Pseudo-random generation from one-way functions (Extended Abstract)

Russell Impagliazzo*
Department of Mathematics
U.C. Berkeley
Russelli@ernie.berkeley.edu

Leonid A. Levin†
Computer Science Department
Boston University
Lnd@bu-cs.bu.edu

Michael Luby‡
International Computer Science Institute
Berkeley, California
Luby@icsi.berkeley.edu

Abstract

We show that the existence of one-way functions is necessary and sufficient for the existence of pseudo-random generators in the following sense. Let f be an easily computable function such that when x is chosen randomly: (1) from $f(x)$ it is hard to recover an x' with $f(x') = f(x)$ by a small circuit, or; (2) f has small degeneracy and from $f(x)$ it is hard to recover x by a fast algorithm. From one-way functions of type (1) or (2) we show how to construct pseudo-random generators secure against small circuits or fast algorithms, respectively, and vice-versa. Previous results show how to construct pseudo-random generators from one-way functions that have special properties ([Blum, Micali 82], [Yao 82], [Levin 85], [Goldreich, Krawczyk, Luby 88]).

We use the results of [Goldreich, Levin 89] in an essential way.

1 Introduction

One of the basic primitives in cryptography and other areas of computer science is a pseudo-random generator. A pseudo-random generator can be used to build secure private key encryption protocols ([Goldwasser, Micali 82], [Goldreich, Goldwasser, Micali 84], [Luby, Rackoff 86]). [Goldreich, Micali, Wigderson 86] shows that any problem in NP has a zero-knowledge proof system if bit commitment is possible, and [Naor 88] shows how to construct a bit commitment protocol based on a pseudo-random generator. [Yao 82] shows that the existence of pseudo-random generators implies that $BPP \subset DTime(2^{n^\epsilon})$ for every $\epsilon > 0$.

On the other hand, there are many natural problems that are conjectured to be one-way functions, whereas it is hard to think of a natural example of a conjectured (perfect) pseudo-random generator. Thus, it is desirable to convert what seems to arise naturally (one-way functions) into a valuable commodity (a pseudo-random generator).

The first construction of a pseudo-random generator [Blum, Micali 82] is based on the intractability of the discrete log problem. [Yao 82] generalizes this by showing that a pseudo-random generator can be constructed from any one-way permutation. [Levin 85] shows that the existence of functions that are one-way on a quadratic number of iterates is necessary and sufficient for the existence of pseudo-random generators. [Goldreich, Krawczyk, Luby 88] show that any one-way function for which the preimage sizes of all elements in the range are roughly equal is sufficient. (The actual condition is slightly weaker.)

We show that the existence of one-way functions is necessary and sufficient for the existence of pseudo-

*Research partially supported by NSF grant CCR 88-13632
†Supported by NSF grant DCR-8607492
‡On leave of absence from the University of Toronto, research partially supported by NSERC operating grant A8092

random generators in the following sense. Let f be an easily computable function such that when x is chosen randomly: (1) from $f(x)$ it is hard to recover an x' with $f(x') = f(x)$ by a small circuit, or; (2) f has small degeneracy and from $f(x)$ it is hard to recover x by a fast algorithm. From one-way functions of type (1) or (2) we show how to construct pseudo-random generators secure against small circuits or fast algorithms, respectively, and vice-versa.

Notation : Let x and y be bit strings. Then, $|x|$ is the length of x, $x \circ y$ is the concatenation of x and y, x_i is the i^{th} bit of x and $x \uparrow i$ is the first i bits of x. If α is a number, then $|\alpha|$ is the absolute value of α. Let x and y be two equal length bit strings. $x \odot y$ is the inner product mod 2 of x and y and $x \oplus y$ is the vector sum mod 2 (i.e. bitwise parity) of x and y.

1.1 Security

In this paper we consider both uniform and non-uniform models of security. The difference between the two models of security is that, in the uniform model, the adversary is a fast algorithm, whereas, in the non-uniform model, the adversary is a small circuit. At first glance, the non-uniform notion of security seems too strong of a requirement to place on a cryptographic protocol. However, a protocol that is only secure in the uniform sense is susceptible to the following type of attack. The time allowed for computation by an adversary before the protocol begins may be much greater than the allowable time during the protocol. The result of the preprocessing can then be used by the adversary to break the protocol within the allowable time. Security in the non-uniform model is equivalent to immunity from this type of attack (see [Karp, Lipton 80]). Also, the existence of a pseudo-random generator with non-uniform security is used to prove that $BPP \subset DTime(2^{n^\epsilon})$ for every $\epsilon > 0$ [Yao 82].

Definition (resources) : A *resource class* R is class of functions from N to N that includes the identity function $r(n) = n$. If $r'(n) \leq r(n) \in R$ then $r'(n^2) + 1 \in R$. Finally, $\log_n(r(n)) < n$ is monotone.

We use resource classes to parameterize the resources allowed for adversaries in breaking one-way functions, pseudo-random generators and other cryptographic tasks. For example, R might be all polynomial bounded functions or all functions bounded by $2^{O(\log^c n)}$ for some constant $c > 1$.

Definition (uniform and non-uniform) : A *uniform* algorithm is a probabilistic Turing machine. The *time bound* $T(n)$ for a uniform algorithm is the maximum running time on inputs of length n. A *non-uniform* algorithm is a pair of Turing machines A and M. A is a preprocessing algorithm that on input of length n produces a string $A(n)$. The running time of $A(n)$ is not limited. Algorithm M accepts as input x and $A(|x|)$. The *time bound* $T(n)$ for a non-uniform algorithm is maximum running time of M over all inputs x and $A(|x|)$. From the results of [Karp, Lipton 80], a non-uniform algorithm with time bound in resource class R is equivalent to a (recursive) family of circuits with the size of the circuit for inputs of length n bounded by a function $r(n)$ where r is in resource class R.

Definition (feasible) : A uniform (non-uniform) algorithm is *feasible* with respect to resource class R if the time bound function is in R.

Definition (negligible) : A function $p : N \rightarrow N$ is *negligible* with respect to resource class R if for all $r \in R$, for almost all n, $p(n) \leq 1/r(n)$.

For the remainder of the paper, unless stated otherwise, a feasible adversary algorithm is always with respect to an arbitrary but fixed resource class R. For each cryptographic task that we define, there are implicitly two definitions being made simultaneously, one with respect to the uniform and the other with respect to the non-uniform model of security. Unless otherwise stated, each definition, lemma, proposition and theorem has two versions, one in each model.

1.2 One-way functions

Intuitively, a function f is *one-way* if it is easy to compute but hard to invert, i.e. given x the value of $f(x)$ can be computed in polynomial-time but every feasible algorithm that receives as input $f(x)$ (when x is a randomly chosen string of length n) can output a y such that $f(y) = f(x)$ with only negligible probability. It has not yet been proven that one-way functions exist (if $P = NP$ then they certainly do not exist, but even if $P \neq NP$ it is not clear if they exist), but there are many examples of functions that seem to be one-way in practice and that are conjectured to be provably one-way. Some examples of conjectured one-way functions are factoring a composite number N that is the product of large randomly chosen primes, square roots modulo such an N, discrete

log modulo a large randomly chosen prime, problems from coding theory and the subset sum problem.

