On the Existence of Pseudorandom Generators

(Extended Abstract)

Oded Goldreich
Dept. of Computer Sc.
Technion
Haifa, Israel

Hugo Krawczyk
Dept. of Computer Sc.
Technion
Haifa, Israel

Michael Luby
Dept. of Computer Sc.
University of Toronto
Ontario, Canada

ABSTRACT – Pseudorandom generators (suggested and developed by Blum and Micali [BM] and Yao [Y]) are efficient deterministic programs that expand a randomly selected k-bit seed into a much longer pseudorandom bit sequence which is indistinguishable in polynomial time from an (equally long) sequence of unbiased coin tosses. Pseudorandom generators are known to exist assuming the existence of functions that cannot be efficiently inverted on the distributions induced by applying the function iteratively polynomially many times (Levin [L]). (This generalizes previous results of [BM] and [Y].) This sufficient condition is also a necessary one, but it seems difficult to check whether particular functions, assumed to be one-way, are also one-way on their iterates. This raises the fundamental question whether the mere existence of one-way function suffices for the construction of pseudorandom generators.

In this paper we present progress towards resolving this question. We consider *regular* functions, in which every image of a k-bit string has the same number of preimages of length k. We show that if a regular function is one-way then pseudorandom generators do exist. In particular, assuming the intractability of general factoring, we can now prove that pseudorandom generators do exist. Another application is the construction of a pseudorandom generator based on the assumed intractability of decoding random linear codes.

1. INTRODUCTION

In recent years randomness has become a central notion in the theory of computation. It is heavily used in the design of sequential, parallel and distributed algorithms, and is of course crucial to cryptography. Once so frequently used, randomness itself has become a resource, and economizing on the amount of randomness required for an application has become a natural concern. It is in this light that the notion of pseudorandom generators was first suggested and the following fundamental result was derived: the number of coin tosses used in any practical application (modeled by a polynomial time computation) can be decreased to an arbitrarily small power of the input length.

The key to the above informal statement is the notion of a pseudorandom generator suggested and developed by Blum and Micali [BM] and Yao [Y]. A *pseudorandom generator* is a deterministic polynomial time algorithm which expands short seeds into longer bit sequences, such that the output ensemble is polynomially-indistinguishable from the uniform probability distribution. More specifically, the generator (denoted G) expands a k-bit seed into a longer, say $2k$-

Research done while the third author was visiting the Computer Science Department of the Technion. First author was supported by grant No. 86-00301 from the United States - Israel Binational Science Foundation (BSF), Jerusalem, Israel. Third author was partially supported by a Natural Sciences and Engineering Research Council of Canada operating grant No. A8092 and by a University of Toronto grant.

0272-5428/88/0000/0012$01.00 © 1988 IEEE

bit, sequence so that for every polynomial time algorithm (distinguishing test) T, any constant $c>0$, and sufficiently large k

$$\left| Prob\left[T(G(X_k))=1\right] - Prob\left[T(X_{2k})=1\right] \right| \leq k^{-c},$$

where X_m is a random variable assuming as values strings of length m, with uniform probability distribution. It follows that the strings output by a pseudorandom generator G can substitute the unbiased coin tosses used by any polynomial time algorithm A, without changing the behavior of algorithm A in any noticeable fashion. This yields an equivalent polynomial time algorithm, A', which randomly selects a seed, uses G to expand it to the desired amount, and then runs A using the output of the generator as the random source required by A. The theory of pseudorandomness was further developed to deal with function generators and permutation generators and additional important applications to cryptography have emerged [GGM, LR]. The existence of such seemingly stronger generators was reduced to the existence of pseudorandom (string) generators.

In light of their practical and theoretical value, constructing pseudorandom generators and investigating the possibility of such constructions is of major importance. A necessary condition for the existence of pseudorandom generators is the existence of one-way functions (since the generator itself constitutes a one-way function). However, it is not known whether this necessary condition is sufficient. Instead, stronger versions of the one-wayness condition were shown to be sufficient. Before reviewing these results, let us recall the definition of a one-way function.

Definition 1: A function $f:\{0,1\}^*\rightarrow\{0,1\}^*$ is called *one-way* if it is polynomial time computable, but not "polynomial time invertible". Namely, there exists a constant $c>0$ such that for **any** probabilistic polynomial time algorithm A, and sufficiently large k

$$Prob\left[A(f(x),1^k)\notin f^{-1}(f(x))\right] > k^{-c}, \quad (*)$$

where the probability is taken over all x's of length k and the internal coin tosses of A, with uniform probability distribution.
(Remark: The role of 1^k in the above definition is to allow algorithm A to run for time polynomial in the length of the preimage it is supposed to find. Otherwise, any function which shrinks the input by more than a polynomial amount would be considered one-way.)

