compressible *to length (exactly) m if there exists functions* Enc *and* Dec *such that* $(\mathsf{Enc}, \mathsf{Dec})$ *compresses X to length (exactly) m.*

Proposition 2.7. *[9] 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

[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.

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

can be achieved either using arithmetic coding [4] or condensers [9]:

Proposition 2.8. *[4, 9] 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 2.9. *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 2.11 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 2.7. □

2.3.1. 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 2.10. *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. If in addition we have* $\mathsf{Enc} : \Sigma^* \to \Sigma^m$, *we say that* $(\mathsf{Enc}, \mathsf{Dec})$ *α-somewhere compresses source X to length exactly m, and that X is α-somewhere compressible to length exactly m.*

Proposition 2.11. *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:

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

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 $2s + O(1)$.

2.3.2. 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 [9]) 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 2.12. *(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 $X_n = G(U_{n^\epsilon})$. □

2.4. Computational Analogues of (Min)Entropy

Definition 2.13. *A source X_n over $\{0,1\}^n$ has* (statistical) min-entropy at least k, denoted by $H_\infty(X_n) \geq k$ *if for every $x \in \{0,1\}^n$, $\Pr[X_n = x] \leq 2^{-k}$.*

The following definitions are from [1], specialized to the class of circuits and random variables over $\{0,1\}^n$.

Definition 2.14. *[1] Let X_n be a source over $\{0,1\}^n$, and let $\epsilon \geq 0$. We say that X_n has* ϵ-HILL-type pseudoentropy at least m, denoted by $H_\epsilon^{\text{HILL}}(X_n) \geq m$, if there exists a source Y over $\{0,1\}^n$ with (statistical) min-entropy at least m such that for every test $T : \{0,1\}^n \to \{0,1\}$ that is computable by a circuit of size at most s, $\mid \Pr[T(X_n) = 1] - \Pr[T(Y) = 1] \mid < \epsilon$.

Definition 2.15. *[1] Let X_n be a source over $\{0,1\}^n$, and let $\epsilon \geq 0$. We say that X_n has* ϵ-metric-type pseudoentropy at least m, denoted by $H_\epsilon^{\text{Metric}}(X_n) \geq m$, if for every test $T : \{0,1\}^n \to \{0,1\}$ that is computable by a circuit of size at most s, there exists a source Y over $\{0,1\}^n$ which has (statistical) min-entropy at least m such that $\mid \Pr[T(X_n) = 1] - \Pr[T(Y) = 1] \mid < \epsilon$.

Definition 2.16. *[1] Let X_n be a source over $\{0,1\}^n$, and let $\epsilon \geq 0$. We say that X_n has* ϵ-Yao-type pseudoentropy[5] at least m, denoted by $H_\epsilon^{\text{Yao}}(X_n) \geq m$, if for every $\ell < m$, X_n is not $(2^{\ell-m} + \epsilon)$-somewhere compressible to length exactly ℓ.

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:

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

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

Note that this lower bound in Corollary 3.2 matches the upper bound in Prop 2.8.

[8] also pointed out a simpler construction of compression and decompression functions that compresses the source $f(U_k)$ to length $n - \log s + O(1)$ for $k < n - 2\log s$, which matches the lower bound in Theorem 3.1 (up to constant multiples of $\log s$). Fix a family of linear pairwise independent hash functions $\mathcal{H} = \{h : \{0,1\}^n \to \{0,1\}^{n-\log s}\}$ (for instance, using linear functions over finite fields or Toeplitz matrices). Pick a random $h \in \mathcal{H}$, which is injective on at least a fraction $1 - 1/n$ of the elements of $f(\{0,1\}^k)$ with constant probability (the analysis is similar to that for Theorem 6.3). Fix one such function h_n, and encode $x \in \{0,1\}^n$ as $h(x)$

[5] One way to interprete Yao-type pseudoentropy is to regard it as a worst-case measure of incompressibility, the way min-entropy can be regarded as a worst-case measure of Shannon entropy. A source has low min-entropy if there is some element in the support with low probability. Similarly, a source has low Yao-type pseudoentropy if there is some subset that can be almost optimally compressed (in the sense of Definition 2.5).

