On the Existence of Pseudorandom Generators

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 and Yao) 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. 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 functions 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. Other applications are the construction of pseudorandom generators based on the conjectured intractability of decoding random linear codes, and on the assumed average case difficulty of combinatorial problems as subset-sum.

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 $2k$-bit, sequence so that for every polynomial time

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.

S. Goldwasser (Ed.): Advances in Cryptology - CRYPTO '88, LNCS 403, pp. 146-162, 1990.
© Springer-Verlag Berlin Heidelberg 1990

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 exponentiation 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.

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 (*)

$$(\forall i, 1 \le i < k^{c+2}) \quad Prob\left[A(f^{(i)}(x), 1^k) \notin f^{-1}(f^{(i)}(x)) \right] > k^{-c},$$

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 paper 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 use the following definition.

Definition 2: A function f is called *regular* if there is a function $m(\cdot)$ such that for every n and for every $x \in \{0,1\}^n$ the cardinality of $f^{-1}(f(x)) \cap \{0,1\}^n$ is $m(n)$.

Clearly, *every* 1-1 function is regular (with $m(n)=1$, $\forall n$). Our main result is

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

A special case of interest is of 1-1 one-way functions. The sufficiency of these functions for constructing pseudorandom generators does not follow from previous works. In particular, Yao's result concerning one-way permutations does not extend to 1-1 one-way functions.

---

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

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.

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 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 construct pseudorandom generators based on the (widely believed) conjecture that decoding random linear codes is intractable, and on the assumed average case difficulty of combinatorial problems as subset-sum.

The main theorem is proved essentially 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-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).*

1.3. Subsequent Results

Recent results of Impagliazzo, Levin and Luby extend our results in two directions [ILL]. First, they generalize the regularity condition deriving a necessary and sufficient condition for the existence of pseudorandom generators. The new condition requires that the function f is one-way on a distribution induced by a function h, while the distribution induced by $f \circ h$ has almost the same entropy as the distribution induced by h. Second, they show that using non-uniform definitions of one-way functions and pseudorandom generator, yields their equivalence.

2. MAIN RESULT

2.0. 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 3: A polynomial time computable function $f : \{0,1\}^* \to \{0,1\}^*$ 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 (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.

2.1. Levin's Criterion: A Modified Version

The proof of the Main Theorem relies on the transformation of a function which is one-way and regular into a function which satisfies a variant of Levin's condition (i.e., being one-way on iterates). The modified condition, relating to functions which leave the first part of their argument unchanged, requires that the function is one-way on a number of iterates which exceeds the length of the second part of its argument. (Levin has required that the function is one-way on a number of iterations exceeding the length of the entire argument.) A precise statement can be found in Lemma 1 bellow. Before proving the sufficiency of the modified condition for constructing pseudorandom generators, we recall the basic ideas behind Levin's condition.

Levin's condition is motivated by Blum-Micali scheme for the construction of pseudorandom generators [BM]. This scheme uses two basic elements. The first, a (strongly) one-way function f, and the second, a boolean predicate $b(\cdot)$ called a "hard-core" of the function f. (Roughly speaking, a Boolean function $b(\cdot)$ is a *hard-core predicate of* f, if it is polynomial time computable, but no polynomial time probabilistic algorithm given $f(x)$, for randomly selected x, can compute the value of $b(x)$ with a probability significantly better than 1/2). A pseudorandom generator G is constructed in the following way. On input x (the seed), the generator G applies iteratively the one-way function $f(\cdot)$ on x for t $(=poly(|x|))$ times (i.e. $f(x), f^{(2)}(x), \ldots, f^{(t)}(x)$). In each application of f, the predicate $b(f^{(i)}(x))$ is computed and the resultant bit is output by the generator. That is, G outputs a string of length t. Blum and Micali show that the above sequence of bits is unpredictable when presented in reverse order (i.e. $b(f^{(t)}(x))$ first and $b(f^{(1)}(x))$ last), provided that the boolean function $b(\cdot)$ is a hard-core predicate on the distribution induced by the iterates $f^{(i)}, 0 \leq i \leq t$. The unpredictability of the sequence is proved by showing that an algorithm which succeeds to predict the next bit of the sequence with probability better than one half can be transformed into an algorithm for "breaking" the hard-core of the function f. Finally applying Yao's Theorem [Y] that unpredictable sequences are

pseudorandom we get that the above G is indeed a pseudorandom generator.

