Pseudo-Random Number Generation
From ANY One-Way Function

(Preliminary Version)

Russell Impagliazzo$^1$ and Mike Luby$^2$

TR-89-002
February, 1989

We construct a pseudo-random number generator from ANY one-way function. Previous results show how to construct pseudo-random number generators from one-way functions that have special properties (Blum and Micali [BM], Yao [Y], Levin [L1], [Goldreich, Krawczyk and Luby [GKL]]). We use techniques borrowed from the theory of slightly-random sources (Santha and Vazirani [SV], Vazirani and Vazirani [VV], Vazirani [V], Chor and Goldreich [CGJ]) and from the theory of universal hash functions (Carter and Wegman [CW]).

We also introduce a weaker kind of one-way function, that we call an informationally one-way function. For an informationally one-way function $f$, given $y = f(x)$ for a randomly chosen $x$, it is hard to generate uniformly a random preimage of $y$. We show that the existence of an informationally one-way function yields a one-way function in the usual sense, and hence a pseudo-random number generator. These results can be combined to show that the following are equivalent: (1) private key encryption; (2) bit commitment; (3) pseudo-random number generators; (4) one-way functions; (5) informationally one-way functions.

$^1$Department of Mathematics, University of California, Berkeley, research partially supported by NSF grant CCR 88-13632 - russell@ernie.berkeley.edu

$^2$International Computer Science Institute, Berkeley, California, on leave of absence from the University of Toronto, research partially supported by NSERC operating grant A8092 - luby@icsi.berkeley.edu
Pseudo-random number generation from
ANY
one-way function
(Preliminary Version)

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

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

January 18, 1989

Abstract

We construct a pseudo-random number generator from ANY one-way function. Previous results show how to construct pseudo-random number generators from one-way functions that have special properties (Blum and Micali [BM], Yao [Y], Levin [L1], Goldreich, Krawczyk and Luby [GKL]). We use techniques borrowed from the theory of slightly-random sources (Santha and Vazirani [SV], Vazirani and Vazirani [VV], Vazirani [V], Chor and Goldreich [CG]) and from the theory of universal hash functions (Carter and Wegman [CW]).

We also introduce a weaker kind of one-way function, that we call an informationally one-way function. For an informationally one-way function $f$, given $y = f(x)$ for a randomly chosen $x$, it is hard to generate uniformly a random preimage of $y$. We show that the existence of an informationally one-way function yields a one-way function in the usual sense, and hence a pseudo-random number generator. These results can be combined to show that the following are equivalent: (1) private key encryption; (2)

*Research partially supported by NSF grant CCR 88-13632
†On leave of absence from the University of Toronto, research partially supported by NSERC operating grant A8092
bit commitment; (3) pseudo-random number generators; (4) one-way functions; (5) informationally one-way functions.

1 Introduction

Informally, a number generator is a function that takes as input a string called the seed and outputs a string at least one bit longer than the seed. The number generator is pseudo-random if, when the seed is chosen randomly, the output is computationally indistinguishable from a truly random string. Pseudo-random number generation has become a basic tool in cryptography and in computational complexity theory. In cryptography, one of their main uses is for private key encryption. In private key encryption, two parties who share a small, secret, random string (called the private key) can securely send each other messages of length much greater than the length of the key over a public channel. Luby and Rackoff [LR], building on the results of Goldreich, Goldwasser, and Micali [GGM], show how to use a pseudo-random number generator to create a very strong form of private-key system, called a pseudo-random block cipher. Pseudo-random generators have many other applications in cryptography as well. For example, Naor [N] shows how to construct a bit commitment protocol based on a pseudo-random number generator. Under the assumption that bit commitment is possible, Goldreich, Micali and Wigderson [GMW] show that any problem in \( \text{NP} \) has a zero-knowledge proof system. In the area of computational complexity, Yao [Y] shows that the existence of pseudo-random number generators implies that \( \text{BPP} \subset \text{DTIME}(2^{n^\varepsilon}) \) for every \( \varepsilon > 0 \).

The first construction of a pseudo-random number generator, due to Blum and Micali [BM], is based on the intractability of the discrete log problem. Yao [Y] generalizes this by showing that a pseudo-random number generator can be constructed from any one-way permutation. These results are further generalized by Levin [L1] who shows that any one-way function which has the additional property that it is one-way on a polynomial number of iterates can be used to construct a pseudo-random number generator. Goldreich, Krawczyk and Luby [GKL] show that any one-way function for which the preimage sizes of all elements in the range are roughly equal can be used for this purpose. (The actual condition is slightly weaker.) All of the constructions mentioned above base the construction of a pseudo-random number generator on a one-way function with special properties. Our main result removes all such restrictions; we construct a pseudo-random number generator from any one-way function.

Intuitively, a one-way function is a function which is computable in polynomial time but inversion of the function on a random input is practically impossible. When the function in question is a permutation the meaning of inverting the function is unambiguous: Given \( f(x) \) the inverting algorithm must find \( x \). When \( f \) is many-to-one the task of the inverting algorithm is less clear. For example, if \( f(x) = x' \) where \( x' \) is all but the last bit of \( x \),
there is no way any algorithm given \( f(x) \) can find \( x \) with probability greater than 1/2. The standard definition of a one-way function handles this problem by having the inverter succeed if it finds any \( y \) such that \( f(y) = f(x) \). However, this allows inverters that might never guess the specific \( x \) that was chosen to generate \( f(x) \). For example, let \( g(x) \) be any one-way permutation and define \( f(x, y) \) to be \( x \) if \( y = 0 \) and \( g(x) \) otherwise. Then, the algorithm that on input \( z = f(x, y) \) outputs \((z, 0)\) always finds some inverse of \( f \) but a very atypical one. The \( f \) defined in this example intuitively has a lot of “one-wayness” built into it (via \( g \)) although it is not at all one-way by the standard definition. Thus, we would like to define a weaker notion of one-way function that captures the “one-wayness” of this \( f \). This motivates the following informal definition: \( f \) is informationally one-way if, given \( f(x) \), it is computationally infeasible to randomly and uniformly generate a preimage of \( f(x) \). We show that the existence of even such a weakly one-way function can be used to construct a one-way function in the standard sense, and hence a pseudo-random number generator. This definition is particularly natural since it is relatively easy to show that any protocol for either of the two basic cryptographic problems of private key encryption or bit commitment yields an informationally one-way function. Thus we can sum up our results as stating that the following are all equivalent: (1) private key encryption; (2) bit commitment; (3) pseudo-random number generators; (4) one-way functions; (5) informationally one-way functions.

2 Definitions and Notation

Notation: We frequently need to think of \( n \) bit strings as vectors over the field with two elements. In these contexts, \( x \oplus y \) denotes the vector sum mod 2 (i.e. bitwise parity) of \( x \) and \( y \), \( x \cdot y \) denotes the inner product mod 2 (i.e. \( x \cdot y = 1 \) if the number of bit positions where both \( x \) and \( y \) are 1 is odd and 0 otherwise) and \( x \| y \) is the concatenation of \( x \) and \( y \). Let \( x_i \) be the \( i \)th bit of \( x \) and let \( x \uparrow i \) be the first \( i \) bits of \( x \).