and decode y by enumerating over the pre-images of $h^{-1}(y)$ (since h is linear, the subspace $h^{-1}(y)$ is efficiently computable) and taking the unique pre-image that is accepted by the membership oracle. This yields compression and decompression circuits of size $O(n^2 + ns)$ that $(1 - 1/n)$-somewhere compresses $f(U_k)$ to length $n - \log s$, which can be transformed into compression and decompression circuits of size $O(n^2 + ns)$ that compresses $f(U_k)$ to length $n - \log s + O(1)$.

In addition, note that we cannot extend to the result to $k < \log s$, since in $O(s)$ time, we can query $\mathcal{O}_f^S$ on all of $\{0,1\}^k$ in that case, and compress $f(U_k)$ to length exactly $\lceil k \rceil$.

3.2. Main Idea

Let compf 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

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

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

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

Observe that $\mathsf{compf} = \mathsf{invertible} \cup \mathsf{forgeable}$[6]. 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 [5]. 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 [5] 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 $|\mathsf{compf}|$, 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 y. 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 (by construction), such that we can describe f (in a information-theoretic sense) given:

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

Greedy-Construct-T

[6] 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 $\mathsf{forgeable}$.

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 $x_1, \ldots, x_q$ be the queries $\mathcal{A}$ makes to $\mathcal{O}_f^S$ (in the order the queries are made), where $x_q = x$ and $q \leq s'$.
3. Remove $f(x_1), \ldots, f(x_{q-1})$ from $f(\mathsf{invert}_f)$ (note that some of these values may have already been removed in previous iterations). Then, all the elements $x_1, \ldots, x_{q-1}$ 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 on $f^{-1}(T)$, start with a look-up table for f on values in $\{0,1\}^k - f^{-1}(T)$, 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 (we will explain why we can simulate oracle responses in the next 2 steps). 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 $f|_{\{0,1\}^k - f^{-1}(T)}$, 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

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

bits. It follows that

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

3.3.2. forgeable is small

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

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

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:

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

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

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

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[7] 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 $x_1,\dots,x_q$ be the queries Dec makes to $\mathcal{O}_f^S$; and let $y'_1,\dots,y'_r$ be the queries Dec makes to $\mathcal{O}_f^M$. Set $a_z = r$, so that the a_z-th query that Dec makes to $\mathcal{O}_f^M$ is $f(x)$ (since the simulation is halted after that).
3. Remove $\mathsf{Enc}(f(x_1)),\dots,\mathsf{Enc}(f(x_q))$ and $\mathsf{Enc}(y'_1),\dots,\mathsf{Enc}(y'_{r-1})$ from $\mathsf{Enc}(f(\mathsf{forge}_f))$ (ignoring those values that are not in $\mathsf{Enc}(f(\mathsf{forge}_f))$). In addition, we continue to remove the lexicographically smallest element in $\mathsf{Enc}(f(\mathsf{forge}_f))$ until we have removed exactly $s-1$ elements in all of step 3.
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 when 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 of 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 $s-1$ elements (other than z) 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. Furthermore, those elements we remove succeed z in lexicographic order, and must have length at least $|z|$. It follows that

$$\begin{aligned}\mathrm{E}_{x\in W}\left[\,|\mathsf{Enc}(f(x))|\,\right] &\leq \mathrm{E}_{x\in \mathsf{forge}_f}\left[\,|\mathsf{Enc}(f(x))|\,\right]\\ &\leq \frac{n}{n-1}\cdot(n-d) \quad \Box\end{aligned}$$

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

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

described using

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

bits. It follows that

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

3.3.3. Rest of the proof From lemmas 3.5 and 3.7, we have (for the parameters stated in the theorem)

$$|\mathsf{compf}|<2^{-s}\tbinom{N}{K}K!$$

The result follows readily from this and lemma 2.4.

4. Separating Metric-type and Yao-type Pseudoentropy

The construction is the same as that used in Theorem 3.1, and the analysis is very similar too.