Notation (functions and probability ensembles) : A *length (function)* $l(n)$ is a monotone increasing function from N to N such that $l(n)$ is computable in time polynomial in n. A *function* f with input length $m(n)$ and output length $l(n)$ specifies for each $n \in N$ a function $f_n : \{0,1\}^{m(n)} \rightarrow \{0,1\}^{l(n)}$. For simplicity, we write $f(x)$ in place of $f_n(x)$. We say that f is *polynomial-time computable* if there is a polynomial-time Turing machine that on input $x \in \{0,1\}^{l(n)}$ computes $f(x)$. A *probability ensemble* D with length $m(n)$ assigns to each positive integer n a probability distribution D_n on bit strings of length $m(n)$. The *uniform ensemble* U assigns to each positive integer n the uniform probability distribution U_n on strings of length n. For $X \subseteq \{0,1\}^{m(n)}$, $D[X]$ is the sum over all $x \in X$ of the probability of x with respect to D_n. We use the notation $x \in_D \{0,1\}^{m(n)}$ to mean that x is randomly chosen from $\{0,1\}^{m(n)}$ according to D_n. We say that D is *polynomial-samplable* if there is a polynomial-time Turing machine M with input length $k(n)$ and output length $m(n)$ such that, for each $n \in N$, $M(x) \in_D \{0,1\}^{m(n)}$ when $x \in_U \{0,1\}^{k(n)}$. Define $f(D)$ to be the probability ensemble with length $l(n)$, where $f(D_n)$ is the probability distribution defined by the random variable $f(x)$ when $x \in_D \{0,1\}^{m(n)}$. For random variable X defined with respect to D_n, $\mathrm{Exp}[X]$ is the expected value of X. Pr is used for probability.

Note : Hereafter, unless stated otherwise, f is always a polynomial-time computable function with input length n and output length $l(n)$ and D and E are always probability ensembles with length n.

Definition (one-way function) : We say that f is *weakly one-way* on D if, for every feasible algorithm M, the inverting probability $\Pr[x = M(f(x))]$ when $x \in_D \{0,1\}^n$ is negligible. We say that f is *somewhat one-way on* D if, for some constant $c > 0$, for every feasible algorithm M, the inverting probability $\Pr[f(x) = f(M(f(x)))]$ when $x \in_D \{0,1\}^n$ is at most $1 - 1/n^c$. We say that f is *one-way on* D if, for every feasible algorithm M, the inverting probability $\Pr[f(x) = f(M(f(x)))]$ when $x \in_D \{0,1\}^n$ is negligible. We say that f is *one-way* if f is one-way on U.

If a function is one-way then it is both weakly one-way and somewhat one-way.

Proposition : If there is a polynomial-samplable D such that f is one-way on D then there is a one-way function g.

The function g is simply the composition of f and the polynomial-time computable sampling function for D. This proposition allows us to state most of our results in terms of one-way functions on the uniform ensemble as opposed to other probability ensembles.

Notation (copies of functions and ensembles) : Let $q(n)$ be a length function. Let D^q be the ensemble with length $n \cdot q(n)$ such that D^q_n is the distribution obtained by independently sampling $q(n)$ times from D_n and concatenating the results. Similarly, let f^q be the function with input and output lengths $n \cdot q(n)$ and $l(n) \cdot q(n)$, respectively, given by:

$$f^q(x_1 \circ \ldots \circ x_{q(n)}) = f(x_1) \circ \ldots \circ f(x_{q(n)}),$$

where $x_1, \ldots, x_{q(n)} \in \{0,1\}^n$.

The following is implicitly used in [Yao 82].

Proposition 1.1 (somewhat one-way $\rightarrow$ one-way) : If f is somewhat one-way on D with associated constant $c > 0$, then f^q is one-way on D^q, where $q(n) = n^{c+1}$.

1.3 Pseudo-random generators

Informally, a polynomial-time computable function f is *pseudo-random* if $f(x)$ is strictly longer than x and if every feasible algorithm can distinguish $f(x)$ from a truly random string of the same length (when x is chosen randomly) with only negligible probability. Intuitively, $f(x)$ "looks" just like a random string to any feasible algorithm, even though it is generated from a string x that is strictly shorter. This intuition is captured in the following definition of [Blum, Micali 82], [Yao 82].

Definition (prg) : f is a *generator* if $l(n) > n$ for all $n \in N$. The *distinguishing probability* $p(n)$ of algorithm M for f is $|\Pr[M(f(x)) = 1] - \Pr[M(y) = 1]|$ when $x \in_U \{0,1\}^n$ and $y \in_U \{0,1\}^{l(n)}$. We say that f is *pseudo-random* if every feasible algorithm has negligible distinguishing probability for f.

The normal definition of a pseudo-random generator insists that the generator can stretch the input by any polynomial amount. The following shows the definition above is equivalent, and follows from [Goldreich, Goldwasser, Micali 84].

Proposition 1.2 (polynomial stretching) : If there is a pseudo-random generator f then, for every constant $c > 1$, there is a pseudo-random generator with input length n and output length n^c.

We can now state one of our main theorems.

THEOREM A (one-way $\rightarrow$ prg) : If there is a one-way function in the non-uniform model of security then there is a pseudo-random generator in the non-uniform model of security.

The proof of Theorem A can be found in Section 5. The following definitions are necessary to state our next main theorem. The following definition is from [Shannon].

Definition (Shannon entropy) : The *(Shannon) entropy* of D_n is given by

$$Ent(D_n) = -\sum_{x \in \{0,1\}^n} D[\{x\}] \cdot \log(D[\{x\}]).$$

The *entropy function* $Ent(D)$ assigns to each $n \in N$ the value $Ent(D_n)$. For a function f, we call $Ent(f(D))$ the *(Shannon) entropy of f on D* and $Ent(D) - Ent(f(D))$ the *degeneracy* of f on D.

THEOREM B : The following are equivalent:

- There is a pseudo-random generator
- There is a polynomial-samplable D and f so that f is weakly one-way and has degeneracy $O(1)$ on D.

Proof : ($\rightarrow$) Without loss of generality, let f be a pseudo-random generator such that on input length n the output length is $2n$ (see Proposition 1.2). Let $g(x) = f(x) \uparrow |x|$ and let $g_i(x)$ be the i^{th} iterate of g on x, i.e. $g_0(x) = x$ and $g_{i+1}(x) = g(g_i(x))$. Let D_n be the distribution given by $g_i(x)$ when $x \in_U \{0,1\}^n$ and $i \in_U \{0, \ldots, n\}$. Note that D is polynomial-samplable. The degeneracy of g on D is the average over i of the degeneracy of g on $g_i(x)$ when $x \in_U \{0,1\}^n$, which is at most $(n-0)/n = 1$. The degeneracy of f on D is thus at most 1, because it is at most the degeneracy of g on D.

We now show that f is one-way on D. Since f is a pseudo-random generator, every feasible algorithm has negligible probability of distinguishing D from U (as in the definition of pseudo-random generator, where D takes the role of $f(x)$). This follows using an argument similar to that used in [Goldreich, Goldwasser, Micali 84]. We claim that f is one-way on U. Let M be any feasible algorithm that has inverting probability $p(n)$ for f on U that is non-negligible. Then, we can use M to distinguish between $f(U_n)$ and U_{2n} with probability at least $p(n) - 1/2^n$. The distinguisher simply outputs 1 on input $y \in \{0,1\}^{2n}$ if $f(M(y)) = y$. The probability that for $y \in_U \{0,1\}^{2n}$ there exists an $x \in \{0,1\}^n$ with $f(x) = y$ is at most $1/2^n$, from which the claim follows. Any feasible algorithm to invert f on D must either distinguish D from U or invert f on U. Since both of these are impossible from the above, f is one-way on D.