The hard part of the proof of Levin's Theorem (namely, that the existence of a function f being one-way on iterates implies the existence of pseudorandom generators) is in showing that the existence of a one-way function implies the existence of a hard-core predicate on the iterates of another function.[3] In order to construct this bit, the original function f is modified into a new one-way function f', and the hard-core predicate $b(\cdot)$ is constructed with respect to the new f'. The function $f'(x)$ consists of the parallel application of the original f on many copies, i.e. $f'(x_1,...,x_s)=(f(x_1),...,f(x_s))$. The x_i's are of equal size, say n, and Levin's construction uses a number of copies $s(n)$ which is any function that grows faster than $c\cdot\log n$, for any constant c. For constructing a pseudorandom generator, following Blum-Micali scheme, f' should be iterated on a seed of length k for at least $k+1$ iterations [4]. Recall that the seed has the form $(x_1,\ldots,x_{s(n)})$. Thus in order to have f' which is one-way for $n\cdot s(n)+1$ iterations, we need that the original function f is one-way for this number of iterations when applied to the substrings x_i of length n. Let $\tau(n)$ be a function which is an upper bound on the function $n\cdot s(n)+1$. For simplicity we may assume $\tau(n)=n^2$. Thus, we get that in order to construct a pseudorandom generator it suffices to have a function f which is strongly one-way for $\tau(n)$ iterations when applied to strings of length n. This is Levin's sufficient (and necessary) condition for the existence of pseudorandom generators. (Observe that Levin's condition as presented in section 1.1 refers to weak one-way functions, and then a greater number of iterations is required).

In our work we use the concept of "one-wayness on iterates" in a slightly modified way. We consider a function $F(\cdot,\cdot)$ defined as

$$F(h,x)=(h,F_0(h,x)) \qquad (*)$$

That is, F applies a function F_0 on its arguments and concatenates the first argument h to this result. The advantage of considering this kind of functions is that in order to construct a pseudorandom generator based on this function, it suffices to require that the function F is strongly one-way for $\tau(|x|)$ iterations, instead of the $\tau(|h|+|x|)$ iterations required by the straightforward application of Levin's result. The way we prove the sufficiency of this condition is as follows. First, we use Levin's modification of the function F into a new function $F'(h',x')$ for which a hard-core predicate does exist. (This is the same as the transformation from f to f' in the above description of Levin's construction). An important and simple observation is that F' preserves the form (*). Then, the

3) This part of the proof can be avoided using a recent result of Goldreich and Levin [GL]. This result states that any function $f'(x,r)=(f(x),r)$, where $|x|=|r|$, has a hard-core predicate for the uniform distribution on r and any distribution on x for which f is one-way.

4) A notable property of pseudorandom generators is that in order to have a generator which expands strings to any polynomial length, it suffices to construct a generator which expands strings of length k into strings of length $k+1$. This generator can be iteratively applied for polynomially many times without harming the pseudorandomness of its output [GrM].

function F' is applied by the generator G for at least $|x'|+1$ iterations. Note that F' remains one-way for all these iterations, as the original $F(h,x)$ is one-way for $\tau(|x|)$ iterates, and then $|x'|+1$ pseudorandom bits can be computed by using the hard-core of F'. The output of G will be the string h' concatenated with the above $|x'|+1$ pseudorandom bits. That is G expands its seed into a string which is at least one bit longer. The pseudorandomness of the output string is proved by noting that it is unpredictable. This is true for the h' part because it was chosen as a truly random string, and true for the other bits as guaranteed by Blum-Micali scheme. Namely, the ability to predict any of these bits would compromise the security of the hard-core of F'. The fact that the string h' is output do not help the predictor because the hard-core predicate of F' is unapproximable even when given h'. Recall that when given $F'(h',x')$ the string h' is explicitly presented.

Summarizing we get the following Lemma.

Lemma 1: Let $\tau(n)=n^2$. A sufficient condition for the existence of a pseudorandom generator is the existence of a function F of the form

$$F(h,x)=(h,F_0(h,x))$$

such that F is strongly one-way for $\tau(|x|)$ iterations.