1.1. Previous Results

The first pseudorandom generator was constructed and proved valid, by Blum and Micali, under the assumption that the discrete logarithm problem is intractable on a non-negligible fraction of the instances [BM]. In other words, it was assumed that exponentition modulo a prime (i.e. the 1-1 mapping of the triple (p,g,x) to the triple $(p,g,g^x \bmod p)$, where p is prime and g is a primitive element in Z_p^*) is one-way. Assuming the intractability of factoring integers of the form $N=p\cdot q$, where p and q are primes and $p\equiv q\equiv 3 \bmod 4$, a simple pseudorandom generator exists [BBS, ACGS][1]. Under this assumption the permutation, defined over the quadratic residues by modular squaring, is one-way.

1) A slightly more general result, concerning integers with all prime divisors congruent to 3 mod 4, also holds [CGG].

Yao has presented a much more general condition which suffices for the existence of pseudorandom generators; namely, the existence of one-way permutations [Y] [2].

Levin has weakened Yao's condition, presenting a necessary and sufficient condition for the existence of pseudorandom generators [L]. Levin's condition, hereafter referred to as *one-way on iterates*, can be derived from Definition 1 by substituting the following line instead of line (*)

$$(\forall i, 1 \le i < k^{3/2})$$

$$Prob\left[A(f^{(i)}(x), 1^k) \notin f^{-1}(f^{(i)}(x)) \right] > k^{-c},$$

where $f^{(i)}(x)$ denotes f iteratively applied i times on x. (As before the probability is taken uniformly over all x's of length k.) Clearly, any one-way permutation is one-way on its iterates. It is also easy to use any pseudorandom generator in order to construct a function which satisfies Levin's condition.

Levin's condition for the construction of pseudorandom generators is somewhat cumbersome. In particular, it seems hard to test the plausibility of the assumption that a particular function is one-way on its iterates. Furthermore, *it is an open question whether Levin's condition is equivalent to the mere existence of one-way functions*

1.2. Our Results

In this extended abstract we present progress towards resolving the above open problem. We consider "regular" functions, in which every element in the range has the same number of preimages. More formally, we call a function f *regular* if there is a function $m(\cdot)$ such that $\forall n$ and for every $x \in \{0,1\}^n$ the cardinality of $f^{-1}(f(x))$ is $m(n)$. Clearly, *every* 1-1 function is regular (with $m(n)=1, \forall n$). Our main result is

Main Theorem (special case): *If there exists a regular one-way function then there exists a pseudorandom generator.*

Regularity appears to be a simpler condition than the intractability of inverting on the function's iterates. Furthermore, many natural functions (e.g. squaring modulo an integer) are regular and thus, using our result, a pseudorandom generator can be *efficiently* constructed assuming that any of these functions is one-way. In particular, if factoring is weakly intractable (i.e. every polynomial time factoring algorithm fails on a non-negligible fraction of the integers) then pseudorandom generators do exist. This result was not known before. (It was only known that the intractability of factoring a special subset of the integers implies the existence of a pseudorandom generator.) Using our results, we can also construct a pseudorandom generator based on the (widely believed) conjecture that decoding random linear codes is intractable.

The main theorem is proved by transforming any given regular one-way function into a function that is one-way on its iterates (and then applying Levin's result [L]). It is interesting to note that not every (regular) one-way function is "one-way on its iterates". To emphasis this point, we show (in Appendix A) that from a (regular) one-

2) In fact, Yao's condition is slightly more general. He requires that f is 1-1 and that there exists a probability ensemble Π which is invariant under the application of f and that inverting f is "hard on the average" when the input is chosen according to Π.

way function we can construct a (regular) one-way function which is easy to invert on the distribution obtained by applying the function *twice*. The novelity of this work is in presenting *a direct way to construct a function which is one-way on its iterates from any regular one-way function (which is not necessarily one-way on its iterates).*

2. MAIN RESULT

2.1. Preliminaries

In the sequel we make use of the following definition of *strongly* one-way function. (When referring to Definition 1, we shall call the function *weak* one-way or simply one-way).

Definition 2: A polynomial time computable function $f:\{0,1\}^* \rightarrow \{0,1\}^*$ is called *strongly one-way* if for any probabilistic polynomial time algorithm A, any positive constant c, and sufficiently large k,

$$Prob\left[A(f(x),1^k) \in f^{-1}(f(x))\right] < k^{-c},$$

where the probability is taken over all x's of length k and the internal coin tosses of A, with uniform probability distribution.

Theorem 1 (Yao [Y]): There exists a strong one-way function if and only if there exists a (weak) one-way function. Furthermore, given a one-way function, a strong one can be constructed.

It is important to note that Yao's construction preserves the regularity of the function. Thus, we may assume without loss of generality, that we are given a function f which is strongly one-way and regular. For the sake of simplicity, we assume f is *length preserving* (i.e. $\forall x, |f(x)|=|x|$). Our results hold also without this assumption (see subsection 2.6).

Notation: For a finite set S, the notation $s \in_R S$ means that the element s is randomly selected from the set S with uniform probability distribution.

2.2. Main Ideas

Suppose we are given a regular and strongly one-way function f. We construct a new function f' that is also strongly one-way, not only for its first application but for $k^{3/2}$ iterations, where k is the length of the input to f'. Using Levin's results [L], this construction suffices for proving our Main Theorem.