2.1 Probability Theory

Notation: Let \( D \) be a probability distribution on a finite set \( S \). For all \( x \in S \), we let \( \Pr_D[x] \) be the probability of \( x \) with respect to \( D \), and for all \( T \subseteq S \), we let \( \Pr_D[T] \) be the sum of the probabilities of all elements of \( T \) with respect to \( D \). If \( x \in T \), we let \( \Pr_D[x|T] = \Pr_D[x]/\Pr_D[T] \) be the conditional probability of \( x \) given that we are choosing an element randomly from \( T \) with respect to \( D \). We let \( x \in_D S \) denote that \( x \) is randomly chosen from \( S \) according to \( D \). When the distribution in question is clear from the context, we use \( \Pr \) in place of \( \Pr_D \).

Definition 2.1: Let \( D \) and \( E \) be probability distributions on a common finite set \( S \). \( D \) and \( E \) are statistically indistinguishable within \( \delta \) if for every \( T \subseteq S \), \(|\Pr_D[T] - \Pr_E[T]| < \delta \).
The following definition is due to Santha and Vazirani [SV].

**Definition 2.2**: Let $D$ be a probability distribution on the set of $m$ bit binary strings. We say $D$ is quasi-random within $\delta$ if $D$ is statistically indistinguishable within $\delta$ from the uniform distribution on $m$ bit strings.

The technique in Yao [Y] provides a useful characterization of quasi-randomness.

**Definition 2.3**: Let $D$ be a distribution on $m$ bit strings. Let $x \in D \{0,1\}^m$. $D$ passes the statistical next bit test within $\delta$ if for every $i = 0, \ldots, m - 1$ and every function $f : \{0,1\}^i \rightarrow \{0,1\}$, the probability that, $f(x \uparrow i) = x_{i+1}$ is at most $1/2 + \delta$.

**Theorem 2.4** [Yao]: If $D$ passes the statistical next bit test within $\delta$ then $D$ is quasi-random within $m\delta$.

Entropy, introduced by Shannon [S], gives a measure of the number of bits of randomness contained in a distribution.

**Definition 2.5**: Let $D$ be a distribution on a finite set $S$. The (Shannon) entropy of $D$ is given by

$$\text{Ent}(D) = -\sum_{x \in S} \Pr_D[x] \cdot \log(\Pr_D[x]).$$

For example, if $D$ is the uniform distribution on $m$ bit strings, then $\text{Ent}(D) = m$. Intuitively, the entropy is approximately the average value, when $x \in_D S$, of the number of bits it takes to express $\Pr_D[x]$ to within a multiplicative factor of 2.

We also need to use a variant definition of entropy introduced by Chor and Goldreich [CG].

**Definition 2.6**: Let $D$ be a distribution on a finite set $S$. The Chor-Goldreich entropy of $D$ is

$$\text{Ent}_{CG}(D) = \min_{x \in S} \{-\log(\Pr[x])\}.$$  

Intuitively, if a distribution has Chor-Goldreich 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 Chor-Goldreich entropy.

### 2.2 Computational Intractability

Some of the definitions given in this subsection are computational analogues of the statistical definitions given in the previous subsection. In order to discuss the asymptotic
complexity of problems, we need to define infinite sequences of objects.

**Definition 2.7:** An ensemble $D$ assigns to each positive integer $n$ a distribution $D_n$ on the set of $n$ bit strings. Let $\text{Ent}(D)$ be the entropy function for $D$, i.e., the function that assigns to each $n$ the real number $\text{Ent}(D_n)$.

**Definition 2.8:** A circuit family $C$ assigns to each positive integer $n$ a boolean circuit $C_n$ with $n$ inputs. $C$ is *poly-size* if there is a polynomial $p$ such that the size (the number of gates) of $C_n$ is less than $p(n)$. Let $x$ be a bit string such that the length of $x$ is the length of the input to $C_n$. We let $C_n(x)$ denote the value of the output of $C_n$ when the input is $x$.

**Definition 2.9:** Two ensembles $D$ and $E$ are *computationally indistinguishable* if for every poly-size circuit family $C$, for every polynomial $q$ and for all but finitely many $n$,

$$\left| \Pr[C_n(x) = 1] - \Pr[C_n(y) = 1] \right| < \frac{1}{q(n)},$$

where $x \in D_n \{0,1\}^n$ and $y \in E_n \{0,1\}^n$.

**Definition 2.10:** The uniform ensemble is the ensemble $U$ that assigns $n$ the uniform distribution $U_n$ on $n$ bit strings.

**Definition 2.11:** An ensemble $D$ is *pseudo-random* if $D$ is computationally indistinguishable from the uniform ensemble $U$.

**Definition 2.12:** A *number generator* is a polynomial time computable function $f$ mapping $n$ bits to $l(n)$ bits, where $l(n) > n$. A number generator defines an ensemble where the distribution assigned to $l(n)$ is defined by choosing $x \in U_n \{0,1\}^n$ and computing $f(x)$. We denote the entropy function of this ensemble by $\text{Ent}(f)$. A number generator is *pseudo-random* if the corresponding ensemble is pseudo-random.

**Definition 2.13:** Let $D$ be an ensemble. $D$ passes the *computational next bit test* if for every poly-size circuit family $C$, for all polynomials $q$, for all but finitely many $n$ and for every $i = 0, \ldots, n - 1$,

$$\Pr[C_{n,i}(x \uparrow i) = x_{i+1}] < \frac{1}{2} + \frac{1}{q(n)},$$

when $x \in D_n \{0,1\}^n$.

**Theorem 2.14 (Yao [Y]):** An ensemble is pseudo-random iff it passes the computational next bit test.

**Theorem 2.15 (Goldreich and Micali [GM]):** If there is a pseudo-random number generator with $l(n) = n + 1$ then for every polynomial $q$ there is a pseudo-random number generator with $l(n) = q(n)$.  

5
Definition 2.16: Let $D$ be an ensemble and $q$ be a polynomial. Let $D^q$ be the ensemble that assigns to positive integer $mq(n)$ the distribution obtained by independently sampling $q(n)$ times from $D_n$ and concatenating the results.

Theorem 2.17 (Yao [Y]): Let $D$ and $E$ be computationally indistinguishable ensembles and let $q$ be a polynomial. Then, $D^q$ and $E^q$ are computationally indistinguishable.

NOTE: All along, we have been defining the complexity of problems in terms of poly-size circuits. Similar definitions in terms of probabilistic polynomial time Turing machines can be made. Theorem 2.17 holds in the circuit model even if $D$ and $E$ cannot be generated in polynomial time. However, Theorem 2.17 does not seem to hold in this case in the Turing machine model. This seems to be crucial for our result.

Definition 2.18: Let $\delta$ be a function from positive integers into the interval $[0, 1]$. We say $f$ is $\delta$-one-way if $f$ is polynomial time computable and for every poly-size circuit family $C$, for all but finitely many $n$,

$$ \Pr[f(C_n(f(x))) = f(x)] < \delta(n), $$

where $x \in U_n \{0, 1\}^n$. We say $f$ is weakly one-way if $f$ is $\delta$-one-way for $\delta = 1 - n^{-c}$ for some constant $c > 0$. We say $f$ is strongly one-way if $f$ is $\delta$-one-way for $\delta = n^{-c}$ for every constant $c > 0$.

Let $f$ be a $1 - n^{-c}$-one-way function. Define

$$ g(x_1, \ldots, x_{n+c}) = f(x_1) \cdots f(x_{n+c}), $$

where $x_1, \ldots, x_{n+c}$ are $n$ bit strings. The following theorem is implicitly used by Yao [Y].