($\leftarrow$) Use Proposition 1.4, Proposition 1.5 and Theorem C below. □

Let $r(n)$ be any function in the resource class. Theorem B remains true if: (1) in the $\rightarrow$ direction of the theorem, $O(1)$ degeneracy is replaced with $1/r(n)$ and (as in the proof) "weakly one-way" is replaced with "one-way"; (2) in the $\leftarrow$ direction of the theorem, $O(1)$ degeneracy is replaced with $\log(r(n))$. $\log(r(n))$ cannot be replaced with anything asymptotically bigger for the following reason. No algorithm on input $f(x)$ can output y such that $y = x$ with probability greater than $1/|f^{-1}(x)|$ when $x \in_U \{0,1\}^n$. Let $s(n)$ be such that for all $r \in R$, $\lim_{n \rightarrow \infty}\{r(n)/s(n)\} = 0$. Then, $f(x \circ y) = x$, where $|y| = \log(s(|x|))$, has degeneracy $\log(s(n))$ and is weakly one-way. On the other hand, f is useless for constructing a pseudo-random generator.

We now define a type of one-way function whose existence is equivalent to that of pseudo-random generators. This characterization is useful in proving Theorem B.

Definition (hidden bit) : A *bit (function)* b for f is a polynomial-time computable function with input length n that outputs a single bit. The *distinguishing probability* $p(n)$ of algorithm M for f and b on D is $|\Pr[M(f(x)) = b(x)] - \Pr[M(f(x) \neq b(x)]|$ when $x \in_D \{0,1\}^n$. b is *hidden* for f on D if every feasible algorithm has negligible distinguishing probability for f on D. b is *hidden* for f if it is hidden for f on U.

Notation (inner product bit) : Let $x, y \in \{0,1\}^n$. Let $f'(x \circ y) = f(x) \circ y$ and let $D'_n = D_n \circ U_n$. It is easy to see that if f is one-way on D then f' is one-way on D'. The *inner product* bit is $b(x \circ y) = x \odot y$.

The following is from [Goldreich, Levin 89].

Proposition 1.3 (weakly one-way → inner product bit is hidden) : Assume that f is weakly one-way on D. Let f', D' and bit b for f' be defined in terms of f and D as in the inner product bit definition. Then, b is hidden for f' on D'.

The idea of a function that hides a bit was introduced in the original construction of a pseudo-random generator [Blum, Micali 82] and has been central to all such constructions since that time. Proposition 1.3 presents an elegant, simple and general method of obtaining a hidden bit from a one-way function. However, naive use of Proposition 1.3 with an arbitrary one-way function has difficulties because of the following. Let f be any one-way function. Define one-way function f' as $f'(x \circ y) = f(x)$, where $|y| = |x|$. From Proposition 1.3, $f''(x \circ y \circ r) = f(x) \circ r$, where $|r| = |x \circ y|$, hides the inner product bit. However, for cryptographic purposes this is useless, as the hidden bit is informationally impossible to recover. The hidden bit is only cryptographically useful if it is also *meaningful* as defined below. (We later show how to use Proposition 1.3 to extract hidden and meaningful bits from one-way functions.)

Definition (meaningful bit) : Let b be a bit for f. An *unbounded adversary* M for b and f is an algorithm with *unbounded time and space resources.* The *distinguishing probability* $p(n)$ of algorithm M on D is $|\Pr[M(f(x)) = b(x)] - \Pr[M(f(x)) \neq b(x)]|$ when $x \in_D \{0,1\}^n$. b is *meaningful* for f on D if there is a constant $c > 0$ and an unbounded adversary M such that $p(n) \geq 1/n^c$. The bit b is *meaningful* for f if it is meaningful for f on U.

Theorem C (hidden and meaningful → prg) : If there is a bit b that is both hidden and meaningful for f then there is a pseudo-random generator.

The proof of Theorem C can be found in Section 4.

Proposition 1.4 : Suppose that D is polynomial-samplable and that f is weakly one-way and has degeneracy $O(1)$ on D. Then there is a polynomial-time computable f' and a polynomial-samplable D' and a bit b for f' such that b is both meaningful and hidden for f' on D'.

Proof Sketch : Construct f', D' and b from f and D as in the definition of the inner product bit. Then, by Proposition 1.3, b is hidden for f' on D'. Because f has degeneracy $O(1)$ on D, it can be shown that b is meaningful for f' on D'. □

Proposition 1.5 : Let D be polynomial-samplable and let bit b be both hidden and meaningful for f on D. Then there is polynomial-time computable function f' and a bit b' for f' such that b' is both hidden and meaningful for f'.

Proof Sketch : Let M be the sampling algorithm for D. The construction is to let $f'(x) = f(M(x))$ and let $b'(x) = b(M(x))$. □

2 Background

Definition (statistically indistinguishable and quasi-random) : D_n and E_n are *statistically indistinguishable within* δ if for every $X \subseteq \{0,1\}^n$, $|D[X] - E[X]| < \delta$. D_n is *quasi-random within* δ if D_n is statistically indistinguishable within δ from U_n.

We need to use a variant definition of entropy used in [Chor, Goldreich 85].

Definition (min-entropy) : The *min-entropy* of D_n is $\min_{x \in \{0,1\}^n}\{-\log(D[\{x\}])\}$.

Intuitively, if a distribution has min-entropy k, it is "at least as random" as the uniform distribution on k bit strings. There are distributions that have arbitrarily large entropy but have only one bit of min-entropy.

Some of the definitions given in this subsection are computational analogues of the statistical definitions given in the previous subsection. The following definition appears in [Goldwasser, Micali 82], [Yao 82] and [Goldwasser, Micali, Rackoff 85].

Definition (comp. indistinguishable) : The *distinguishing probability function* $p(n)$ of algorithm M for D and E is $|\Pr[M(x) = 1] - \Pr[M(x') = 1]|$ when $x \in_D \{0,1\}^n$ and $x' \in_E \{0,1\}^n$. D is *computationally indistinguishable* from E if feasible algorithm has negligible distinguishing probability.

The following two propositions are the crucial point in this paper where constructions involving the uniform and non-uniform models of security diverge. Although analogs of each other, these two propositions have a subtle difference in addition to the notions of security involved. Both say, intuitively, that, if two ensembles are computationally indistinguishable,

then many samples from one ensemble are indistinguishable from many samples of the other. If the ensembles in question are both *polynomial-samplable*, then Proposition 2.3 says this is true with respect to both models of security. However, Proposition 2.4 says that in the non-uniform model it is also true for *arbitrary* ensembles. The reason for this difference is as follows. In the uniform model of security, receiving several samples from an ensemble that is not polynomial-samplable might give an adversary information that it could not compute itself. This extra information might allow the adversary to be able to distinguish between the two ensembles. However, this kind of information can never be helpful to a non-uniform adversary, since it is succinctly describable (being extracted from a polynomial number of polynomial length samples) and hence could be included in the non-uniform "advice" string. This distinction has repercussions throughout the paper and is ultimately the reason why our proof of Theorem A does not hold in the uniform model of security. The following propositions appear in [Goldwasser, Micali, Rackoff 85]. Let $q(n)$ be a length function.

Proposition 2.3 : If D and E are *polynomial-samplable* probability ensembles that are computationally indistinguishable (in either model of security). then D^q and E^q are computationally indistinguishable (in the same model of security as before).