The following are the main ideas behind this construction. Since the function f is strongly one-way, any algorithm trying to invert f can succeed only with negligible probability. Here the probability distribution on the range of f is induced by choosing a random element from the domain and applying f. However, this condition says nothing about the capability of an algorithm to invert f when the distribution on the range is substantially different. For example, there may be an algorithm which is able to invert f if we consider the distribution on the range elements induced by choosing a random element from the domain and applying f twice or more (see Appendix A). To prevent this possibility, we "randomly" redistribute, after each application of f, the elements in the range to locations in the domain. We prove the validity of our construction by showing that the probability distribution induced on the range of f by our "random" transformations is close to the distribution induced by the first application of f.

The function f' we construct must be deterministic, and therefore the "random" redistribution must be deterministic (i.e. uniquely defined by the input to f'). To achieve this, we use high quality hash functions. More specifically, we use hash functions which map n-bit strings to n-bit strings, such that the locations assigned to the strings by a randomly selected hash function are uniformly distributed and n-wise independent. For properties and implementations of such functions see [CW, J, CG, Lu]. We denote this set of hash functions by $H(n)$. Elements of $H(n)$ can be described by bit strings of length n^2. In the sequel $h(\in H(n))$ refers to both the hash function and to its representation.

2.3. The Construction of f'

We view the input string to f' as containing two types of information. One part of the string is the description of hash functions that implement the "random" redistributions and the other part is interpreted as inputs for the original function f. Each of the hash functions described in the input is used in several iterations of f'.

The following is the definition of the function f':

$$f'(i_1, i_2, h_0, \cdots, h_{t(n)-1}, x_0, \ldots, x_{s(n)-1})$$
$$= (i'_1, i'_2, h_0, \cdots, h_{t(n)-1}, x_0, \cdots, x_{i_1-1},$$
$$h_{i_2}(f(x_{i_1})), x_{i_1+1}, \cdots, x_{s(n)-1})$$

where $0 \le i_1 \le s(n)-1, 0 \le i_2 \le t(n)-1$, $x_j \in \{0,1\}^n$, and $h_j \in H(n)$. The quantities i'_1 and i'_2 are defined as $i'_2 = (i_2+1) \bmod t(n)$ and $i'_1 = i_1+1$ if $i_2 = t(n)-1$ and $i'_1 = i_1$ otherwise.

We fix $s(n)=n^6, t(n)=n^5$ and use $s(n)\cdot t(n) = n^{11}$ iterations of f' starting with $i_1 = i_2 = 0$. The length of the input to f' is $k = O(\log n) + t(n)\cdot n^2 + s(n)\cdot n \le 3n^7$, and thus the number of iterations is at least $k^{3/2}$, as required from the function f'. The rest of this section is devoted to the proof of the following theorem.

Theorem 2: Let f be a regular and strongly one-way function. Then the function f' defined above is strongly one-way for $k^{3/2}$ iterations on strings of length k.

Our Main Theorem follows from Theorem 2 and Levin's result (see [L] and Appendix B).

Let $h_0, h_1, \cdots, h_{t(n)-1}$ be $t(n)$ functions from the set $H(n)$. For $r = 1, \cdots, t(n)$, let g_r be the function $g_r = f\, h_{r-1} f\, h_{r-2} f \cdots h_0 f$ acting on strings of length n, let $G_r(n)$ be the set of all such functions g_r, let g be $g_{t(n)}$ and let $G(n)$ be the set of such functions g. From the above description of the function f' it is apparent that the inversion of an iterate of f' boils down to the problem of inverting f when the probability distribution on the range of f is $g_r(x)$ where $x \in_R \{0,1\}^n$. We show that, for most $g \in G(n)$, the number of preimages under g for each element in its range is close (up to a polynomial factor) to the number of preimages for the same range element under f. This implies that the same statement is true for most $g_r \in G_r(n)$ for all $r = 1, \cdots, t(n)$. The proof of this result reduces to the analysis of the combinatorial game that we present in the next subsection.

2.4. The game

Consider the following game played with M balls and M cells where $t(n) \ll M \le 2^n$. Initially each cell contains a

single ball. The game has $t(n)$ iterations. In each iteration, cells are mapped randomly to cells by means of an independently and randomly selected hash function $h \in_R H(n)$. This mapping induces a transfer of balls so that the balls residing (before an iteration) in cell σ are transferred to cell $h(\sigma)$. We are interested in bounding the probability that some cells contain "too many" balls when the process is finished. We show that after $t(n)$ iterations the probability that there is any cell containing more than some polynomial in n balls is negligibly small (i.e. less than any polynomial in n fraction).

We first proceed to determine a bound on the probability that a specific set of n balls is mapped after $t(n)$ iterations to a single cell.

Lemma 3: The probability that a specific set of n balls is mapped after $t(n)$ iterations to the same cell is bounded above by $p(n) = \left\lceil \frac{n \cdot t(n)}{M} \right\rceil^{n-1}$

Proof Sketch: Let $B = \{b_1, b_2, \cdots, b_n\}$ be a set of n balls. Notice that each execution of the game defines for every ball b_i a path through $t(n)$ cells. In particular, fixing $t(n)$ hash functions $h_0, h_1, \cdots, h_{t(n)-1}$, a path corresponding to each b_i is determined. Clearly, if two such paths intersect at some point then they coincide beyond this point. We modify these paths in the following way. The initial portion of the path for b_i that does not intersect the path of any smaller indexed ball is left unchanged. If the path for b_i intersects the path for b_j for some $j < i$ then the remainder of the path for b_i is chosen randomly and independently of the other paths from the point of the first such intersection.