Theorem 2.19: The function $g$ just described in terms of $f$ is a strongly one-way function.

2.3 Carter-Wegman Universal Hash Functions

The concept of a universal hash function, introduced by Carter and Wegman [CW], has proved to have far reaching and a broad spectrum of applications in the theory of computation. We utilize their properties fully in this paper.

Definition 2.20: 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 every pair of distinct $n$ bit strings $x$ and $y$, the distribution $(h(x), h(y))$, for $h$ randomly selected uniformly from $H_{n,m}$, is the uniform distribution on pairs of $m$ bit strings. A system of hash functions consists of one such family for all pairs $n$ and $m$, together with a 1-1 map from $H_{n,m}$ to the integers from 0 to $|H_{n,m}|-1$. (Thus, we can think of elements of $H_{n,m}$ as being indexed by integers. Hereafter, we use $h$ to represent both the hash function and
the index of the hash function.) A system \( H_{n,m} \) is nice if there is a polynomial time (in \( n \) and \( m \)) computable function \( l(n,m) \) so that \( |H_{n,m}| = 2^{l(n,m)} \) (i.e., the elements of \( H_{n,m} \) are in a 1-1 correspondence with the strings of length \( l(n,m) \) ), and a polynomial time computable function that, on input \( h \in H_{n,m} \) and \( x \in \{0,1\}^n \), computes \( h(x) \).

Comment : Nice systems of pairwise independent universal hash functions do exist. For example, 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 \( n \) matrix and \( b \) is a vector of length \( m \). Then, \( h(x) = Mx \oplus b \). Here, \( l(n,m) = (n+1)m \). We can randomly and uniformly generate \( h \in H_{n,m} \) by choosing \( h \in U_{(n+1)m} \{0,1\}^{(n+1)m} \). Hereafter, whenever we refer to a family or system of hash functions, we mean the family discussed in this comment.

3 Slightly-Randomness and Pseudo-Randomness

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 Xener diodes, that would have to actually be utilized in real life. (See Blum [B], Santha and Vazirani [SV], Vazirani [V], Vazirani and Vazirani [VV], and Chor and Goldreich [CG].)

We care about the following lemma because it is very useful in the proofs of our constructions. However, it is probably best thought of as a result in the theory of slightly-randomness. 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 Chor-Goldreich entropy greater than \( m \). A fair coin is used to generate a random hash function mapping \( n \) bits to \( m - 6e \) 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.

**Lemma 3.1** : Let \( D \) be a distribution with Chor-Goldreich entropy at least \( m \) on strings of length \( n \). Let \( H = H_{n,m-6e-1} \). Then the distribution \( h \cdot h(x) \), where \( h \) is randomly and uniformly chosen from \( H \) and \( x \in D \{0,1\}^n \), is quasi-random within \( \delta = \frac{2m}{n} \).

**Proof** : We need to show that NO statistical test can distinguish between \( h \cdot h(x) \) and a random string of length \( |h| + m - 6e \) (regardless of computing power) with probability
greater than $\delta$. The first $|h|$ bits are truly random. By Theorem 2.4, it suffices to show that any method of predicting any of the last $m - 6e - 1$ bits from the preceding bits succeeds with probability at most $\frac{1}{2} + \frac{\delta}{m-6e-1} \leq \frac{1}{2} + \frac{\delta}{2e}$. It is sufficient to prove this for only the last bit, because predicting an earlier bit from all the previous bits can be thought of as an instance of the same problem with a larger value of $e$.

Let $l = m - 6e - 2$. From $h$ and the first $l$ bits of $h(x)$, our adversary would like to predict the last bit of $h(x)$. Since our adversary has unlimited computational power, there is in fact a single best algorithm to use. The (known) hash function $h$ partitions $\{0,1\}^n$ into $2^l$ components, according to the first $l$ bits of $h(x)$. The adversary is given $h$ and told the component $C$ in which $x$ lies. The best strategy to predict the last bit of $h(x)$ is to split $C$ into two parts $C_0$ and $C_1$, where $C_0$ is the set of all strings $y \in C$ such that the last bit of $h(y)$ is 0 and $C_1$ is the set of all strings $y \in C$ such that the last bit of $h(y)$ is 1. She then chooses the bit that corresponds to the larger of $C_0$ and $C_1$ as her prediction. The probability of success is exactly $\frac{\max \{\Pr(C_0), \Pr(C_1)\}}{\Pr(C)}$. Define the discrepancy of $C$ as

$$disc(C) = \frac{|\Pr(C_0) - \Pr(C_1)|}{\Pr(C)}.$$

Then the probability of success is exactly $\frac{1}{2}(1 + disc(C))$. Thus, we need to show that with high probability, if we pick $h$ and $x$ at random (uniformly, and according to $D$, respectively), the component $C$ that we land in is almost evenly divided between $C_0$ and $C_1$, i.e. has small discrepancy. To get our desired bound of $\frac{1}{2} + \frac{\delta}{2e}$, it suffices to show that $disc(C) < \frac{\delta}{2e}$ with probability at least $1 - \frac{1}{2e}$.

To allow calculation, instead of dealing directly with the discrepancy, we deal with the following probability. For a fixed $C$, randomly choose $x$ and $y$ independently from $C$ according to $D$. The correlation of $C$ is the probability that the last bit of $h(x)$ and the last bit of $h(y)$ are the same, i.e.

$$corr(C) = \left(\frac{1}{2}(1 + disc(C))\right)^2 + \left(\frac{1}{2}(1 - disc(C))\right)^2 = \frac{1}{2} + \frac{1}{2}disc(C)^2,$$

so $disc(C) < \frac{1}{2e}$ iff $corr(C) < \frac{1}{2} \left(1 + \frac{1}{2e}\right)$. Thus, we wish to show that when $h$ and $x$ are chosen randomly

$$\Pr\left[corr(C) < \frac{1}{2} \left(1 + \frac{1}{2e}\right)\right] \geq 1 - \frac{1}{2e}.$$

For every $C$, $corr(C) \geq \frac{1}{2}$. Therefore, in order to show that the probability $corr(C)$ is significantly bigger than $1/2$ is small, it suffices to show that the expected value of $corr(C)$ is close to $1/2$. If the probability that $corr(C) > \frac{1}{2} \left(1 + \frac{1}{2e}\right)$ were larger than $\frac{1}{2e}$, we would have the expected value being at least

$$\frac{1}{2} \cdot \frac{1}{2e} \left(1 + \frac{1}{2e}\right) + \frac{1}{2} \left(1 - \frac{1}{2e}\right) = \frac{1}{2} \left(1 + \frac{1}{2e}\right).$$

Thus, we show that the expected value of $corr(C)$ is less than $\frac{1}{2} \left(1 + \frac{1}{2e}\right)$. This expected value is precisely the following probability: "Randomly choose $h$. Choose $x$ at random
according to $D$. Choose $y$ at random in the same component $C$ as $x$. Do $h(x)$ and $h(y)$ agree in their last bits?"