Proposition 2.4 : If D and E are *arbitrary* probability ensembles that are computationally indistinguishable in the *non-uniform* model of security then D^q and E^q are computationally indistinguishable in the non-uniform model of security.

The concept of a universal hash function, introduced in [Carter, Wegman], has proved to have far reaching and a broad spectrum of applications in the theory of computation.

Definition (hash functions) : Let $H_{n,m}$ be a family of functions from n bit strings to m bit strings. We say $H_{n,m}$ is a family of *pairwise independent universal hash functions* if, for all $x, y \in \{0,1\}^n$, $x \neq y$, $h(x) \circ h(y) \in_U \{0,1\}^{2m}$ when $h \in_U H_{n,m}$. A *system of hash functions* consists of one such family for all pairs n and m. The following system of pairwise independent universal hash functions has several nice properties. Let $H_{n,m}$ be the set of all m by $n+1$ matrices over the field with two elements. We think of a hash function from this system as $h = (M, b)$, where M is an m by n bit matrix and b is a bit vector of length m. Then, $h(x) = (M \cdot x) \oplus b$. We can choose $h \in_U H_{n,m}$ by choosing $h \in_U \{0,1\}^{(n+1)m}$. Hereafter, whenever we refer to a family or system of hash functions, we mean the family defined here.

3 Combinatorial Lemmas

Due to its importance in such basic algorithms as primality testing, randomness has become an interesting computational resource in its own right. Recently, various studies for extracting good random bits from biased "slightly-random" sources that nevertheless possess a certain amount of entropy have been made; these sources model the imperfect physical sources of randomness, such as Geiger counter noise and Zener diodes, that would have to actually be utilized in real life. (See [Blum 84], [Santha, Vazirani 84], [Vazirani 85], [Vazirani, Vazirani 85], [Chor, Goldreich 85] [McInnes 87].)

The following lemma is very useful in many of our constructions of various kinds of one-way functions and pseudo-random generators. However, it is probably best thought of as a result in the theory of slightly-randomness, rather than cryptography. Intuitively, it can be thought of as a method for extracting "good" random bits from a slightly-random source using real random bits as a "catalyst". In more detail, the various components of the lemma should be interpreted as follows. Suppose we have a slightly-random source that yields a distribution on strings of length n with min-entropy greater than m. A fair coin is used to generate a random hash function mapping n bits to $m-4e$ bits, where e is a small integer. We then sample from the slightly-random source and apply our hash function to the result. The lemma states that the resulting bits are essentially randomly distributed and almost uncorrelated with the bits used to generate the hash function. Thus, we have managed to convert almost all the entropy of the slightly-random source into uniform random bits while maintaining our original supply of uniform random bits. Previously, [McInnes 87] proved a related lemma.

In this extended abstract, we prove a version of this lemma that suffices for the purposes of subsequent constructions. Many generalizations are possible, including much weaker restrictions on the hash functions used, and the substitution of Renyi entropy for min-entropy (see [Renyi 70]). These generalizations will appear in the full paper; some of them can be

used to make our constructions more efficient.

Lemma 3.1 (smoothing min-entropy) : Let D_n have min-entropy at least m and let $l = m - 4e$. Then the distribution $h \circ h(x)$, where $h \in_U H_{n,l}$ and $x \in_D \{0,1\}^n$, is quasi-random within $\delta = 3/2^e$.

Proof : We need to show that *no* statistical test can distinguish between $h \circ h(x)$ and a random string of length $|h| + l$ with probability greater than δ. Each $h \in H_{n,l}$ partitions $\{0,1\}^n$ into 2^l equivalence classes, where class i is $X_i(h) = \{x \in \{0,1\}^n : h(x) = i\}$. Let $Y_i(h) = D[X_i(h)]$. In the following, probabilities and expected values are with respect to $h \in_U H_{n,l}$. By the properties of $H_{n,l}$, $\mathrm{Exp}[Y_i(h)] = 1/2^l$. Let $disc_i(h) = 1/2^l - Y_i(h)$. Let $small(h) = \{i \in \{0,1\}^l : disc_i(h) > 0\}$, i.e. the set of i such that $Y_i(h)$ is smaller than average. Let $p(h)$ be the distinguishing probability for a statistical test. Each input $h \circ i$ to the test such that the output is 1 adds exactly $disc_i(h)$ to $p(h)$. From this, it is clear that the *best* test is when the output is 1 for all $i \in small(h)$, in which case $p(h) = \sum_{i \in small(h)} disc_i(h)$. Let $toosmall(h) = \{i \in \{0,1\}^l : disc_i(h) > 1/2^{l+e}\}$. Using Chebychev's Inequality, the pairwise independence properties of $H_{n,l}$ and the fact that min-entropy of D_n is at least m, it is straightforward to show that $\Pr[i \in toosmall(h)] \leq 1/2^{2e}$ for each fixed $i \in \{0,1\}^l$. We say h *is bad* if $|toosmall(h)| \geq 2^l/2^e$. From the bound on $\Pr[i \in toosmall(h)]$ it is easy to see that $\Pr[h \text{ is bad}] \leq 1/2^e$ using Markov's Inequality. For every h, $p(h) \leq 1$. For each good h, $p(h) \leq |toosmall(h)|/2^l + 2^l/2^{l+e} \leq 2/2^e$. Thus, the overall distinguishing probability for the best test is at most $3/2^e$. □

Can we replace the condition that D has high min-entropy in Lemma 3.1 by the weaker and more natural condition that D have high entropy in the usual (Shannon) sense? Not directly. For example, a distribution can have high Shannon entropy yet still have one element output with probability 1/2; thus, any function computed based on *one* sample from this distribution generates some output with probability at least 1/2, and therefore is highly non-random. This problem hints at a partial solution: take multiple independent samples from the distribution.

Lemma 3.2 (entropy → min-entropy) : Let $k(n)$ be a length function. For every probability ensemble D there is a probability ensemble E with length function $n \cdot k(n)$ satisfying:

- The min-entropy of E_n is at least $k(n) \cdot Ent(D_n) - n \cdot k(n)^{2/3} - k(n) \cdot 2^{-n}$.
- E_n is statistically indistinguishable from D_n^k within $2^{-k(n)^\epsilon} + k(n) \cdot 2^{-n}$ for a fixed $\epsilon > 0$.

Proof Sketch : Let $Y \subseteq \{0,1\}^n$ be the set of elements with probability at least 2^{-2n} with respect to D, and let $Y' = \{0,1\}^n - Y$. Let D'_n be the distribution on $\{0,1\}^n$ described as: (1) for all $x \in Y$, $D'[\{x\}] = D[\{x\}]/D[Y]$; (2) for all $x \in Y'$, $D'[\{x\}] = 0$. It is easy to show that $D[Y'] \leq 2^{-n}$, and from this it follows that $Ent(D'_n) \geq Ent(D_n) - 2^{-n}$.