2.2. Main Ideas

We prove the Main Theorem by transforming any regular and (strongly) one-way function into a new strongly one-way function f' for which the conditions of Lemma 1 hold.

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 (and the application of f) 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. The first part of the input is the description of hash functions that implement the "random" redistributions and the other part is interpreted as the input for the original function f.

The following is the definition of the function f':

$$f'(h_0, \cdots, h_{t(n)-1}, i, x) = (h_0, \cdots, h_{t(n)-1}, i^+, h_i(f(x)))$$

where $x \in \{0,1\}^n$, $h_j \in H(n)$, $0 \le i \le t(n)-1$. The function $t(n)$ is a polynomial in n, and i^+ is defined as $(i+1) \bmod t(n)$.

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 $t(n)$ iterations on strings x of length n.

Our Main Theorem follows from Theorem 2 and Lemma 1 by choosing $t(n) \ge \tau(n)$.

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, for $t(n)$ a polynomial, 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 $p(n) = \left[\frac{n \cdot t(n)}{M}\right]^{n-1}$.

Proof: 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 $h_0, h_1, \cdots, h_{t(n)-1}$ 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 exists 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. Fix a partition Π. Observe that every B, $|B| \geq n'$, for which $\xi_B(\Pi)=1$, contributes at least $\binom{n'}{n}$ *different* subsets A of size n for which $\chi_A=1$. Thus we get that

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

Dividing both sides of this inequality by $\binom{n'}{n}$, and averaging according to the probability distribution on the partitions Π, the left hand side is an upper bound for $q(n')$, while the right hand side is bounded above by $\frac{\binom{|S|}{n} \cdot p(n)}{\binom{n'}{n}}$. □

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 $4t(n) \cdot n^2+n$ balls which end up in the same cell is bounded above by 2^{-n}.

Proof: Let S be the set of M balls in the above game. Each game defines a partition of the balls according to their position after $t(n)$ iterations. The probability distribution on these partitions is induced by the uniform choice of the mappings h. Theorem 5 follows by using Lemma 4 with $n'=4t(n) \cdot n^2+n$, and the bound $p(n)$ of Lemma 3. □

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 $f^{-1}(z_i)$ (which we have denoted by $m(n)$) 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 in an analogous way. 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. □

The meaning of Lemma 7 is that the function f is hard to invert on the distribution induced by the functions g_r, $r=1,\ldots,t(n)$, thus proving the strong one-wayness of the function f' for $t(n)$ iterations. Theorem 2 follows.

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: For every n, 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. Alternatively, we can transform the above f into a length preserving and regular function $\hat{f}$ by defining $\hat{f}(xy)=\hat{f}(x)$, where $|x|=n$, $|y|=n'-n$.

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.

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

Definition 4: 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 5: The function f is *weakly regular* if there is a polynomial $p(\cdot)$ and a function $b(\cdot)$ such that the histogram of f satisfies (for all n)

i) $hist_f(n,b(n)) \geq \dfrac{2^n}{p(n)}$

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

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.

Observe that if the $b(n)$-th level of the histogram contains all of the 2^n strings of length n then we can apply a similar analysis as done for the regular case. The only difference is that we have to analyze the game of subsection 2.4 not for cells of equal size, but for cells that differ in their size by a multiplicative factor of at most two. Similar

arguments hold when considering the case where the $b(n)$-th level of the histogram contains at least $1/p(n)$ of the strings and the rest of strings lie below this level (i.e. $hist_f(n,k)=0$, for $k>b(n)$). Note that the "small" balls of low levels cannot cause the cells of the $b(n)$-th level to grow significantly. On the other hand, for balls bellow level $b(n)$ nothing is guaranteed. Thus, we get that in this case the function f' we construct is weakly one-way on its iterates. More precisely, it is hard to invert on its iterates for at least a $1/p(n)$ fraction of the input strings. In order to use this function for generating pseudorandom bits, we have to transform it into a strongly one-way function. This is achieved following Yao's construction [Y] by applying f' in parallel on many copies. For the present case the number of copies could be any function of n which grows faster than $c \cdot p(n) \cdot \log n$, for any constant c. This increases the number of iterations for which f' has to remain one-way by a factor equal to the number of copies used in the above transformation. That is, the number $t(n)$ of necessary iterates increases from the original requirement of $\tau(n)$ (see section 2.1) to a quantity which is greater than $c \cdot p(n) \cdot \tau(n) \cdot \log n$, for any constant c. Choosing this way the function $t(n)$ in the definition of f' in section 2.3, we get f' which is one-way for the right number of iterations.