Theorem 4.1. *For any k satisfying $7\log s+O(1)<k<n$, there exists (injective) functions $f:\{0,1\}^k\to\{0,1\}^n$ such that limited to the model of circuits of size s with oracle access to $\mathcal{O}_f$, $H^{\text{Metric}}_{1/s}(f(U_k))\leq k+1$ but $H^{\text{Yao}}_{1/s}(f(U_k))>n-2\log s-O(1)$. Furthermore, $f(U_k)$ is samplable.*

First, we observe that if we consider the model of efficient algorithms with $s=\mathit{poly}(n)$ bits of advice, then for any injective function $f:\{0,1\}^k\to\{0,1\}^n$, $f(U_k)$ is $2^{-k}s/n$-somewhere compressible to length exactly $\lceil\log(s/n)\rceil$. To accomplish this, use the s bits of advice to specify the concatenation of $f(a_0),f(a_1),\ldots,f(a_{s/n-1})$, where a_i denotes the binary representation of i (padded with leading 0's to length k), and we can compress these s/n strings optimally to length $\lceil\log(s/n)\rceil$ by sending $f(a_i)$ to a_i (truncated to the last $\lceil\log(s/n)\rceil$ bits). This tells us that $H^{\text{Yao}}_0(f(U_k))\leq k$. Therefore, to establish the $n-2\log s-O(1)$ lower bound on $H^{\text{Yao}}_{1/s}(f(U_k))$, we have to use the $1/s$ in an essential manner, as it allows us to simply neglect subsets of $f(\{0,1\}^k)$ of density less than $1/s$ that could be optimally compressed, and only establish a compressibility lower bound for subsets of density at least $1/s$.

Proof. To see why $H^{\text{Metric}}_{1/s}(f(U_k))\leq k+1$ (which holds for all injective functions f), consider the test T given by $\mathcal{O}^M_f$. Then, $\Pr[T(f(U_k))=1]=1$, whereas for any source Y over $\{0,1\}^n$ of min-entropy at least $k+1$, $\Pr[T(Y)=1]=\sum_{z\in f(\{0,1\}^k)}\Pr[Y=z]\leq 1/2$.

Again, let us fix n and k and study for which d do we get $H^{\text{Yao}}_{1/s}(f(U_k))\leq n-d$. We will also retain the notation from Section 3 (though the references are somewhat different). Let compf be the set of functions $f\in\mathcal{F}$ for which $H^{\text{Yao}}_{1/s}(f(U_k))\leq n-d$. For each such f, there exists $\ell<n-d$, a set $D_f\subset f(\{0,1\}^k)$ such that $|D_f|\geq(2^{\ell-(n-d)}+1/s)\cdot 2^k$ and the uniform distribution over D_f is compressible to length exactly ℓ using some oracle circuits $(\mathsf{Enc},\mathsf{Dec})$ of size s. We may then define

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

Clearly, invert_f and forge_f form a partition of $f^{-1}(D)$. In addition, we define

$$\begin{aligned}
\mathsf{invertible} &= \left\{f\in\mathsf{compf}:|\mathsf{invert}_f|\geq\tfrac{1}{2s}\cdot 2^k\right\}\\
\mathsf{forgeable} &= \left\{f\in\mathsf{compf}:\right.\\
&\qquad\left.|\mathsf{forge}_f|\geq\left(2^{\ell-(n-d)}+\tfrac{1}{2s}\right)\cdot 2^k\right\}
\end{aligned}$$

Again, observe that $\mathsf{compf}=\mathsf{invertible}\cup\mathsf{forgeable}$.