Consider the random variable X with values in the interval $[0, 2n]$ defined by $X(d) = -\log(D'[\{d\}])$, where $d \in \{0,1\}^n$. When $d_i \in_{D'} \{0,1\}^n$ independently for $i = 1, \ldots, k(n)$, $Y = \sum_{i=1,\ldots,k(n)} X(d_i)$ is the sum of independent random variables on the interval $[0, 2n]$ with expected value $Ent(D'_n)$. Hence, by an elementary extension of Chernoff bounds, with exponentially high probability (in $k(n)$) Y has value within an additive factor of $n \cdot k(n)^{2/3}$ of its expectation. Thus, with exponentially high probability, $Y > k(n) \cdot Ent(D'_n) - n \cdot k(n)^{2/3}$. This means that only with exponentially small probability the sequence $d_1, \ldots, d_{k(n)}$ has probability greater than $2^{-k(n) \cdot Ent(D'_n) + n \cdot k(n)^{2/3}}$. Restricting D'^k_n to the complement of this exponentially small in probability set of sequences, and renormalizing the distribution as before, we obtain E_n. □

Corollary 3.3: Let $k(n) = n^c$ for any constant $c > 0$. Let h be uniformly and randomly chosen from $H_{n \cdot k(n), k(n) \cdot Ent(D) - 2n \cdot k(n)^{2/3}}$, and let $d_i \in_U \{0,1\}^n$ independently for $i = 1, \ldots, k(n)$. Then the distribution $h \circ h(d_1 \circ \ldots \circ d_{k(n)})$ is quasi-random within an exponentially small in n amount.

Proof : Combine Lemma 3.1 and Lemma 3.2. □

4 Computational Entropy

Intuitively, pseudo-random generators transform a small amount of randomness into a much larger string that is random for all practical purposes. Of course, in information-theoretic terms, no such increase is possible. Applying a fixed function to a string can only decrease its informational content, not increase it. (Formally, if D is a distribution and f a function on the same finite set, the Shannon entropy of D is at least as large as that of the induced distribution

$f(d)$, where d is chosen according to D.) Thus, in some sense, any pseudo-random generator determines an ensemble that (asymptotically) has a greater *computational* entropy than it has Shannon entropy. In this section, we formalize this intuitive notion of the computational entropy of an ensemble. This definition provides one of the main conceptual tools in our paper.

Just as Shannon entropy quantifies the amount of randomness in a distribution, computational entropy quantifies the amount of "apparent" randomness (to feasible algorithms) of a distribution. For example, we can relax the notion of a generator f being pseudo-random by allowing its output to be computationally indistinguishable from an ensemble D that is not necessarily the uniform ensemble, where D has more Shannon entropy than the input to f. We call such an f a *pseudo-entropy generator*. In this case, we say that the computational entropy of f is at least the Shannon entropy of D.

The notion of computational entropy is also useful in the case when the Shannon entropy of D is not necessarily greater than that of the input to f. We say that f has *false* entropy if the computational entropy of f exceeds the Shannon entropy of f. (but not necessarily the Shannon entropy of the input to f).

We use computational entropy in constructions of pseudo-random generators as follows. The first step is to show how to use a pseudo-entropy generator to construct a pseudo-random generator. The next step is to show how to convert any function with false entropy into a pseudo-entropy generator.

We obtain Theorem C as a direct consequence of these constructions. In Section 5, we prove that any one-way function in the non-uniform sense can be used to construct a function with false entropy in the non-uniform sense, thus completing the proof of Theorem A.

In the following, $s(n)$ is a function from N to positive reals and the probability ensemble for the inputs of f is U.

Definition (uniform computational entropy) : We say f has *uniform computational entropy at least* $s(n)$ if there is a *polynomial-samplable* ensemble D such that D is computationally indistinguishable (in the uniform sense) from $f(U)$ and $Ent(D_n) \geq s(n)$.

Definition (non-uniform computational entropy) : We say f has *non-uniform computational entropy at least* $s(n)$ if there is an (*arbitrary*) ensemble D such that D is computationally indistinguishable (in the non-uniform sense) from $f(U)$ and $Ent(D_n) \geq s(n)$.

The difference between these two definitions is necessary so that the following proposition holds in both models of security.

Proposition 4.1 (additivity of computational entropy) : Let $q(n)$ be a length function. If f has computational entropy at least $s(n)$ then f^q has computational entropy at least $s(n) \cdot q(n)$.

Proof : The uniform case follows from Proposition 2.3 and the non-uniform case follows from Proposition 2.4. □

Definition (pseudo-entropy generator) : We say f is a *pseudo-entropy generator* if there is a constant $c > 0$ such that f has computational entropy at least $n + 1/n^c$.

Definition (false entropy) : We say f has *false entropy* if there is a constant $c > 0$ such that f has computational entropy at least $Ent(f(U)) + 1/n^c$.

Lemma 4.2 (pseudo-entropy → pseudo-random) : If there is a pseudo-entropy generator then there is a pseudo-random generator.

Proof : Let f be the (uniform/non-uniform) pseudo-entropy generator, and let D be the (polynomial-samplable/arbitrary) probability ensemble with $Ent(D_n) \geq n + n^{-c}$ that is computationally indistinguishable from $f(U)$ (with length $l(n)$). Let $k(n) = n^{3c+2}$. By Proposition 2.3/2.4, $f^k(U)$ is computationally indistinguishable from D^k. Note that $Ent(D_n^k) = k(n) \cdot Ent(D_n) \geq k(n) \cdot (n + n^{-c})$. Let $j(n) = k(n) \cdot (n + n^{-c}) - 2n \cdot k(n)^{2/3}$. Let $h \in_U H_{k(n) \cdot l(n), j(n)}$. Let D'_n be the probability distribution defined by $h \circ h(y_1 \circ \ldots \circ y_{k(n)})$, where, for all $i = 1, \ldots, k(n)$, $y_i \in_D \{0,1\}^{l(n)}$ independently. By Corollary 3.3, D'_n is quasi-random within an exponentially small in n amount. Let the generator be defined by $g(h \circ x_1 \circ \ldots \circ x_{k(n)}) = h \circ h(f(x_1) \circ \ldots \circ f(x_{k(n)}))$. Let E'_n be the probability distribution defined by the output of g when the input is $h \in_U H_{k(n) \cdot l(n), j(n)}$ and, for all $i = 1, \ldots, k(n)$, $x_i \in_U \{0,1\}^n$ independently. Then, since $f^k(U)$ is computationally indistinguishable from D^k, it follows that E' is computationally indistinguishable from D'. Because D' is quasi-random within an exponential small in n amount, and because by choice of $k(n)$ the output of g is longer than the input, g is a pseudo-random generator. □

Our present goal is to transform a function f with false entropy into a pseudo-entropy generator g. The major obstacle is that f could be many-to-one. In this case, even though the output of f seemingly has more entropy than it really has, the Shannon entropy of the output of f may be much less than the length of the input; intuitively the application of f to the input may cause more of a loss in Shannon entropy than the corresponding gain in false entropy. For example, if f is 16-to-1, the probability distribution $f(U_n)$ has $n-4$ bits of Shannon entropy, four bits less than the input length. Even if f has three bits of false entropy, $f(U_n))$ still has one less bit of computational entropy than the input length. We need a method of recovering this loss in Shannon entropy without affecting the false entropy.

We do this in two steps. First, we let $f' = f^q$ for a suitable length function $q(n)$. This has two effects; the false entropy in f' is q times that of f and, for a randomly chosen input x to f', the size of the preimage of $f'(x)$ is, with high probability, relatively close to the expected preimage size.