Under the condition that $x \neq y$, the probability that the last bit of $h(x)$ equals the last bit of $h(y)$ is exactly $1/2$. Thus, when we pick $(h, x, y)$ as above, the probability that the last bit of $h(x)$ equals the last bit of $h(y)$ is precisely $1/2$ times the quantity $1 + \Pr[x = y]$. We show that $\Pr[x = y]$ is at most $1/2^z$. Recall that $x$ and $y$ are picked by taking a random $h$, a random $x$ according to $D$, and then a random $y$ according to $D$ in the same component as $x$ with respect to $h$. Now, once we have picked $h$ and $x$, and $x$ lies in component $C$, the probability that we pick $y = x$ is $\Pr[x|C] = \Pr[x]/\Pr[C] < \frac{2^{z-m}}{2^m}$, because $D$ has Chor-Goldreich entropy at least $m$. Thus, we must show that with high probability that $\Pr[C]$ is large. Fix $h$. There are $2^l$ components in all. Call a component small if it has probability at most $\frac{2^{z+1}}{2^m}$. Then the probability of landing in a small component is at most the product of the largest possible size of a small component and an upper bound on the total number of small components, i.e.

$$\frac{2^{z+1}}{2^m} \cdot 2^l = \frac{2^{z+1}}{2^m} \cdot 2^{m-z-2} < \frac{1}{2^{z+1}}.$$ 

Thus, with probability at least $1 - \frac{1}{2^{z+1}}$, $x$ lands in a component of size at least $\frac{2^{z+1}}{2^m}$. In this case, $\Pr[y = x]$ is no more than $\frac{2^{z-m}}{2^m} = \frac{1}{2^z}$. Thus, the unconditional probability that $y = x$ is no more than $\frac{1}{2^z}$. Backtracking through our argument, we find we are done with the proof. □

To motivate the first application of Lemma 3.1, consider a private key encryption scheme. This is a system by which two people who have previously met and privately selected a random $n$ bit key $k$, can send messages of length longer than $n$ securely over an insecure channel. In other words, a computationally limited eavesdropper who does not know $k$ should not be able to deduce any information about which $n + 1$ bit message was sent from overhearing the conversation on the insecure channel. For now, we only consider deterministic, one round protocols for private key encryption. More formally:

**Definition 3.2:** A (deterministic, one round) **private key encryption system**, is a pair of polynomial time computable functions of two inputs, $E(k, x)$ and $D(k, z)$, such that $D(k, E(k, x)) = x$, where $m = |x| > |k| = n$. It is secure if from $E(k, x)$ and all but 1 bit of $x$, no poly-size circuit family can predict the hidden bit of $x$ with probability exceeding $\frac{1}{2} + \frac{1}{q(n)}$ for any polynomial $q$. (Here, $k$ and $x$ are chosen uniformly at random.)

Note that the function $E(k, x)$ is at most $2^n$ to 1. This is simply because no two messages can map to the same encryption under the same key. Thus, the distribution $E(k, x)$ (where $k$ and $x$ are randomly and uniformly chosen) has Chor-Goldreich entropy at least $m$. Consider the following two distributions $A$ and $B$ on $n + m$ bit strings. $A$ is given by $E(k, x)$, where both $k$ and $x$ are randomly and uniformly chosen. $B$ is given by $E(k, x)\|r$, where $k$, $x$ and $r$ are randomly chosen uniformly and $|r| = |x| = m$. $B$
has Chor-Goldreich entropy at least $2m$ since the Chor-Goldreich entropy of $E(k, x)$ is at least $m$ and $r$ is chosen independently. By a variant of Theorem 2.14, since no bit of $x$ is predictable given the other bits of $x$ and and $E(k, x)$, $A$ and $B$ are computationally indistinguishable. Distribution $A$ can be generated in polynomial time from a seed of length $n + m < 2m \leq \text{Ent}^CG(B)$. Thus, with a seed of length $n + m$, we can generate what looks, to a poly-size circuit, just like a random element of a distribution with entropy larger than $n + m$! This is analogous to a pseudo-random number generator, where on a seed of length $n$, it is possible to generate what looks like a random element of the uniform distribution on strings of a length greater than $n$. This motivates the following definition:

**Definition 3.3**: We call number generator $f$ *pseudo-slightly-random* if there is an ensemble $D$, where $\text{Ent}^CG(D_n) \geq n + 1$, such that the ensemble defined by $f$ is computationally indistinguishable from $D$.

**Theorem 3.4**: If there is a pseudo-slightly-random generator, then there is a pseudo-random generator.

**Proof**: Let $f$ be a pseudo-slightly-random generator, let $D$ be the ensemble mentioned in Definition 3.3 and let $q(n) = n$. Then, by Theorem 2.17, we have the ensemble determined by $f'(x_1, \ldots, x_n) = f(x_1) \cdots f(x_n)$ is computationally indistinguishable from $D^q$. $\text{Ent}^CG(D^q_n) \geq (n + 1)n = n^2 + n$. Let $m = l(n) \cdot n$, where $l(n)$ is the length of the output of $f$ on inputs of length $n$. Let $h$ be chosen uniformly at random from $H_{m, n^2 + 1}$, and let $d_1 \cdots d_n$ be randomly selected according to $D^q_n$. Then, by Lemma 3.1, the distribution given by $h \cdot h(d)$ is statistically indistinguishable from the uniform distribution on $|h| + n^2 + 1$ bit strings within $2^{-O(n)}$. Also, since $D^q$ is computationally indistinguishable from the ensemble determined by $f'$, the ensemble defined by $h \cdot h(d)$ is computationally indistinguishable from the ensemble defined by $h \cdot h\left(f'(x_1, \ldots, x_n)\right)$. Thus, this latter ensemble is computationally indistinguishable from the uniform distribution on $|h| + n^2 + 1$ bit strings. Thus, $g(h, x_1, \ldots, x_n) = h \cdot h(f'(x_1, \ldots, x_n))$ is a pseudo-random number generator since the total length of the inputs is $|h| + n^2$ and the ensemble defined by $g$ is computationally indistinguishable from a random string of length $|h| + n^2 + 1$. □

**Corollary 3.5**: There exists a secure (deterministic) private key encryption scheme iff there is a pseudo-random number generator.

**Proof**: We have already discussed how any private key encryption scheme yields a pseudo-slightly-random generator. From Theorem 3.4, we conclude that any secure private key encryption scheme can be used to construct a pseudo-random number generator. The converse, that a pseudo-random number generator yields a secure private key encryption scheme, is a well-known application of pseudo-random generators. □

In fact, from the work of [GGM] and [LR], we know that the existence of any pseudo-
random number generator is sufficient to construct the strongest form of private key scheme suggested to date, the pseudo-random block cipher. In the final paper we will show that an even weaker form of private key encryption (suggested by Rackoff [R]), an interactive probabilistic private key encryption scheme, suffices to construct a pseudo-random number generator.

Can we replace the condition that $D$ has high Chor-Goldreich 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 $1/2$ (and thus the distribution has Chor-Goldreich entropy $1$); thus, any function computed based on one sample from this distribution generates some output with probability at least $\frac{1}{2}$, and therefore is highly non-random. This problem hints at a partial solution: take multiple independent samples from the distribution.

**Lemma 3.6** : Let $D$ be a distribution on $n$ bit strings. Let $k$ be a positive integer. Then there is a distribution $E$ on $nk$ bit strings satisfying:

1. $\text{Ent}^{CG}(E) \geq k \cdot \text{Ent}(D) - nk^{2/3} - kn2^{-n}$.