By the same analysis as used in Lemma 3.5, we have for $k>7\log s+O(1)$,

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

Furthermore, we have the following analogue of Lemma 3.6:

Lemma 4.2. *Take any $f\in\mathsf{forgeable}$, and let $(\mathsf{Enc},\mathsf{Dec})$ be oracle circuits of size s that compress the uniform distribution over D_f to length exactly ℓ, and also satisfy $|\mathsf{forge}_f|\geq\left(2^{\ell-(n-d)}+\frac{1}{2s}\right)\cdot 2^k$. Then, f can be described using*

$$a\ell+\log\tbinom{K}{a}+\log\tbinom{N-a}{K-a}(K-a)!+a\log s$$

bits, given Dec*, where* $a = \left(2^{\ell-(n-d)} + \frac{1}{2s}\right) \cdot 2^k/s$*.*

Proof. (sketch) The proof is similar to that for Lemma 3.6: again, we show that there exists a subset W of forge_f of size a, such that we can describe f given:

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

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. Now, Enc has fixed output length ℓ, so we may describe $\mathsf{Enc}(f(W))$ using just $a\ell$ bits. □

This yields the following analogue of Lemma 3.7: for $k > 7\log s + O(1)$ and $d > 2\log s + O(1)$,

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

and thus

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

Combining the upper bounds on $|\mathsf{invertible}|$ and $|\mathsf{forgeable}|$, we have

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

and the theorem follows. □

5. An Incompressible Function From Cryptography

5.1. On Incompressible Functions

5.1.1. Motivation Dwork, Lotspiech and Naor [3] 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.

5.1.2. A Formal Definition

Definition 5.1. *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 $(\mathsf{Enc}, \mathsf{Dec})$ 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 5.2. *We say the function* $f : \{0,1\}^k \to \{0,1\}^n$ *is* incompressible *if there exists no polynomial size circuits* $(\mathsf{Enc}, \mathsf{Dec})$ *that securely compresses f to length $o(n)$.*

The key distinction between the notion of compressibility here and that in section 2.3 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 (since the seed is exposed to Enc), so it is no longer necessarily the case that f cannot be efficiently compressed to length $n - O(1)$.

5.2. Existence of Incompressible Functions in the Random Oracle Model

Theorem 5.3. *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 5.2) with probability $1 - \mathrm{neg}(k)$ *(over the choices made by the random oracle).*

Let $\mathcal{H}$ be the set of functions $h : \{0,1\}^k \to \{0,1\}^k$, 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 $(\mathsf{Enc}, \mathsf{Dec})$ that compresses h_f, we define

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

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 $(\mathsf{Enc}, \mathsf{Dec})$ securely compresses h_f:

Lemma 5.4. *Suppose* $(\mathsf{Enc}, \mathsf{Dec})$ *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* $\mathrm{poly}(s)$ *such that*

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

In particular, if $s = \mathrm{poly}(n)$*, then* $(\mathsf{Enc}, \mathsf{Dec})$ *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 $(\mathsf{Enc}, \mathsf{Dec})$ of size s that compresses h_f to length $n/2$ and satisfy $|\,\mathsf{invert}_{h,(\mathsf{Enc},\mathsf{Dec})}| \leq 2^k/n$. It follows from lemma 5.4 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 5.3, it suffices to establish a (strong) upper bound on $|\mathsf{s-compressible}|$.

Lemma 5.5. *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*

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

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 $a\left(\frac{n}{n-1} \cdot \frac{n}{2} + \log n + 1\right)$ bits 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 $y_1 \dots y_\ell$.

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 $f(x+1), \dots, f(x+\ell)$ (given by $y_1, \dots, y_\ell$ respectively). In addition, we can use the ordering $\mathsf{Enc}(W, f_h(W))$ to figure out x. We can then add $(x_1, y_1), \dots, (x_\ell, y_\ell)$ 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 5.6.

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

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

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

bits. It follows that

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

5.2.1. A Remark on Oracle Implementation Consider an implementation of the random 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 $O(\log p)$.

On the other hand, Theorem 5.3 does suggest a practical realization of an incompressible function: let $H : \{0,1\}^* \to \{0,1\}^k$ be a cryptographic hash function (such as `SHA-1`). Under the assumption that cryptographic hash functions are a practical realization of random oracles, f as defined below is an incompressible function:

$$f(x) = H(x+1)H(x+2)\dots H(x+n/k)$$

Using the same techniques, we can also prove that the function obtained by replacing $\mathcal{O}$ in 5.3 with a random permutation oracle, or taking a random function from $\{0,1\}^k$ to $\{0,1\}^n$ yields an incompressible function.