We then apply the technique of outputting, in addition to $f'(x)$, the output of a randomly chosen hash function applied to x that produces a string of length roughly $|x|$ minus the Shannon entropy of the probability distribution determined by f'. This technique, that we refer to as "hashing the preimages of a function", is also used in proving Theorem A (see Section 5) and in many of the applications in [Impagliazzo, Luby 89]. We now give a highly intuitive presentation of this technique. Let f be a one-way function. Let $rank(x) = |\{y < x : f(y) = f(x)\}|$, i.e. the rank of x among all preimages of $f(x)$. For now, we make the highly unreasonable assumption that $rank(x)$ is easily computable. Consider the function $g(x) = f(x) \circ rank(x)$. $g(x)$ is one-to-one and so $Ent(g(U_n)) = n$, i.e. g has degeneracy zero. Furthermore, the task of inverting x is at least as hard as that of inverting f. On input $f(x) \circ r$, an adversary doesn't just have to find *some* preimage of $f(x)$, it has to be able to find the r^{th} smallest preimage. (Similarly, if f has false entropy, g has at least as much.) $rank(x)$ is not in general computable. However, the value of a random hash function on x frequently serves the same purpose. Let $g(x \circ h) = f(x) \circ h \circ h(x)$. For a fixed value of $f(x)$, and a random hash function h that outputs $\log(|f^{-1}(f(x))|)$ bits, the distribution $h \circ h(x)$ is almost uniform (See Lemma 3.1), and thus could have been generated by the adversary. Consequently, $g(x \circ h)$ is as hard to invert as $f(x)$. On the other hand, x is usually uniquely determined by $f(x) \circ h \circ h(x)$ and thus g has little degeneracy.

Lemma 4.3 and Lemma 4.4 show how to construct a pseudo-entropy generator from a function f with false entropy that satisfies a technical condition: $Ent(f(U_n))$ can be approximated fairly well in time polynomial in n. This condition is not essential; we later sketch a slightly more complicated construction of a pseudo-random generator without assuming this condition for f.

Lemma 4.3 : Consider the probability ensembles D and E defined as follows. Fix and $c > 0$, $k(n) = n^c$ and $j(n) = (n - Ent(f(U_n))) \cdot k(n) - 2n \cdot k(n)^{2/3}$. D_n is given by

$$f(x_1) \circ \ldots \circ f(x_{k(n)}) \circ h \circ h(x_1 \circ \ldots \circ x_{k(n)}),$$

where $x_i \in_U \{0,1\}^n$ independently for $i = 1, \ldots, k(n)$ and where $h \in_U H_{n \cdot k(n), j(n)}$. E is given by

$$f(x_1) \circ \ldots \circ f(x_{k(n)}) \circ r,$$

where the x_i are randomly chosen as above and $r \in_U \{0,1\}^{|h|+j(n)}$. Then, D_n is statistically indistinguishable from E_n within an exponentially small in n amount.

Proof : We claim that, with high probability if, for $i = 1, \ldots, n$, we independently choose $x_i \in_U \{0,1\}^n$ and fix $y_i = f(x_i)$ then the following distribution is quasi-random within an exponentially small in n amount. Let $S_{y_1, \ldots, y_{k(n)}}$ be the set of all sequences $x'_1 \circ \ldots \circ x'_{k(n)}$ where $x'_i \in f^{-1}(y_i)$. Randomly and uniformly choose a sequence $x'_1 \circ \ldots \circ x'_{k(n)} \in S$ and a random h. The distribution is defined as $h \circ h(x'_1 \circ \ldots \circ x'_{k(n)})$. From Lemma 3.1, this distribution is quasi-random within an exponentially small in n amount if the min-entropy of the uniform distribution on $S_{y_1, \ldots, y_{k(n)}}$ is substantially greater than $j(n)$. Define $X(y) = \log(|f^{-1}(y)|)$. Then, the min-entropy of the uniform distribution on $S_{y_1, \ldots, y_{k(n)}}$ is simply

$$\log(|S_{y_1, \ldots, y_{k(n)}}|) = \sum_{i=1, \ldots, k(n)} X(y_i).$$

This is the sum of $k(n)$ independent random samples where the range of each possible value is $[0, n]$. Thus, using Chernoff bounds, the sum is within an additive factor of $n \cdot k(n)^{2/3}$ of its expected value with probability exponentially close to 1. The expected value of $X(y_i)$ when y_i is chosen as described above is $n - Ent(f(U_n))$. Thus, the expected value of the

sum is $(n - Ent(f(U_n))) \cdot k(n)$ from which the claim follows. □

Lemma 4.4 : Assume there is a polynomial-time computable function f with at least n^{-c} bits of false entropy. Assume that we can approximate $Ent(f(U_n))$ to within an additive factor of $n^{-(c+1)}$ in time polynomial in n. Then there is a pseudo-entropy generator g.

Proof : Let $k(n) = n^{3c+2}$, let $a(n)$ be the approximation of $Ent(f(U_n))$ and let $j(n) = (n - a(n)) \cdot k(n) - 2n \cdot k(n)^{2/3}$ Let $g(h \circ x_1 \circ \ldots \circ x_{k(n)}) = f(x_1) \circ \ldots \circ f(x_{k(n)}) \circ h \circ h(x_1 \circ \ldots \circ x_{k(n)})$, where $h \in H_{n \cdot k(n), j(n)}$ and $x_1, \ldots, x_{k(n)} \in \{0,1\}^n$. We claim that $g(U)$ has computational entropy at least $n \cdot k(n) + |h| + 1$. Let D be the probability ensemble that is computationally indistinguishable from $f(U)$ in the definition of false entropy. (In the uniform model of security, D is polynomial-samplable.) Then,

$$Ent(D_n) \geq Ent(f(U_n)) + n^{-c} \geq a(n) + n^{-c} - n^{-(c+1)}.$$

(The last term is the round-off error involved in approximating $Ent(f(U_n))$ by $a(n)$.) From Lemma 4.3, $g(h \circ x_1 \circ \ldots \circ x_{k(n)}) = f(x_1) \circ \ldots \circ f(x_{k(n)}) \circ h \circ h(x_1 \circ \ldots \circ x_{k(n)})$, is statistically indistinguishable from the distribution $g(h \circ x_1 \circ \ldots \circ x_{k(n)}) = f(x_1) \circ \ldots \circ f(x_{k(n)}) \circ r$, when $h \in_U H_{n \cdot k(n), j(n)}$, $r \in_U \{0,1\}^{|h| + j(n)}$ and, for $i = 1, \ldots, k(n)$, $x_i \in_U \{0,1\}^n$, independently. Since D is computationally indistinguishable from $f(U)$, this last ensemble is computationally indistinguishable from $D^k \circ r$ by Proposition 2.3/2.4 (depending on the model of security). $D_n^k \circ r$ has Shannon entropy

$$k(n) \cdot Ent(D_n) + |r| = k(n) \cdot Ent(D_n) + |h| + j(n).$$

Since $Ent(D_n) \geq a(n) + n^{-c} - n^{-(c+1)}$, this quantity is at least $n \cdot k(n) + |h| + 1$ for our choice of $k(n)$. □

Lemma 4.5 (false entropy → prg) : If there is an f that has false entropy at least n^{-c} for some constant $c \geq 0$ then there is a pseudo-random generator.

Proof Sketch : There are only n^{c+1} possible values for $Ent(f(U_n))$ to within an additive factor of n^{-c}. For each of the n^{c+1} possible values of $Ent(f(U_n))$, combining the constructions given in Lemmas 4.4, 4.2, and Proposition 1.2 yields a candidate for a pseudo-random generator stretching a bit string of length n into one of length n^{c+3}. For each of the possible values, we uniformly and randomly choose a string of length n as the input; thus the total length of all of our input is n^{c+2}. At least one of the candidates uses the correct value for $Ent(f(U_n))$ and hence its output is pseudo-random. We "exclusive-or" all n^{c+1} outputs to produce the final output of length n^{c+3}. Because at least one of the outputs from the candidates is pseudo-random, the final output is also pseudo-random. Thus, we have stretched an input of length n^{c+2} to a pseudo-random output of length n^{c+3}. □

Lemma 4.6 (hidden and meaningful bit → false entropy) : If there is a function f that hides a meaningful bit, then there is a function g and a constant $d > 0$ such that g has false entropy at least $1/n^d$.