Finally, consider the case in which there exist strings above the $b(n)$-th level. When considering the game of subsection 2.4 we want to show that, also in this case, most of the cells of the $b(n)$-th level do not grow considerably. This is guaranteed by condition (ii) in Definition 5. Consider the worst case possibility in which in every iteration the total weight of the "big" balls (those above level $b(n)$) is transferred to cells of the $b(n)$-th level. After $t(n)$ iterations this causes a concentration of "big" balls in the $b(n)$-th level having a total weight of at most $t(n) \cdot \frac{2^n}{(n \cdot p(n))^2}$. Choosing $t(n)=\frac{1}{2}p(n)n^2$ this weight will be at most $\frac{2^n}{2p(n)}$. But then one half of the weight in the $b(n)$-th level remains concentrated in balls that were not effected by the "big" balls. In other words we get that the function f' so constructed is one-way for $t(n)$ iterations on $\frac{1}{2p(n)}$ of the input strings. Applying Yao's construction , as explained above, we get a function f' which fill the criterion of Lemma 1 and then suitable for the construction of pseudorandom generators.

Further Remarks:

1) A finer analysis allows to substitute the exponent 2, in condition (ii) of Definition 5, by any constant greater than 1.

2) The entire analysis holds when defining histograms with polynomial base (instead of base 2). Namely, $hist_f(n,k)$ is the cardinality of the set

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

where $Q(n)$ is a polynomial.

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 computational problems is "hard on a non-negligible fraction of the instances".

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

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

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. □

3.3. PRG Based on the Average Difficulty of Combinatorial Problems

Some combinatorial problems which are believed to be hard on the average can be used to construct a regular one-way function and hence be a basis for a pseudorandom generator. Consider, for example, the *Subset-Sum Problem.*

Input: Modulo M, $|M|=n$, and $n+1$ integers $a_0,a_1,\cdots,a_n$ of length n-bit each.

Question: Is there a subset $I\subseteq\{1,\ldots,n\}$ such that $\sum_{i\in I}a_i\equiv a_0(\mathrm{mod}\,M)$

Conjecture: The above problem is hard on the average, when the a_i's and M are chosen uniformly in $[2^{n-1},2^n-1]$.

Under the above conjecture, the following weakly-regular function is one-way

$$f_{SS}(a_1,a_2,\cdots,a_n,M,I)=(a_1,a_2,\cdots,a_n,M,(\sum_{i\in I}a_i \bmod M))$$

ACKNOWLEDGEMENTS

We are grateful to Josh (Cohen) Benaloh, Manuel Blum, Leonid Levin, Richard Karp, Charles 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. Goldrcich 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. Sñub, *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, YT-22* (Nov. 1976), pp. 644-654

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

[GGM] Goldreich, 0., S. Goldwasser, and S. Micali, "How to Construct Random Functions", *Jour, ofACM,* Vol. 33, No. 4,1986, pp. 792-807.

[GKL] Goldreich, O., H. Krawczyk and M. Luby, "On the Existence of Pseudorandom Generators", *Proc. 29th IEEE Symp. on Foundations of Computer Science,* 1988, pp 12-24.

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

[GrM] Goldreich, O., and S. Micali, "The Weakest Pseudorandom Bit Generator Implies the Strongest One", manuscript, 1984.

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

[ILL] Impagliazzo, R., L.A., Levin and M.G. Luby, "Pseudorandom Number Generation from any One-Way Function", preprint, 1988.

[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, "Homogeneous Measures and Polynomial Time Invariants", *Proc. 29th lEEESymp. on Foundations of Computer Science,* 1988, pp 36-41.

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

[LR] M. Luby and C. Rackoff, "How to Construct Pseudorandom Permutations From Pseudorandom Functions", *SIAMJour. on Computing,* Vol. 17,1988, pp. 373-386.

[McS] McWilliams, FJ., 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 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 $0^n f(0^n) \in \bar{f}^{-1}(0^n f(0^n))$.