2. $E$ is statistically indistinguishable from $D^k$ within $2^{-k^\epsilon} + k2^{-n}$ for a fixed $\epsilon > 0$.

**Proof** : Let $D'$ be the distribution obtained by restricting $D$ to the set of elements with probability at least $2^{-n}$. The sum of the probabilities of all such elements is at most $2^{-n}$, and thus contribute at most $n2^{-n}$ to $\text{Ent}(D)$. Then $D'$ is statistically indistinguishable from $D$ within $2^{-n}$ and $\text{Ent}(D') \geq \text{Ent}(D) - n2^{-n}$.

Consider the random variable $X$ with values in the interval $[0, 2n]$ defined by $X(d) = -\log(\PrD[d])$. When $d$ is randomly chosen according to $D'$, the expected value of $X$ is $\text{Ent}(D')$. When $d_1, \ldots, d_k$ are selected independently and randomly according to $D'$, $Y = \sum_{i=1, \ldots, k} X(d_i)$ is a sum of independent random variables on the interval $[0, 2n]$ with expected value $\text{Ent}(D')$. Hence, by an elementary extension of Chernoff bounds, with exponentially high probability (in $k$) $Y$ has value within an additive factor of $nk^{2/3}$ of its expectation. Thus, with exponentially high probability, $Y > k\text{Ent}(D') - nk^{2/3}$. This means that only with exponentially small probability the sequence $d_1, \ldots, d_k$ has probability greater than $2^{-k\text{Ent}(D') + nk^{2/3}}$. Restricting $D^k$ to the complement of this exponentially small in probability set of sequences, we obtain $E$. □

**Corollary 3.7** : Let $D$ be a distribution on $n$ bit strings. Let $k = n^c$. Let $h$ be uniformly and randomly chosen from $H_{nk; k\text{Ent}(D) - 2nk^{2/3}}$, and let $d_i$ be independently and randomly chosen according to $D$ for $i = 1, \ldots, k$. Then the distribution $h\upharpoonright_1(d_1, \ldots, d_k)$ is quasi-random within an exponentially small in $n$ amount.

**Proof** : Combine Lemmas 3.1 and 3.6. □
manner statistically close but still $n^{-c}$ distinguishable from the uniform distribution is considered to be unsuccessful. This is a very weak form of a one-way function, but the following theorem shows that such a one-way function can be used to construct a one-way function in the usual sense.

**Theorem 4.5**: If there is an informationally one-way function then there is a one-way function.

**Proof**: Let $f$ be an informationally one-way function with associated constant $c$. Let $g(h, i, x) = h \oplus f(h(x)) \oplus (i + 2c \log n)$ (where $i \in \{1, \ldots, n\}$ and $h \in H_{n, n+2c \log n}$. In the full paper, we prove that $g$ is a weakly one-way function. By Theorem 2.19, this suffices to construct a strongly one-way function. □

Almost all cryptosystems, for any application, utilize some type of one-way function. However, it is conceivable that cryptographic goals such as private key encryption and bit commitment could be accomplished without using any one-way function. In the full paper, we will show that any protocol whatsoever for these tasks yields in a straightforward manner an informationally one-way function, and hence by Theorem 4.5, a one-way function in the usual sense.

## 5 The Main Result

In this section we show that any one-way function can be used to construct a pseudo-random number generator. Our proofs make precise many intuitive notions that have been motivating constructions of pseudo-random generators since the original Blum and Micali [BM] construction. Before we give these proofs we discuss the roles that these ideas play both in our and previous work.

Blum and Micali's construction, which has been a paradigm for almost all later constructions of pseudo-random number generators, takes a specific intractability assumption, discrete log, and shows how to use this intractability to extract an extra bit of computational entropy while maintaining the Shannon entropy of the seed. To be more precise, they show that under this intractability assumption, there is a computable permutation that completely hides a computable bit of the seed. Since a permutation does not lose Shannon entropy, the hidden bit can be thought of as giving us an extra bit of computational entropy with no corresponding loss in Shannon entropy. Yao [Y] generalizes this by showing how to convert the intractability of inverting any one-way permutation into a hidden bit. Thus, this is a general method for converting computational difficulty into computational entropy. Goldreich and Levin [GL] have recently proved that any one-way function hides a computable bit. However, a general one-way function may lose Shannon entropy by being many-to-one. Thus, one bit of computational entropy is gained but many bits of Shannon entropy are lost, resulting in a net loss. Levin [L1] shows that the loss of Shannon entropy is manageable as long as some computational intractability remains after
a procedure that finds \( x \) given \( f(x) \) and \( \text{rank}(x) \) then given just \( f(x) \), we could find some preimage by randomly and uniformly choosing a value for \( \text{rank}(x) \) in the range from 0 to “the size of the preimage of \( f(x) \)” - 1. In fact, this algorithm produces a random preimage of \( f(x) \). In actuality it is impossible to compute \( \text{rank}(x) \) in polynomial time, but it turns out that giving \( h(x) \) along with \( f(x) \), where \( h \) is randomly chosen from an appropriate family of hash functions, has approximately the same effect as giving \( \text{rank}(x) \) in the sense that giving \( h(x) \) in addition to \( f(x) \) uniquely determines \( x \) while at the same time does not make it computationally easier to find \( x \).

**Definition 4.1**: We say that \( f \) is size computable if there is a polynomial time algorithm that for every \( x \) outputs an estimate of the number of preimages of \( f(x) \) accurate within a multiplicative factor of 2.

**Definition 4.2**: We say that \( f \) is one-to-one within \( \delta(n) \) if, with probability at least \( 1 - \delta(n) \), \( f(x) \) has the unique preimage \( x \) when \( x \in \{0,1\}^n \).

**Theorem 4.3**: Assume that there is a size computable weakly one-way function. Then, for every \( c > 0 \), there is a strongly one-way function that is one-to-one within \( n^{-c} \).

**Proof**: Let \( f \) be the size computable one-way function, and let \( A \) be the algorithm that computes preimage sizes for \( f \). Let \( d \) be such that no poly-size circuit family can invert \( f \) with probability greater than \( 1 - n^{-d} \). Let \( e = 2(c + d + 1) \). Let \( \text{pre}(x) = \log(A(x)) \); \( \text{pre}(x) \) is within 1 of the log of the preimage size. Let

\[
g(x, h) = h \# f(x) \# h(x) \uparrow (\text{pre}(x) + e \log n),
\]

where \( h \in H_{n,n+e \log n} \).

We omit the proof that this \( g \) is a weakly one-way function that is one-to-one within \( 2n^{-c} \). From \( g \) we construct \( g' \), where \( g' \) takes as input multiple independent copies of the input to \( g \) and outputs the concatenation of \( g \) applied to each copy. For the correct number of copies, \( g' \) is strongly one-way (proved using Theorem 2.19) and is still be one-to-one within \( n^{-c} \). \( \square \)

**Definition 4.4**: We say that a polynomial time computable function \( f \) is informationally one-way if there is a constant \( c \) so that for every poly-size family of probabilistic circuits \( C \) (the circuit has random inputs it uses to generate a probability distribution) and for all but finitely many \( n \), the distribution \( x \# f(x) \) is statistically distinguishable from the distribution \( C_n(f(x)) \# f(x) \) with probability at least \( n^{-c} \), for \( x \in \{0,1\}^n \) and the random inputs to \( C_n \) uniformly chosen.