Because the functions h_i are chosen totally independently of each other and because each of them has the property of mapping cells in an n-independent manner, it follows that the modified process just described is equivalent to a process in which a totally random path is selected for each ball in B. Consider the modified paths. We say that two balls b_i and b_j *join* if and only if their corresponding paths intersect. Define *merge* to be the reflexive and transitive closure of the relation join (over B). The main observation is that if $h_0, h_2, \cdots, h_{t(n)-1}$ map the balls of B to the same cell, then $b_1, b_2, \cdots, b_n$ are all in the same equivalence class with respect to the relation merge. In other words, the probability that the balls in B end up in the same cell in the original game is bounded above by the probability that the merge relation has a single equivalence class (containing all of B). Let us now consider the probability of the latter event.

If the merge relation has a single equivalence class then the join relation defines a connected graph with the n balls as vertices and the join relation as the set of edges. The "join graph" is connected if and only if it contains a spanning tree. Thus, an upper bound on the probability that the "join graph" is connected is obtained by the sum of the probabilities of each of the possible spanning trees which can be embedded in the graph. Each particular tree has probability at most $(t(n)/M)^{n-1}$ to be embedded in the graph ($t(n)/M$ is an upper bound on the probability of each edge to appear in the graph). Multiplying this probability by the (Cayley) number of different spanning trees (n^{n-2} cf. [E, Sec. 2.3]), the lemma follows. □

A straightforward upper bound on the probability that there is some set of n balls which are merged is the probability that n specific balls are merged multiplied by the number of possible distinct subsets of n balls. Unfortunately, this bound is worthless (as $\binom{M}{n} \cdot p(n) > 1$ (This phenomena is independent of the choice of the parameter n.). Instead we use the following technical lemma.

Lemma 4: Let S be a finite set, and let Π denote a partition of S. Assume we have a probability distribution on partitions of S. For every $A \subseteq S$, we define $\chi_A(\Pi)=1$ if A is contained in a single class of the partition Π and $\chi_A(\Pi)=0$ otherwise. Let n and n' be integers such that $n < n'$. Let $p(n)$ be an upper bound on the maximum over all $A \subseteq S$ such that $|A|=n$ of the probability that $\chi_A = 1$. Let $q(n')$ be an upper bound on the probability that there is some $B \subseteq S$ such that $|B| \geq n'$ and $\chi_B = 1$. Then

$$q(n') \leq \frac{\binom{|S|}{n} \cdot p(n)}{\binom{n'}{n}}$$

Proof: For $B \subseteq S$ we define $\xi_B(\Pi)=1$ if B is exactly a single class of the partition Π and $\xi_B(\Pi)=0$ otherwise. For every fixed partition Π,

$$\sum_{B \subseteq S, |B| \geq n'} \xi_B(\Pi) \leq \frac{1}{\binom{n'}{n}} \cdot \sum_{A \subseteq S, |A|=n} \chi_A(\Pi)$$

The lemma follows. □

Remark: Lemma 4 is useful in situations when the ratio $\frac{p(n)}{p(n')}$ is smaller than $\binom{|S|-n}{n'-n}$. Assuming that $n' \ll |S|$, this happens when $p(n)$ is greater than $|S|^{-n}$. Lemma 3 is such a case, and thus the application of Lemma 4 is useful.

Combining Lemmi 3 and 4, we get

Theorem 5: Consider the game played for $t(n)$ iterations. Then, the probability that there is $4t(n) \cdot n + n$ balls which end up in the same cell is bounded above by 2^{-n}.

Proof: By straightforward calculation, applying Lemma 4 with $|S|=M$ and $n'=4t(n) \cdot n + n$. □

2.5. Proof of Theorem 2

We now apply Theorem 5 to the analysis of the function f'. As before, let $G(n)$ be the set of functions of the form $g = f\, h_{t(n)-1} f \cdots h_0 f$. The functions $h = h_j$ are hash functions used to map the range of f to the domain of f. We let $h_0, \cdots, h_{t(n)-1}$ be randomly chosen uniformly and independently from $H(n)$, and this induces a probability distribution on $G(n)$. Denote the range of f (on strings of length n) by $R(n) = \{z_1, z_2, \ldots, z_M\}$. Let each z_i represent a cell. Consider the function h as mapping cells to cells. We say that h maps the cell z_i to the cell z_j if $h(z_i) \in f^{-1}(z_j)$, or in other words $f(h(z_i)) = z_j$. By the regularity of the function f, we have that the size of $f^{-1}(z_i)$ is equal for all $z_i \in R(n)$, and therefore the mapping induced on the cells is uniform. It is now apparent that $g \in_R G(n)$ behaves exactly as the random mappings in the game described in Section 2.4, and thus Theorem 5 can be applied. We get

Lemma 6: There is a constant c_0, such that for any constant $c > 0$ and sufficiently large n

$$Prob\left[\exists z \text{ with } |g^{-1}(z)| \geq n^{c_0} \cdot m(n)\right] \leq \frac{1}{n^c},$$

where $g \in_R G(n)$.