6. Necessity of One-Way Functions

In the previous sections, we gave separation results between pseudoentropy and compressibility and between metric-type and Yao-type pseudoentropy for samplable sources in an oracle setting in which one-way functions exist. In this section, we show that the existence of one-way functions are in fact *necessary* to prove such results.

Informally, a function f is (uniformly) one-way if it is easy to compute but hard to invert by probabilistic polynomial-time algorithms. We do not know whether one-way functions exist, but even if they do not exist, it seems conceivable that there may exist a easy to compute many-to-one function f for which that finding a pre-image is easy, whereas uniformly sampling from the pre-images is hard. This turns out to be impossible, as stated precisely in the following definition and theorem:

Definition 6.1. *[7] A function f is* distributionally one-way *if, for some constant $c > 0$, for every feasible (probabilistic) algorithm A, the distribution defined by $x \circ f(x)$ and the distribution defined by $A(f(x)) \circ f(x)$ have statistical distance at least n^{-c}, where $x \in_R \{0,1\}^n$.*

Theorem 6.2. *[7] If there is a distributionally one-way function, then there is a (uniformly) one-way function.*

Now, suppose one-way functions and thus distributionally one-way functions do not exist. Let X be a samplable flat source X over $\{0,1\}^n$ of entropy k. If the sampling algorithm S for X uses exactly k random bits, then we can optimally compress X by inverting the function S. However, S could use $n^{O(1)}$ random bits. We know that a random hash function h from $\{0,1\}^n$ to $\{0,1\}^{k+2\log n}$ maps most elements in the support of X to distinct elements. Thus, to compress X, we simply apply h. To decompress, it is not sufficiently to simply invert h, since we need to identify the pre-images of h that is in the support of X; moreover, there may be an exponential number of pre-images (though if h is linear, these pre-images have a compact description). To address both problems, we will instead sample from the preimages of $h \circ S$ (approximately uniformly).

Theorem 6.3. *If (uniformly) one-way functions do not exist, then any samplable flat source X over $\{0,1\}^n$ of entropy k can be compressed to length $k + 2\log n + O(1)$. Furthermore, $H^{\text{YAO}}_{1/2}(X) \leq k + 2\log n + O(1)$.*

Proof. Let X be a samplable flat source X over $\{0,1\}^n$ of entropy k. Let $W = \text{Sup}(X)$, and $S : \{0,1\}^r \to \{0,1\}^n$ be an efficient sampling algorithm for X, that is, $S(U_r) = X$. Fix a family of linear pairwise independent hash functions $\mathcal{H} = \{h : \{0,1\}^n \to \{0,1\}^{k+2\log n}\}$, and for each $h \in \mathcal{H}$, define:

$$C_h = \big\{x \in W \mid \exists\, y \in W : y \neq x \wedge h(y) = h(x)\big\}$$

that is, C_h is subset of W with collisions under h. A simple union bound tells us that for any $x \in W$,

$$\Pr_h\left[\exists y \in W : y \neq x \;\wedge\; h(y) = h(x)\right] \leq \frac{2^k}{2^{k+2\log n}} = \frac{1}{n^2}$$

Therefore, $\mathrm{E}\big[\,|C_h|\,\big] \leq \frac{1}{n^2} \cdot 2^k$. We say that $h \in \mathcal{H}$ is good if $|C_h| \leq \frac{1}{n} \cdot 2^k$, that is, h maps most elements in W to distinct images. Then, a straightforward application of Markov's inequality yields $\Pr_h\left[|C_h| > \frac{1}{n} \cdot 2^k\right] < \frac{1}{n}$; that is, at most a $1/n$ fraction of $\mathcal{H}$ is not good.