Intuitively, an informationally one-way function is one where it is computationally infeasible to randomly generate preimages of \( f(x) \). With this definition, we fix a standard of invertibility \( c \). Even an algorithm that randomly generates preimages of \( f(x) \) in a
We generalize the concept of a pseudo-slightly-random generator, replacing Chor-Goldreich entropy with Shannon entropy. To make the connection with results discussed later in this paper, we first introduce a general framework for contrasting Shannon entropy with what we call "computational entropy".

**Definition 3.8**: Let \( s \) and \( t \) be functions from integers to positive reals. Let \( f \) be a polynomial time computable function. We say \( f \) has computational entropy at least \( s \) if there is an ensemble \( D \) such that \( D \) is computationally indistinguishable from the ensemble determined by \( f \) and \( \text{Ent}(D_n) \geq s(n) \). We say \( f \) has false entropy at least \( t \) if \( f \) has computational entropy at least \( t + \text{Ent}(f) \), i.e. the computational entropy exceeds the Shannon entropy by at least \( t \). We say \( f \) is a pseudo-entropy generator if \( f \) has computational entropy at least \( n + n^{-c} \) for some constant \( c \).

**Theorem 3.9**: If there is a pseudo-entropy generator then there is a pseudo-random number generator.

**Proof**: Let \( f \) be the pseudo-entropy generator, and let \( D \) be the ensemble with Shannon entropy greater than \( n + n^{-c} \) that is computationally indistinguishable from the ensemble defined by \( f \). Let \( k(n) = n^{3c+1}l(n)^3 \). Let
\[
g(h, x_1, \ldots, x_{k(n)}) = h \cdot h(f(x_1), \ldots, f(x_{k(n)})),
\]
where \( h \in H \) and \( H \) is a family of hash functions from \( k(n)l(n) \) bits to \( (n + n^{-c})k(n) - 2l(n)k(n)^2/3 \). Since \( f \) is computationally indistinguishable from \( D \), by Theorem 2.17,
\[
f(x_1), \ldots, f(x_k)
\]
is computationally indistinguishable from \( D^k \). Hence, \( g(h, x_1, \ldots, x_{k(n)}) \) is computationally indistinguishable from \( h \cdot h(d_1, \ldots, d_{k(n)}) \), which by Corollary 3.7 is quasi-random within an exponentially small in \( n \) amount. By our choice of \( k(n) \), the output of \( g \) is longer than the input to \( g \) by at least one bit, and therefore \( g \) is a pseudo-random number generator.
\( \Box \)

## 4 Hashing the Preimages

In this section, we introduce what proves to be an important step in our construction of a pseudo-random number generator from a one-way function, the hashing of the pre-images of a function. The intuition behind this idea is as follows. Assume for this discussion that \( f \) is a one-way function and that, given \( f(x) \), it is possible to compute in polynomial time the number of preimages of \( f(x) \), i.e. the number of \( y \) such that \( f(y) = f(x) \). We can use the lexicographical ordering on the preimages of \( f(x) \) to define the rank of \( x \), \( \text{rank}(x) \), which is the number of preimages of \( f(x) \) that precede \( x \). Given \( f(x) \) and \( \text{rank}(x) \), \( x \) is uniquely determined but is still hard to find computationally. This is because if we have
many iterates of the one-way function. If the function loses substantial Shannon entropy in each iteration, then after \( n \) iterations all of the entropy would be gone and hence all computational intractability as well. Thus, such a function must have many iterations where inverting the function is hard and very little Shannon entropy is lost. Recently Levin [L2] shows that if a function is one-way on a computable distribution and doesn’t lose much Shannon entropy on this distribution then it can be used to construct a pseudo-random number generator. His construction has a novel idea for recapturing the Shannon entropy of the original distribution without compromising the one-wayness of the function. We borrow this idea. Another idea we borrow is taken from Goldreich, Krawczyk and Luby [GKL], who use hash functions to randomly redistribute a uniform distribution on the range of a function into a roughly uniform distribution on the domain. This allows them to show that any regular (all preimage sizes roughly the same) one-way function can be used to construct a pseudo-random number generator.

The above discussion highlights some of the steps needed to convert computational difficulty, in the form of a one-way function, into a pseudo-random number generator. We now outline the major steps involved. First we give a general method for converting computational intractability into computational entropy. We obtain, from a one-way function \( f \), another function \( g \) such that the computational entropy of \( g(x) \) for randomly chosen \( n \) bit strings \( x \) is significantly greater than its Shannon entropy. However, the Shannon entropy of \( g(x) \) might be much less than \( |x| \), i.e. the Shannon entropy of the input distribution. We call this difference in Shannon entropy the drop in entropy of \( g \). We then give a method for extracting almost all of the drop in entropy of \( g \) by extracting additional information \( h(x) \) from the input \( x \) such that \( h(x) \) and \( g(x) \) are independent. Thus the Shannon entropy of \( h(x) \) together with the computational entropy of \( g(x) \) exceeds \( |x| \). We then convert the entropy in \( h(x) \) and \( g(x) \) into a distribution computationally indistinguishable from the uniform distribution on strings of length equal to the Shannon entropy of \( h(x) \) plus the computational entropy of \( g(x) \). At each stage of this process we rely heavily on the use of universal hash functions, and in particular on Lemm 3.1 and 3.6.

We use the following two results of Goldreich and Levin [GL].

**Theorem 5.1 (Goldreich and Levin [GL])** : Let \( f \) be a strongly one-way function. Then, for any poly-size circuit family \( C \), for all \( c > 0 \) and for all but finitely many \( n \), on input \( r, f(x) \), where \( x, r \in U_n \{0,1\}^n \), \( \Pr[C_n(f(x)|r) = r \cdot x] < \frac{1}{2} + n^{-c} \).

Goldreich and Levin generalize this result as follows.

**Definition 5.2** : Let \( f \) be a number generator and let \( D \) be an ensemble. We say \( f \) is strongly one-way on \( D \) if for every poly-size circuit family \( C \), for every \( c > 0 \) and for all but finitely many \( n \),

\[
\Pr[f(C_n(f(x))) = f(x)] < n^{-c},
\]

where \( x \in D_n \{0,1\}^n \). (Note that it may not be possible to randomly generate elements
according to $D$ in polynomial time, and that an inverting algorithm need not find an inverse $y$ of $f(x)$ such that $y$ has non-zero probability with respect to $D$.

Theorem 5.3 (Goldreich and Levin [GL]) : Let $f$ be strongly one-way on $D$. Then, for any poly-size circuit family $C$, for all $c > 0$ and for all but finitely many $n$, on input $r \parallel f(x)$, where $x \in D_n \{0,1\}^n$ and $r \in U_n \{0,1\}^n$, $\Pr[C_n(f(x), x) = r \cdot x] < \frac{1}{2} + n^{-c}$.

The following theorem allows us to convert the intractability of inverting a one-way function $f$ into computational entropy.

Theorem 5.4 : If there is a strongly one-way function then there is a polynomial time computable function with false entropy at least $\frac{1}{n}$.