Let us denote by $G'(n)$ the set of functions $g \in G(n)$ such that for all z in the range of f $|g^{-1}(z)| < n^{c_0} \cdot m(n)$. By the above lemma, $G'(n)$ contains almost all of $G(n)$. It is clear that if $g \in G'(n)$ then for all z in the range of f and for all $r = 1, \cdots, t(n)$ the function g_r defined by the first r iterations of g satisfies $|g_r^{-1}(z)| < n^{c_0} \cdot m(n)$.

Lemma 7: For any probabilistic polynomial time algorithm A, for any positive constant c and sufficiently large n and for all $r = 1, \cdots, t(n)$,

$$Prob(A(g_r, z) \in f^{-1}(z)) < n^{-c}$$

where $g_r \in_R G_r(n)$ and $z = g_r(x)$, $x \in_R \{0,1\}^n$.

Proof: We prove the claim for $r = t(n)$ and the claim for $r = 1, \cdots, t(n)$ follows directly. Assume to the contrary that there is a probabilistic polynomial time algorithm A and a constant c_A such that $Prob(A(g,z) \in f^{-1}(z)) > n^{-c_A}$, where $g \in_R G(n)$ and $z = g(x)$, $x \in_R \{0,1\}^n$.

By using A, we can demonstrate an algorithm A' that inverts f, contradicting the one-wayness of f. The input to A' is $z = f(x)$ where $x \in_R \{0,1\}^n$. A' chooses $g \in_R G(n)$ and outputs $A(g,z)$. We show that A' inverts f with non-negligible probability. By assumption there is a non-negligible subset $G''(n)$ of $G'(n)$ such that, for each $g \in G''(n)$, A succeeds with significant probability to compute a $y \in f^{-1}(z)$ where $z = g(x)$ and $x \in_R \{0,1\}^n$. Since $g \in G'(n)$, for all z in the range of f the probability induced by g on z differs by at most a polynomial factor in n from the probability induced by f. Thus, for $g \in G''(n)$, A succeeds with significant probability to compute a $y \in f^{-1}(z)$ where $z = f(x)$ and $x \in_R \{0,1\}^n$. This is exactly the distribution of inputs to A', and thus A' succeeds to invert f with non-negligible probability, contradicting the strong one-wayness of f. □

Theorem 2 follows from Lemma 7 by noting that an algorithm A, given g and $g(x)$ as inputs, can simulate the extra information given to the inverter of f'. **Q.E.D**

2.6. Extensions

In the above exposition we assumed for simplicity that the function f is length preserving, i.e. $x \in \{0,1\}^n$ implies that the length of $f(x)$ is n. This condition is not essential to our proof and can be dispensed with in the following way. If f is not length preserving then it can be modified to have the following property: There is an n' such that $x \in \{0,1\}^n$ implies that the length of $f(x)$ is n'. This modification can be carried out using a padding technique that preserves the regularity of f. We can then modify our description of f' to use hash functions mapping n'-bit strings to n-bit strings.

Another extension is a relaxation of the regularity condition. A useful notion in this context is the histogram of a function.

Definition 3: The *histogram* of the function $f : \{0,1\}^* \to \{0,1\}^*$ is a function $hist_f : \mathbf{N} \times \mathbf{N} \to \mathbf{N}$ such that $hist_f(n,k)$ is the cardinality of the set

$$\{x \in \{0,1\}^n : \left\lfloor \log_2 |f^{-1}(f(x))| \right\rfloor = k\}$$

Regular functions have trivial histograms. Let f be a regular function such that for all $x \in \{0,1\}^n$, $|f^{-1}(f(x))| = m(n)$. The histogram satisfies $hist_f(n,k) = 2^n$ for

$$k = \left\lfloor \log_2(m(n)) \right\rfloor$$

and $hist_f(n,k) = 0$ otherwise. Weakly regular functions have slightly less dramatic histograms.

Definition 4: The function f is *weakly regular* if there is a constant $\varepsilon > 0$, a polynomial $p(\cdot)$ and a function $b(\cdot)$ such that the histogram of f satisfies (for all n)

i) $hist_f(n, b(n)) > \frac{2^n}{p(n)}$

ii) $\sum_{k=b(n)+1}^{n} hist_f(n,k) < \frac{2^n}{(n \cdot p(n))^{1+\varepsilon}}$

Clearly, this definition extends the original definition of regularity. Using our techniques one can show that the existence of weakly regular strongly one-way functions implies the existence of pseudorandom generators. For weakly regular functions with constant $\varepsilon > 5$ (used in Definition 4) the argument follows in a straightforward way (Details of this will be included in the final version of this paper.). To prove the statement for arbitrary $\varepsilon > 0$, we use a slightly different construction in which the generator outputs the description of the hashing functions (as part of its output). (The argument is further simplified by using a recent result of Goldreich and Levin [GL] (see Appendix C).)