Proof Sketch : Let b be the hidden and meaningful bit for f and let $c > 0$ be such that an information adversary has distinguishing probability $p(n) > 1/n^c$. Let $g(x) = f(x) \circ b(x)$. Let D be the probability ensemble such that D_n is given by $f(x) \circ \beta$ when $x \in_U \{0,1\}^n$ and $\beta \in_U \{0,1\}$. Since f hides the bit b, $g(U)$ is computationally indistinguishable from D. (Also, D is polynomial-samplable, which is needed in the uniform model of security). Since $b(x)$ has correlation at least $1/n^c$ to $f(x)$ and since β is independent of $f(x)$, it can be shown that $Ent(D_n) \geq Eng(g(U_n)) + 1/16n^{3c}$. □

Theorem C : If there is a bit b that is both hidden and meaningful for f then there is a pseudo-random generator.

Proof : Combine Lemmas 4.6 and 4.5. □

5 Pseudo-Random Generators from One-Way Functions

In the previous section we reduced the problem of constructing a pseudo-random generator to the problem of finding a function with false entropy. In this section we show, in the non-uniform model, how to construct a function g with false entropy from *any* one-way function f. Let D be a probability ensemble that is computationally indistinguishable from $g(U)$. The construction has the property that if there is a (uniform/non-uniform) feasible algorithm for distinguishing D from $g(U)$ then there is a (uniform/non-uniform) feasible algorithm for inverting f. The only problem is that D is not polynomial-samplable. In particular, even if $g(U)$ is computationally indistinguishable from D in the uniform model, we see no way

to prove that $g^q(U)$ is computationally indistinguishable from D^q in the uniform model (see Proposition 2.3).

Lemmas 5.1 (one-way → non-uniform false entropy) : If there is a one-way function f (in the non-uniform sense) then there is a polynomial-time computable function g with false entropy at least $1/3n$ (in the non-uniform sense).

Proof : Let $l(n) = \lceil \log(n+1) \rceil$ and let $k(n) = 2^l(n) - 1$. Define g by

$$g(r \circ i \circ h \circ x) = r \circ i \circ h \circ f(x) \circ (h(x) \uparrow (i + l(n))) \circ r \odot x,$$

where $r, x \in \{0,1\}^n$, $i \in \{0,1\}^{l(n)}$ (i is to be thought of as a number between 0 and $k(n)$) and $h \in H_{n,k(n)+l(n)}$. Let $m(n) = |r \circ i \circ h \circ x|$. Let U'_n be the uniform probability distribution strings of length $m(n)$. We describe a probability ensemble D that is computationally indistinguishable from $g(U')$ such that $Ent(D_n) \geq Ent(g(U'_n)) + 1/3n$. Although D is not going to be polynomial-samplable, it is easiest to think of D as being generated from $r \circ i \circ h \circ x \in_U \{0,1\}^{m(n)}$ and from independently chosen bit $\beta \in_U \{0,1\}$. Define $i(x) = \lfloor \log(|f^{-1}(f(x))|) \rfloor$. $D(r \circ i \circ h \circ x \circ \beta)$ is exactly the same as $g(r \circ i \circ h \circ x)$ except for possibly the last bit, which in g is always $r \odot x$. The last bit of $D(r \circ i \circ h \circ x \circ \beta)$ is $r \odot x$ unless $i = i(x)$, in which case it is β.

Let ensemble E be such that E_n is the distribution on $i(x) \circ h \circ x$ when $x \in_U \{0,1\}^n$ and $h \in_U H_{n,k(n)+l(n)}$. We define

$$g'(i \circ h \circ x) = i \circ h \circ f(x) \circ (h(x) \uparrow (i + l(n))).$$

Claim : g' is one-way on E.

Proof of Claim : Assume there is an algorithm M with time bound $T(n)$ that inverts $g'(i \circ h \circ x)$ with probability at least $p(n) = 1/T(n)$ for some function $T(n)$ in resource class R when $i \circ h \circ x$ is randomly chosen according to E_n, i.e. on input $i(x) \circ h \circ f(x) \circ (h(x) \uparrow (i(x) + l(n)))$, M finds y such that $f(y) = f(x)$ and $h(y) \uparrow (i(x) + l(n)) = h(x) \uparrow (i(x) + l(n))$. We use only the fact that $f(y) = f(x)$ in the proof of the claim. We construct an algorithm M' with time bound polynomial in $T(n)$ and n that inverts $f(x)$ with probability at least $p(n)/2$. Let $j(n) = 8 \log(T(n))$. On input $f(x)$, M' runs through all possible $i = 0, \ldots, k(n)$. For each i, M' chooses $h \in_U H_{n,k(n)+l(n)}$ and $s \in_U \{0,1\}^{i-j(n)}$. For each $t \in \{0,1\}^{j(n)+l(n)}$, M' forms $u = s \circ t$ and simulates M on input $i \circ h \circ f(x) \circ u$. We claim that with probability at least $p(n)/2$ there is a round, i.e. an i and a t, where this simulation yields a y with $f(y) = f(x)$. The round in question is when $i = i(x)$. We know that the probability that M, on input $i(x) \circ h \circ f(x) \circ (h(x) \uparrow (i(x)+l(n)))$, for x and h randomly selected, yields such a y, is at least $p(n)$. If instead of trying the one value $h(x) \uparrow (i(x)+l(n))$ we try $(h(x) \uparrow (i(x) - j(n))) \circ t$ for all possible extensions $t \in \{0,1\}^{j(n)+l(n)}$, the chance that M finds such a y can only increase. Fix $f(x)$ and consider the distribution $h \circ (h(x') \uparrow (i(x) - j(n)))$ for x' randomly and uniformly chosen from $f^{-1}(f(x))$. The distribution on x' has min-entropy $\log(|f^{-1}(f(x))|) \geq i(x)$. Hence, by Lemma 3.1, the distribution $h \circ (h(x') \uparrow (i(x) - j(n)))$ is statistically indistinguishable within $3p(n)^2$ from the distribution $h \circ s$ when $h \in_U H_{n,k(n)+l(n)}$ and $s \in_U \{0,1\}^{i(x)-j(n)}$. Thus the probability that, for at least one t, M on input $i(x) \circ h \circ f(x) \circ s \circ t$ finds a preimage of $f(x)$ for s chosen at random differs from the probability when s is given by $h(x) \uparrow (i(x) - j(n))$ by at most $3p(n)^2$. Thus, this probability for a random s is at least $p(n) - 3p(n)^2 \geq p(n)/2$. From this contradiction of the one-wayness of f, we conclude that g' is one-way on E. **End of Claim Proof**

We now conclude the proof of the lemma. To distinguish D from $g(U')$ is equivalent to being able to predict $r \odot x$, for $i(x) \circ h \circ x$ randomly chosen according to E_n and and $r \in_U \{0,1\}^n$, given $r \circ g'(i(x) \circ h \circ x)$. From the above claim and from Proposition 1.3 it follows that this task is computationally infeasible, and thus $g(U')$ and D are computationally indistinguishable.