Proof : Let $f$ be the strongly one-way function. We construct from $f$ a new polynomial time computable function $g$ with false entropy at least $\frac{1}{2n}$. The input to $g$ consists of two $n$ bit strings $r$ and $x$, $i \in \{1, \ldots, n\}$ and $h \in H_{n, n + \log n}$. We define

$$g(r, x, i, h) = r \parallel i \parallel h \parallel f(x) \parallel h(x) \parallel (i + \log n) \parallel r \cdot x.$$ 

We describe an ensemble $D$ that is computationally indistinguishable from the ensemble determined by $g$ such that $\text{Ent}(D) \geq \text{Ent}(g) + \frac{1}{2n}$. Although $D$ is not going to be polynomial time computable, it is easiest to think of $D$ as being generated from $r$, $x$, $i$, $h$ and a random bit $R$. $D$ is defined exactly the same as $g$ except for possibly the last bit, which in $g$ is always $r \cdot x$. The last bit of $D$ is $r \cdot x$ unless $i = \lceil \log(|f^{-1}(f(x))|) \rceil$, in which case the last bit of $D$ is $R$.

Define $i(x) = \lceil \log(|f^{-1}(f(x))|) \rceil$. Let ensemble $E$ be such that $E_n$ is the distribution on triples $(x, i(x), h)$ where $x \in U_n \{0,1\}^n$ and $h$ is randomly and uniformly chosen in $H$. We define

$$g'(x, i, h) = i \parallel h \parallel f(x) \parallel h(x) \parallel (i + \log n).$$

We claim (and prove below in the next paragraph) that $g'$ is strongly one-way on $E$. To distinguish $D$ from $g$ is equivalent to being able to predict $r \cdot x$, for $(x, i(x), h)$ randomly chosen according to $E_n$ and and $r \in U_n \{0,1\}^n$, given $r \parallel g'(x, i(x), h)$. From this claim and from Theorem 5.3 it follows that this task is computationally infeasible, and thus $g$ and $D$ are computationally indistinguishable.

We now prove the above claim. Assume there is a circuit $C_n$ that inverts $g'(x, i, h)$ with probability at least $n^{-c}$ when $(x, i, h)$ is randomly chosen according to $E_n$, i.e. on input

$$i(x) \parallel h \parallel f(x) \parallel h(x) \parallel (i(x) + \log n),$$

finds $y$ such that $f(y) = f(x)$ and $h(y) \parallel (i(x) + \log n) = h(x) \parallel (i(x) + \log n)$. We use only the fact that $f(y) = f(x)$ to prove this claim. We construct a circuit $C'_n$ of size polynomial in the size of $C_n$ and $n$ that inverts $f(x)$ with probability $\frac{1}{4n^2}$. On input $f(x)$, $C'_n$ runs through all possible $i = 1, \ldots, n$. For each $i$, $C'_n$ selects a random $h$ and a random
For each $\log n$ bit string $t$, $C_{n}'$ forms $u = s^t$ and simulates $C_{n}$ on input $i(x)yf(x)\|t$. We claim that with probability at least $\frac{1}{n^c}$ there is a round, i.e. an $i$ and a $t$, where in this simulation yields a $y$ with $f(y) = f(x)$. The round in question will have $i = i(x)$. We know that the probability that $C_{n}$, on input $i(x)yf(x)\|h(x)\|t(i(x)+\log n)$, for $x$ and $h$ randomly selected, yields such a $y$, is at least $n^{-c}$. Obviously, if instead of trying the one value $h(x)\|t(i(x)+\log n)$ we try $h(x)\|t(i(x)-12\log n)$ for all possible extensions $t$ of length $(2c + 1)\log n$, the chance that $C_{n}$ finds such a $y$ can only increase. Fix $f(x)$ and consider the distribution $h\|h(x')\|t(i(x)-12\log n)$ for $x'$ uniformly chosen from $f^{-1}(f(x))$. The distribution on $x'$ has Chor-Goldreich entropy $\log(|f^{-1}(f(x))|) \geq i(x)$. Hence, by Lemma 3.1, the distribution $h\|h(x')\|t(i(x)-12\log n)$ is statistically indistinguishable within $n^{-2c}$ from the distribution $h\|s$ where $s$ is uniformly chosen and $s \in U_{i(x)-12\log n} \{0,1\}^{|i(x)-12\log n|}$. Thus the probability that, for at least one $t$, $C_{n}$ on input $i(x)yf(x)\|h(x)\|s\|t$ finds a preimage of $f(x)$ for $s$ chosen at random differs from the probability when $s$ is given by $h(x)\|t(i(x)-12\log n)$ by at most $n^{-2c}$. Thus, this probability for a random $s$ is at least $n^{-c} - n^{-2c}$. From this contradiction of the strong one-wayness of $f$, we conclude that $g'$ is strongly one-way on $E$.

All that remains is to show that $\text{Ent}(D) > \frac{1}{2n} + \text{Ent}(g)$. This follows from the fact that with probability at least $1 - \frac{1}{n}$ over randomly chosen $x$ and $h$, $x$ is uniquely determined by $f(x)$ and $h(x)\|t(i(x)+\log n)$. Thus, triples of the form $(x,i(x),h)$, where $x$ is uniquely determined by $f(x)$ and $h(x)\|t(i(x)+\log n)$, are a $\left(1 - \frac{1}{n}\right)$ fraction of all triples $(x,i,h)$. For any such triple, and any $n$ bit string $r$, the last bit of $g(r,x,i,h)$ is completely determined by the preceding bits, whereas in $D$, the last bit is $R$, a random bit independent of the preceding bits. Thus, with probability at least $\left(1 - \frac{1}{n}\right)$, $D$ adds one random bit of entropy to $g$. Note that for all triples of the form $(x,i,h)$, $D$ has as much entropy as $g$. Thus, the claim follows. □

So far, we have constructed from the original one-way function $f$ another function $g$ such that its computational entropy is significantly greater than its Shannon entropy. However, the Shannon entropy of $g$ might be much less than the Shannon entropy of the input to $g$. Thus, it is possible that this loss in Shannon entropy might exceed the gain in computational entropy, resulting in a net loss of computational entropy. The following lemma gives us a way of recovering this lost Shannon entropy without compromising our gain in computational entropy.

**Lemma 5.5**: Let $g$ be any function from $n$ bit strings to $m$ bit strings. Let $k = n^c$. Consider the distributions $D$ and $E$, defined as follows. $D$ is given by

$$g(x_1)\|\cdots\|g(x_k)\|h\|h(x_1)\|\cdots\|h(x_k),$$

where the $x_i$ are randomly, uniformly and independently chosen $n$ bit strings and where $h$ is a randomly chosen hash function mapping $nk$ bits to $(n - \text{Ent}(g))k - 2nk^{2/3}$ bits. $E$ is given by

$$g(x_1)\|\cdots\|g(x_k)\|R,$$
where the $x_i$ are chosen the same way as in $D$ and $R$ is a \(|h| + (n - \text{Ent}(g))k - 2nk^{2/3}\) bit string randomly and uniformly chosen. Then, $D$ is statistically indistinguishable from $E$ within an exponentially small in $n$ amount.