For the applications in Section 3, and possibly for other cases, the following extension (referred to as *semi-regular*) is useful. Let $\{f_x\}_{x \in \{0,1\}^*}$ be a family of regular functions, then our construction can be still applied to the function f defined as $f(x,y) = (x, f_x(y))$ The idea is to use the construction for the application of the function f_x, while keeping x unchanged.

3. APPLICATIONS : Pseudorandom Generators Based on Particular Intractability Assumptions

In this section we apply our results in order to construct pseudorandom generators (*PRG*s) based on the assumption that one of the following two computational problems is "hard on a non-negligible fraction of the instances". The first problem is factoring (arbitrary!) integers. The second problem is decoding random linear codes.

3.1. PRG Based on the Intractability of the General Factoring Problem

It is known that pseudorandom generators can be constructed assuming the intractability of factoring integers of a special form [Y]. More specifically, in [Y] it is assumed that any polynomial time algorithm fails to factor a non-negligible fraction of integers that are the product of primes congruent to 3 modulo 4. With respect to such an integer N, squaring modulo N defines a permutation over the set of quadratic residues mod N, and therefore the intractability of factoring (such N's) yields the existence of a one-way permutation [R]. It was not known how to construct a one-way permutation or a pseudorandom generator assuming that factoring a non-negligible fraction of *all* the integers is intractable. In such a case modular squaring is a one-way function, but this function does not necessarily induce a permutation. Fortunately, modular squaring is a semi-regular function (see subsection 2.6), so we can apply our results.

Assumption IGF (*Intractability of the General Factoring Problem*): There exists a constant $c > 0$ such that for **any** probabilistic polynomial time algorithm A, and

sufficiently large k

$$Prob\left[A(N) \text{ does not split } N \right] > k^{-c},$$

where $N \in_R \{0,1\}^k$.

Corollary 8: The IGF assumption implies the existence of pseudorandom generators.

Proof: Define the following function $f(N,x)=(N, x^2 \bmod N)$. Clearly, this function is semi-regular. The one-wayness of the function follows from IGF (using Rabin's argument [R]). Using an extension of Theorem 2 (see subsection 2.6) the corollary follows. □

Subsequently, J. (Cohen) Benaloh has found a way to construct a one-way permutation based on the IGF assumption. This yields an alternative proof of Corollary 8.

3.2. PRG Based on the Intractability of Decoding Random Linear Codes

One of the most outstanding open problems in coding theory is that of decoding random linear codes. Of particular interest are random linear codes with constant information rate which can correct a constant fraction of errors. An (n,k,d)-*linear code* is an k-by-n binary matrix in which the bit-by-bit XOR of any subset of the rows has at least d ones. The Gilbert-Varshamov bound for linear codes guarantees the existence of such a code provided that $k/n < 1-H_2(d/n)$, where H_2 is the binary entropy function [McS, ch. 1, p. 34]. The same argument can be used to show (for every $\varepsilon > 0$) that if $k/n < 1-H_2((1+\varepsilon)\cdot d/n)$, then almost all k-by-n binary matrices constitute (n,k,d)-linear codes.

We suggest the following function $f:\{0,1\}^* \to \{0,1\}^*$. Let C be an k-by-n binary matrix, $x \in \{0,1\}^k$, and $e \in E_t^n \subseteq \{0,1\}^n$ be a binary string with at most $t = \lfloor (d-1)/2 \rfloor$ ones, where d satisfies the condition of the Gilbert-Varshamov bound (see above). Clearly E_t^n can be uniformly sampled by an algorithm S running in time polynomial in n (i.e. $S:\{0,1\}^{poly(n)} \to E_t^n$). Let $r \in \{0,1\}^{poly(n)}$ be a string such that $S(r) \in E_t^n$. Then,

$$f(C,x,r) = (C, C(x)+S(r)),$$

where $C(x)$ is the codeword of x (i.e. $C(x)$ is the vector resulting by the matrix product xC). One can easily verify that f just defined is semi-regular (i.e. $f_C(x,r)=C(x)+S(r)$ is regular for all but a negligible fraction of the C's). The vector $xC+e$ ($e=S(r)$) represents a codeword perturbed by the error vector e.

Assumption IDLC (*Intractability of Decoding Random Linear Codes*): There exists a constant $c>0$ such that for **any** probabilistic polynomial time algorithm A, and sufficiently large k

$$Prob\left[A(C, C(x)+e) \neq x \right] > k^{-c},$$

where C is a randomly selected k-by-n matrix, $x \in_R \{0,1\}^k$ and $e \in_R E_t^n$.

Now, either assumption IDLC is false which would be an earth-shaking result in coding theory or pseudorandom generators do exist.

Corollary 9: The IDLC assumption implies the existence of pseudorandom generators.