Next, consider the function $f : \{0,1\}^k \times \mathcal{H} \to \mathcal{H} \times \{0,1\}^{k+2\log n}$ given by $f(a,h) = (h, h(S(a)))$. Clearly, f is polynomial-time computable, so under the assumption that one-way functions and hence distributionally one-way functions do not exist, there exists a probabilistic polynomial-time $A = (A_1, A_2)$ with $A : \mathcal{H} \times \{0,1\}^{k+2\log n} \to \{0,1\}^k \times \mathcal{H}$ such that the distribution defined by $(a, h, f(a,h))$ and that defined by $(A(f(a,h)), f(a,h))$ have statistical distance at most $1/n$, where $a \in_R \{0,1\}^r$ and $h \in \mathcal{H}$. Considering the statistical test induced by f, we obtain:

$$\Pr_{M,a,h}\Big[f(A(f(a,h))) = f(a,h)\Big] \geq 1 - \frac{1}{n}$$

(with the probability taken over the random coin tosses of M, as well as $a \in_R \{0,1\}^r$ and $h \in \mathcal{H}$) and thus,

$$\Pr_{M,a,h}\Big[f(A(f(a,h))) = f(a,h) \text{ and } h \text{ is good}\Big] \geq 1 - \frac{2}{n}$$

We may then fix the coin tosses of M and also fix $h = h_0$ such that h_0 is good, and

$$\Pr_a\Big[f(A(f(a,h_0))) = f(a,h_0)\Big] \geq 1 - \tfrac{2}{n}$$

Now, consider any a such that $f(A(f(a,h_0))) = f(a,h_0)$. Writing $A(f(a,h_0)) = (a', h')$ (where $A_1(f(a,h_0)) = a'$ and $A_2(f(a,h_0)) = h'$), we have $f(a',h') = f(a,h_0)$, that is, $h' = h_0$ and $h_0(S(a')) = h_0(S(a))$. Furthermore, with probability at least $1 - 1/n$ over a, $S(a) \notin C_{h_0}$, in which case $S(a') = S(a)$, that is, $S(A_1(h_0, h_0(S(a)))) = S(a)$. Therefore, if we define the circuits $(\mathsf{Enc}, \mathsf{Dec})$ given by $\mathsf{Enc}(y) = h_0(y)$ and $\mathsf{Dec}(y) = S(A_1(h_0, y))$, then we have:

$$\begin{aligned} &\Pr_a\Big[\mathsf{Dec}\big(h_0(S(a))\big) = S(a)\Big] \\ =\; &\Pr_{x \in X}\Big[\mathsf{Dec}(h_0(x)) = x\Big] \geq 1 - \frac{3}{n} \end{aligned}$$

Hence, $(\mathsf{Enc}, \mathsf{Dec})$ $(1 - \frac{3}{n})$-somewhere compresses X to exactly length $k + 2\log n$, so X can be compressed to length $k + 2\log n + O(1)$ by polynomial-sized circuits, and $H^{\text{YAO}}_{1/2}(X) \leq k + 2\log n + O(1)$. □

Note that a similar result holds even if X is only approximately samplable, that is, $S(U_r)$ and X are only statistically close (or even just computationally indistinguishable by polynomial-sized circuits).

7. Discussion

The proofs for the separation results 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 (at least as strong as the existence of one-way functions) remains open.

8. Acknowledgements

Most of the ideas in this work originated from discussions with Luca Trevisan; I am very grateful to him for getting me started on this problem, and for his guidance and support throughout the course of this work. In addition, I would like to thank Cynthia Dwork for pointing out [3] to us; Ronen Shaltiel for helpful discussions regarding [1]; and Andrej Bogdanov for raising the question of whether one-way functions are necessary for the separation results.

References

[1] Boaz Barak, Ronen Shaltiel and Avi Wigderson. "Computational Analogues of Entropy", *Proceedings of RANDOM 2003.*

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

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

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

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

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

[7] Russell Impagliazzo and Michael Luby. "One-way Functions are Essential for Complexity Based Cryptography", *Proceedings of FOCS 1989.*

[8] Luca Trevisan, private communication.

[9] Luca Trevisan, Salil Vadhan and David Zuckerman, "Compression of Samplable Sources", *Proceedings of CCC 2004.*