All that remains to be shown is that $Ent(D_n) > 1/3n + Ent(g(U'_n))$. This follows from the fact that with probability at least $1 - 1/n$, when $x \in_U \{0,1\}^n$ and $h \in_U H_{n,k(n)+l(n)}$, x is uniquely determined by $f(x)$ and $h(x) \uparrow (i(x) + l(n))$. When x is fixed and $i \in_U \{0,1\}^{l(n)}$, $i = i(x)$ with probability at least $1/2n$. Thus, with probability at least $(1 - 1/n)/2n \geq 1/3n$ the last bit of $g(r \circ i \circ h \circ x)$ is completely determined by the preceding bits, whereas in $D(r \circ i \circ h \circ x \circ \beta)$ the last bit β is always independent of the preceding bits, and thus there is one extra bit of entropy in D_n. Furthermore, in all cases, $r \odot x$ adds at most one bit of entropy to the preceding bits of $g(r \circ i \circ h \circ x)$. □

Theorem A : If there is a one-way function in the non-uniform model of security then there is a pseudo-

random generator in the non-uniform model of security.

Proof : Combine Lemma 5.1 and Lemma 4.5. □

6 Open Problems

The results presented in this paper unify different concepts in theoretical cryptography. When combined with other work ([Goldreich, Goldwasser, Micali 84], [Luby, Rackoff 86], [Goldreich, Micali, Wigderson 86], [Naor 88]), they show that applications ranging from private key encryption to zero-knowledge proofs can be based on any one-way function. [Impagliazzo, Luby 89] shows that most cryptographic applications that are impossible in a world where anything that is informationally possible is computationally possible must be implicitly based on a one-way function.

Several very interesting open questions remain. We have shown that in the non-uniform model of security any one-way function yields a pseudo-random generator. Is this also true in the uniform model of security?

A more general problem is to characterize the conditions under which cryptographic applications are possible. Although there are characterizations for some applications in this paper combined with [Impagliazzo, Luby 89], many others remain open. [Naor, Yung 89] give a signature scheme that can be based on any one-way permutation. Can this assumption be weakened to give a signature scheme based on any one-way function? Some applications seem unlikely to be shown possible based on any one-way function, e.g. [Impagliazzo, Rudich 89] give strong evidence that secret exchange is an application of this kind.

Another important issue is that of efficiency; the construction we give here for a pseudo-random generator based on any one-way function increases the size of the seed by a large polynomial amount. For practical applications, it would be nice to have a much more parsimonious construction. We would like to develop constructions that are more efficient than the existing ones, with the goal being a private key cryptosystem based on the intractability of some natural problem that is as fast as any cryptosystem used in practice.

7 Acknowledgements

This research evolved over a long period of time and was greatly influenced by many people. We thank Amos Fiat, Moni Naor, Ronitt Rubinfeld, Manuel Blum, Steven Rudich, Noam Nisan, Lance Fortnow, Umesh Vazirani, Charlie Rackoff, Oded Goldreich and Hugo Krawczyk for their insights and contributions to this work. We in particular thank Charlie, Umesh and Manuel for their advice and enthusiasm, Lance for suggesting a version of Lemma 3.2 and Oded and Hugo for introducing the third author to this question.

References

[1] Blum, M., "Independent Unbiased Coin Flips From a Correlated Biased Source: A Finite State Markov Chain", 25^{th} *FOCS*, 1984, pp. 425-433.

[2] Blum, M., and Micali, S., "How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits", *SIAM J. on Computing*, Vol. 13, 1984, pp. 850-864, *FOCS 1982.*

[3] Carter, J., and M. Wegman, "Universal Classes of Hash Functions", *JCSS*, 1979, Vol. 18, pp. 143-154.

[4] Chor, B., and O. Goldreich, "Unbiased Bits from Sources of Weak Randomness and Probabilistic Communication Complexity", *SIAM J. on Computing*, Vol. 17, 1988, *FOCS 1985.*

[5] Goldreich, O., S. Goldwasser, and S. Micali, "How to Construct Random Functions", *J. of ACM*, Vol. 33, No. 4, 1986, pp. 792-807, *FOCS 1984.*

[6] Goldreich, O., Krawczyk, H. and Luby, M., "On the Existence of Pseudorandom Generators", 29^{th} *FOCS*, 1988, pp. 12-24.

[7] Goldreich, O., and L.A. Levin, "A Hard-Core Predicate for any One-Way Function", these proceedings.

[8] Goldreich, O., Micali, M. and Wigderson, A., "Proofs that Yield Nothing but Their Validity and a Methodology of Cryptographic Protocol Design", 27^{th} *FOCS*, 1986, pp. 174-187, *Tech. Report TR498*, Comp. Sci. Dept., Technion, submitted to *JACM.*

[9] Goldwasser, S. and Micali, S., "Probabilistic Encryption," *JCSS*, Vol. 28, No. 2, April 1984, pp. 270-299, *STOC 1982.*

[10] Goldwasser, S., Micali, S. and Rackoff, C., "The Knowledge Complexity of Interactive Proof Systems," *SIAM J. on Computing*, Vol. 18, No. 1, 1989, pp. 186-208, *STOC 1985.*

[11] Impagliazzo, R. and Luby, M., "One-way functions are essential for information based cryptography," submitted to *CRYPTO 1989.*

[12] Impagliazzo, R. and Rudich, S., "Limits on the Provable Consequences of One-way Functions", these proceedings

[13] Karp, R., Lipton, R., "Turing Machines that Take Advice," *L'Enseignement Mathematique*, Vol. 28, pp. 191-209 (1982), *STOC 1980*

[14] Levin, L.A., "One-Way Function and Pseudorandom Generators", *Combinatorica*, Vol. 7, No. 4, 1987, pp. 357-363, *STOC 1985.*

[15] Luby M., and Rackoff, C., "How to Construct Pseudorandom Permutations From Pseudorandom Functions", *SIAM J. on Computing*, Vol. 17, 1988, pp. 373-386, *STOC 1986*

[16] McInnes, J., "Cryptography Using Weak Sources of Randomness," *Tech. Report 194/87*, U. of Toronto, 1987.

[17] Naor, M., personal communication, 1988.

[18] Naor, M. and Yung, M., "Universal One-way Hash Functions and Their Applications", these proceedings.

[19] Renyi, A., **Probability Theory**, North-Holland, Amsterdam, 1970.

[20] Shannon, C., "A Mathematical Theory of Communication", *Bell Systems Technical Journal*, 27, 1948, pp. 379-423 and pp. 623-656.

[21] Santha, M. and Vazirani, U., "Generating Quasi-random Sequences from Slightly-random Sources", 25^{th} *FOCS*, 1984, pp. 434-440, *JCSS*, Vol. 33, No. 1, 1986.

[22] Vazirani, U., "Towards a Strong Communication Complexity Theory or Generating Quasi-random Sequences from Two Communicating Slightly-random Sources", 17^{th} *STOC*, 1985, pp. 366-378, *Combinatorica*, Vol. 7, No.4, 1987.

[23] Vazirani, U. and Vazirani, V., "Random Polynomial Time is Equal to Slightly-random Polynomial Time", 26^{th} *FOCS*, 1985, pp. 417-428, submitted to *JACM.*

[24] Yao, A.C., "Theory and Applications of Trapdoor Functions", 23^{rd} *FOCS*, 1982, pp. 80-91.