Proof: The one-wayness of the function f follows from IDLC. Using an extension of Theorem 2 (see subsection 2.6) the corollary follows. □

4. CONCLUDING REMARKS

We have shown a construction that when applied to a weakly regular one-way function yields a function which is one-way on its iterates. We believe that our construction may have the same affect when applied to arbitrary one-way functions. However, such a result cannot be proven by a better analysis of the game presented in subsection 2.4. (Consider for example a one-way function with a smoothly decreasing histogram. Then, after t iterations in the corresponding game all but approximately $1/t$ of the non-empty cells will be occupied with balls much heavier than the cell's size.) A possible line of attack is to try to take advantage of the nature of a polynomial time algorithm trying to invert the function on various inputs, and in particular with different hash functions.

ACKNOWLEDGEMENTS

We are grateful to Josh (Cohen) Benaloh, Leonid Levin, Charlie Rackoff, Ronny Roth and Avi Wigderson for very helpful discussions concerning this work.

The first author wishes to express special thanks to Leonid Levin and Silvio Micali for infinitely many discussions concerning pseudorandom generators.

REFERENCES

[ACGS] W. Alexi, B. Chor, O. Goldreich and C.P. Schnorr, "RSA and Rabin Functions: Certain Parts Are As Hard As the Whole", *SIAM Jour. on Computing*, Vol. 17, 1988, pp. 194-209.

[BBS] L. Blum, M. Blum and M. Shub, *A Simple Secure Unpredictable Pseudo-Random Number Generator*, *SIAM Jour. on Computing*, Vol. 15, 1986, pp. 364-383.

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

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

[CG] Chor, B., and O. Goldreich, "On the Power of Two-Point Sampling", to appear in *Jour. of Complexity*.

[CGG] Chor, B., O. Goldreich, and S. Goldwasser, "The Bit Security of Modular Squaring Given Partial Factorization of the Modulos", *Advances in Cryptology - Crypto 85 Proceedings*, ed. H.C. Williams, Lecture Notes in Computer Science, 218, Springer Verlag, 1985, pp. 448-457.

[DH] W. Diffie, and M. E. Hellman, "New Directions in Cryptography", *IEEE transactions on Info. Theory*, IT-22 (Nov. 1976), pp. 644-654

[E] S. Even, *Graph Algorithms*, Computer Science Press, 1979.

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

[GL] Goldreich, O., and L.A. Levin, "A Hard-Core Predicate for any One-Way Function", in preparations.

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

[J] A. Joffe, "On a Set of Almost Deterministic k-Independent Random Variables", *the Annals of Probability*, 1974, Vol. 2, No. 1, pp. 161-162.

[L] L.A. Levin, "One-Way Function and Pseudorandom Generators", *Combinatorica*, Vol. 7, No. 4, 1987, pp. 357-363. A preliminary version appeared in *Proc. 17th STOC*, 1985, pp. 363-365.

[L2] L.A. Levin, "Homogenous Measures and Polynomial Time Invariants", these proceedings.

[Lu] M. Luby, "A Simple Parallel Algorithm for the Maximal Independent Set Problem", *SIAM J. Comput.*, Vol. 15, No. 4, Nov. 1986, pp. 1036-1054.

[LR] M. Luby and C. Rackoff, "How to Construct Pseudorandom Permutations From

Pseudorandom Functions", *SIAM Jour. on Computing*, Vol. 17, 1988, pp. 373-386.

[McS] McWilliams, F.J., and N.J.A. Sloane, *The Theory of Error Correcting Codes*, North-Holland Publishing Company, 1977.

[R] M.O. Rabin, "Digitalized Signatures and Public Key Functions as Intractable as Factoring", MIT/LCS/TR-212, 1979.

[RSA] R. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public Key Cryptosystems", *Comm. ACM*, Vol. 21, Feb. 1978, pp 120-126

[S] A. Shamir, "On the Generation of Cryptographically Strong Pseudorandom Sequences", *ACM Transaction on Computer Systems*, Vol. 1, No. 1, February 1983, pp. 38-44.

[Y] Yao, A.C., "Theory and Applications of Trapdoor Functions", *Proc. of the 23rd IEEE Symp. on Foundation of Computer Science*, 1982, pp. 80-91.

Appendix A: One-way functions which are not one-way on their iterates

Assuming that f is a (regular) one-way function, we construct a (regular) one-way function $\bar{f}$ which is easy to invert on the distribution obtained by iterating $\bar{f}$ twice. Assume for simplicity that f is length preserving (i.e. $|f(x)| = |x|$). Let $|x| = |y|$ and let

$$\bar{f}(xy) = 0^{|y|} f(x)$$

Clearly, $\bar{f}$ is one-way. On the other hand, for every $xy \in \{0,1\}^{2n}$, $\bar{f}(\bar{f}(xy)) = 0^n f(0^n)$ and $0^n f(0^n) \in \bar{f}^{-1}(0^n f(0^n))$.

Appendix B: A survey of Levin's work [L]

The Main Theorem stated in the introduction is proved by combining Theorem 2 with the following result of Levin.

Theorem [L]: *If there exists a function which is one-way on iterates then there exists a pseudorandom generator.*

In the sequel we provide a sketch of Levin's proof [L], which follows Yao's argument [Y]. The proof consists of four parts, and is sketched here since the original proof is complete except for "obvious details" (i.e. obvious to Levin).