**Proof:** We claim that, with high probability if, for $i = 1, \ldots, n$, we independently choose $x_i \in \{0, 1\}^n$ and fix $y_i = g(x_i)$ then the following distribution is quasi-random within an exponentially small in $n$ amount. Let $S_{y_1, \ldots, y_k}$ be the set of all sequences $x_1 \# \ldots \# x_k$ where $x_i \in g^{-1}(y_i)$. Randomly and uniformly choose a sequence $x_1 \# \ldots \# x_k \in S$ and a random $h$. The distribution is defined as

$$h \# h(x_1 \# \ldots \# x_k).$$

From Lemma 3.1, this distribution is quasi-random within an exponentially small in $n$ amount if the Chor-Goldreich entropy of the uniform distribution on $S_{y_1, \ldots, y_k}$ is substantially greater than $(n - \text{Ent}(g))k - 2nk^{2/3}$ (this is the number of bits that $h$ outputs). The Chor-Goldreich entropy of the uniform distribution on $S_{y_1, \ldots, y_k}$ is simply

$$\log(|S_{y_1, \ldots, y_k}|) = \sum_{i=1}^{k} \log(|g^{-1}(y_i)|).$$

This is the sum of $k$ independent random variables with range $0$ to $n$, and thus, using Chernoff bounds, is within an additive factor of $nk^{2/3}$ of its expected value with probability exponentially close to 1. The expected value of each random variable in the sum is $n - \text{Ent}(g)$, and thus the claim follows. \( \square \)

This lemma almost allows us to, from a generator $g$ with $n^{-c}$ bits of false entropy, construct a pseudo-entropy generator. The problem with the construction in the lemma is that we may not be able to actually compute $\text{Ent}(g)$. For now, assume that we can compute this value. We later show how to remove this restriction.

**Theorem 5.6:** Let $g$ be a number generator with at least $n^{-c}$ bits of false entropy. Assume that we can compute in polynomial in $n$ time the Shannon entropy of $g$ restricted to inputs of length $n$ to within an additive factor of $n^{-(c+1)}$. We can construct from $g$ a pseudo-entropy generator $g'$.

**Proof:** Let $k = n^{3c+4}$. Let $a$ be the approximation of the entropy of $g$ restricted to inputs of length $n$.

$$g'(h, x_1, \ldots, x_k) = g(x_1) \# \ldots \# g(x_k) \# h \# h(x_1 \# \ldots \# x_k),$$

where $h$ is a hash function from $nk$ bits to $(n-a)k - 2nk^{2/3}$ and $x_1, \ldots, x_k$ are $n$ bit strings. We claim that $g'$ has computational entropy at least $nk + |h| + 1$.

Let $D$ be the ensemble associated with $g$ in Definition 3.8, i.e. $D$ is computationally indistinguishable from $g$, and $\text{Ent}(D_n) \geq a + n^{-c} - n^{-(c+1)}$. (The last term is the round-off error involved in approximating $\text{Ent}(g)$ by $a$.) From Lemma 5.5, the distribution

$$g'(h, x_1, \ldots, x_k) = g(x_1) \# \ldots \# g(x_k) \# h \# h(x_1 \# \ldots \# x_k),$$
is statistically indistinguishable from the distribution

\[ g'(h, x_1, \ldots, x_k) = g(x_1) \oplus \cdots \oplus g(x_k) \oplus R, \]

where \( R \) is randomly and uniformly chosen among strings of length \(|h \oplus h(x_1 \oplus \cdots \oplus x_k)|\). Since \( D \) is computationally indistinguishable from \( g \), this last ensemble is computationally indistinguishable from \( D^k \oplus R \) (by Theorem 2.17). \( D^k \oplus R \) has Shannon entropy \( k \text{Ent}(D_n) + |R| = k \text{Ent}(D_n) + |h| + (n - a)k - 2nk^{2/3} \). Since \( \text{Ent}(D_n) \geq a + n^{-c} - n^{-(c+1)} \), this quantity is at least \( nk + |h| + 1 \) for our choice of \( k \). \( \square \)

**Corollary 5.7**: Let \( g \) be as described in Theorem 5.6. Then, there is a pseudo-random number generator.

**Proof**: Combine Theorem 5.6 with Theorem 3.9. \( \square \)

The final step is to convert any \( g \) with false entropy into a pseudo-random number generator, even if there is no way to compute \( \text{Ent}(g) \) in polynomial time. The idea is that there are only \( n^{c+1} \) possible values for \( \text{Ent}(g) \) to within \( n^{-c} \). Thus, we can try all possible values, and for at least one value we obtain random looking bits. By taking the vector sum of all of the strings generated (exclusive-or), we are guaranteed a random looking string. If, for each of the \( n^{c+1} \) possible values we generate a \( n^{c+3} \) bit string from an \( n \) bit seed we obtain more pseudo-random bits of output than bits of input. In the final paper we will give this argument in full detail; it consists of merely modifying the theorems presented here using standard known techniques, i.e. those used to prove Theorem 2.15. This allows us to conclude:

**THEOREM 5.8**: If there is a weakly one-way function then there is a pseudo-random number generator.

**Conclusions and Open Problems**

The results presented in this paper unify different concepts in theoretical cryptography. When combined with other work ([GGM], [LR], [GMW], [N]), they show that applications ranging from private key encryption to zero-knowledge proofs can be based on any one-way function. We also show 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. Our work uses as a standard of security for cryptography the immunity of a protocol to attack from an adversarial poly-size circuit family. We feel this is in many ways the natural model. However, it would be interesting to obtain the same results with respect to an adversary restricted to probabilistic polynomial time Turing machine computations. We do not see how to duplicate our results with respect to this model.
nal problem is to characterize the conditions under which cryptographic applica-
possible. Although we give characterizations for some applications in this paper,
has remain open. Naor and Yung [NY] have recently given a signature scheme
be based on any one-way permutation. Can this be weakened to give a signature
based on any one-way function? Some applications seem unlikely to be shown
based on any one-way function, e.g. Impagliazzo and Rudich [IR] give strong
that secret exchange is an application of this kind.
her general problem is to find other notions of intractability that can be used
ographic applications. It may turn out that not every “hard problem” can be
as an inversion of a polynomial time computable function. A result related
of question can be found in the work of Nisan and Wigderson [NW], where
ype of “hard problem” that would imply the same type of consequences to
ic time that the existence of a pseudo-random number generator has.
important issue is that of efficiency; the construction we give here for a pseudo-
number generator based on any one-way function increases the size of the seed by
omial amount. For practical applications, it would be nice to have a much
sinous construction.

Acknowledgements
research evolved over a long period of time and was greatly influenced by many
we thank Amos Fiat, Moni Naor, Ronitt Rubinfeld, Manuel Blum, Steven Rudich,
isa, Lance Fortnow, Umesh Vazirani, Leonid Levin, Charlie Rackoff, Oded Gol-
d Hugo Krawczyk for their insights and contributions to this work. In particular,
esh and Manuel for their advice and enthusiasm, Charlie for raising the
whether private key encryption systems are possible without a pseudo-random
ator, Lance for suggesting a version of Lemma 3.6, Oded and Hugo for intro-
d second author to this question and sharing their tremendous insight into this
Leonid for allowing us to benefit from his vast knowledge about one-way
entropy.

ref
n. M., “Independent Unbiased Coin Flips From a Correlated Biased Source: A
l. M., and Micali, S., “How to Generate Cryptographically Strong Sequences
t. B., and O. Goldreich, “Unbiased Bits from Sources of Weak Randomness and
Stochastic Communication Complexity”, 26th FOCS, 1985, pp. 429-442, journal
Archer, J., and M. Wegman, “Universal Classes of Hash Functions”, JCSS, 1979,
3, pp. 143-154.