Levin starts with a function f which is one-way on its iterates. The first two steps in his argument are intended to achieve a function f_2 together with a polynomial time computable predicate b_2 so that $b_2(z)$ is hard to predict (better than 50-50) given $f_2(z)$, where z is taken from a distribution obtained by applying f_2 several times. (An alternative argument has been recently presented in [GL].)

First, it is shown that the function f can be slightly modified into a function f_1 so that there exist a predicate b_1 satisfying the following two conditions: (1) the predicate is polynomial time computable (i.e. there exist a polynomial time algorithm A such that $A(x) = b_1(x)$); but (2) the predicate cannot be efficiently 0.99-approximated from the value of the function (i.e. every polynomial time algorithm A' trying to guess $b_1(x)$ from $f_1(x)$ has success probability $Prob(A'(f_1(x)) = b_1(x)) \leq 0.99$, where x is taken from a distribution generated by repeated applications of f). For this argument, Levin uses an error-correcting code which can correct a constant fraction (say 4%) of errors and has polynomial time encoding and decoding algorithms. Let $C(x)$ denote the codeword of the information word x, and let $1 \leq i \leq |C(x)|$. Define $f_1(i,x) = (i, f(x))$ and $b_1(i,x)$ as the i-th bit of $C(x)$. Clearly, f_1 is one-way and b_1 is polynomial time computable. Now, assume that A' guesses $b_1(i,x)$ correctly with probability 0.99, when the probability is taken over all choices of i and x. Then at least half the probability mass of the x's is concentrated on x's for which A' has success probability ≥ 0.98. Running A' on $(1, f(x)), (2, f(x)), \ldots, (|C(x)|, f(x))$, with probability $\geq 1/2$, we obtain at least 0.96 of the $b_1(i,x)$'s. Applying the decoding algorithm, in

this case, we retrieve x, contradicting the one-wayness of f. This concludes the first part of the proof, strengthening Yao's argument that f has a bit b such that $b(x)$ cannot be approximated (given $f(x)$) better than with probability $1-\frac{1}{|x|}$.

Second, it is shown that by applying f_1 in parallel sufficiently many times and XORing the corresponding values of b_1 one gets a function f_2 and a predicate b_2 such that $b_2(y)$ can be efficiently guessed from $f_2(y)$ only with probability negligibly close to half. (This statement is known as "Yao's XOR technique" and its proof is quite involved.) In this case the number of copies should only be more than logarithmic, since we started with a predicate which cannot be guessed better than with some constant probability.

Next, one applies the construction (and reasoning) of Blum and Micali [BM] to obtain a generator which outputs bit sequences which cannot be efficiently predicted. More specifically, $G(y)=\sigma_1\cdots\sigma_{2|y|}$, the output of the generator on input y, is a $2|y|$-bit long string where $\sigma_i=b_2(f_2^{2|y|-i}(y))$. Ability to predict the i-th bit of the output given the previous bits is translated to approximating $b_2(z)$ given $f_2(z)$, where z is taken from the probability distribution induced by applying f_2 iteratively $2|y|-i$ times.

Finally, one uses Yao's theorem [Y] (see also [GGM]) that an ensemble is unpredictable if and only if it is pseudorandom.

It is important to note that the length of the argument to the functions, constructed throughout Levin's proof, increases while the number of iterations on which these functions are guaranteed to be one-way remains as in the original function f. Let k be the length of the argument to the original function. Then the argument to f_1 has length $k+O(\log k)=O(k)$ and the length of the argument to f_2 is $O(k\cdot(\log k)^2)$. When using the Blum and Micali construction we need to be able to apply the function f_2 a number of times which is twice the length of its argument. The function f_2 needs only be one-way for this number of iterations. Thus, if f is one-way on its first $k\cdot(\log k)^2<k^{3/2}$ iterates then we are done.

Remark: The above argument gives a pseudorandom generator which expands k-bit strings to $2k$-bit strings. Using such a pseudorandom generator, one can construct a pseudorandom function [GGM], which clearly yield a pseudorandom generator which expands k-bit strings to $p(k)$-bit strings, for an arbitrary polynomial p.

Appendix C: Announcement of a New Result [GL]

A central tool for the construction of pseudorandom generators and secure encryption functions is the "hard-core predicate of a function". This is a predicate $b(x)$ which is polynomial time computable given x, but cannot be efficiently approximated (i.e. guessed better than 50-50) given only the value of the function $f(x)$.

Assuming the existence of one-way functions, it is known how to construct such predicates. The known construction, due to Yao, uses the one-way function f to construct a more complicated one-way function which has a hard-core predicate [Y]. The construction applies the original one-way function f many times just to get one "hard-core" bit, and thus is undesirable from a practical point of view.

A new result of Goldreich and Levin [GL] shows that every one-way function has a hard-core predicate, and thus there is no need to apply the function many times in order to get one hard-core bit. More specifically, it is shown that the exclusive-or of a random subset of the bits of x is "a hard-core" predicate of $f(x)$, provided that f is a one-way function. The result extends to $\log|x|$ such subsets and to any distribution on the x's for which f is hard to invert.