SIAM J. COMPUT.
Vol. 22, No. 6, pp. 1163–1175, December 1993

003

ON THE EXISTENCE OF PSEUDORANDOM GENERATORS*

ODED GOLDREICH†, HUGO KRAWCZYK‡, AND MICHAEL LUBY§

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 that is indistinguishable in polynomial time from an (equally long) sequence of unbiased coin tosses. A fundamental question is to find simple conditions, as the existence of one-way functions, which suffice for constructing pseudorandom generators. This paper considers *regular* functions, in which every image of a k-bit string has the same number of preimages of length k. This paper shows how to construct pseudorandom generators from any regular one-way function.

Key words. pseudorandom generators, one-way functions, cryptography, randomness, complexity theory

AMS subject classifications. 11K45, 11T71, 68Q99, 94A60

1. Introduction.

In recent years, randomness has become a central notion in the theory of computation. It is used heavily 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 that 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 algorithm (distinguishing test) T, any constant $c > 0$, and sufficiently large k

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

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

*Received by the editors April 20, 1989; accepted for publication (in revised form) July 1, 1992. A preliminary version of this paper was presented at the IEEE 29th Annual Symposium on Foundations of Computer Science, 1988.

†Department of Computer Science, Technion, Haifa 32000, Israel. This author was supported by grant 86-00301 from the United States-Israel Binational Science Foundation (BSF), Jerusalem, Israel.

‡Department of Computer Science, Technion, Haifa 32000, Israel. Present address, IBM T. J. Watson Research Center, P. O. Box 704, Yorktown Heights, New York 10598.

§International Computer Science Institute, 1947 Center Street, Berkeley, California 94704-1105, and Department of Mathematics, University of California, Berkeley, California 94720. A large portion of the author's research contribution to this work was done while visiting the Computer Science Department of the Technion. At the time, the author was a faculty member at the University of Toronto. This author's research was partially supported by Natural Sciences and Engineering Research Council of Canada operating grant A8092, by a University of Toronto grant, by National Science Foundation grant CCR-9016468, and by grant 89-00312 from the United States-Israel Binational Science Foundation (BSF), Jerusalem, Israel.

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). On the other hand, 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 $f : \{0, 1\}^* \to \{0, 1\}^*$ 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 that 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 nonnegligible 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(*)

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

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 that 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

[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.

function is one-way on its iterates. Furthermore, it has been an open question whether or not Levin's condition is equivalent to the mere existence of one-way functions.

1.2. Our results. In this paper, we consider "regular" functions, in which every element in the range has the same number of preimages. We show how to construct pseudorandom generators from any regular one-way function.

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 the following theorem.

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.

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 nonnegligible 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 emphasize 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 novelty of this work is in presenting *a direct way to construct a function that 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 [ILL] and Hastad [H], inspired by the current work, has resolved the problem of equivalence of existence of one-way functions and pseudorandom generators, in the affirmative. However, in light of the inefficiency of their construction, some of the ideas presented in the current work may be useful in future attempts to construct more efficient pseudorandom generators from one-way functions.

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., for all x, $|f(x)| = |x|$). Our results hold also without this assumption (see §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 one-way on iterates condition. The modified condition relates to functions, which leave the first part of their argument unchanged. It requires that the function is one-way on a number of iterates, which exceeds the length of the second part of its argument. (Levin requires that the function is one-way on a number of iterations exceeding the length of the *entire* argument.)

More precisely, we consider functions $F(\cdot, \cdot)$ defined as

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

That is, F applies a function F_0 on its arguments and concatenates the first argument h to this result. We prove the following condition.

LEMMA 1. *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 $|x| + 1$ iterations.

Before proving Lemma 1, let us recall the 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 probablility significantly better than $\frac{1}{2}$.) 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 crucial ingredient in the proof of Levin's condition, as well as of our modified version, is the existence of a hard-core predicate for any (slightly modified) one-way function. A recent

result of Goldreich and Levin [GL] greatly simplifies the original proof in [L]. 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 strongly one-way. This hard-core predicate is the inner product modulo 2 of r and x (viewed as vectors over Z_2).

Finally, we recall the following notable property of pseudorandom generators: in order to have a generator that expands strings to any polynomial length, it suffices to construct a generator that 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]. We now prove Lemma 1.

Proof of Lemma 1. Note that $F^{(i)}(h, x) = (h, F_0^{(i)}(h, x))$. Thus, the condition in the lemma implies that $F_0(h, x)$ is hard to invert for $|x| + 1$ iterations even when h is given to the inverter. We construct the following generator, G, which expands its input by one bit. Let s be the seed for G, so that $s = (\bar{r}, h, x)$, where $|x| = n$, $\bar{r} = r_0, \ldots, r_n)$, and for all i, $|r_i| = n$. Then, we define

$$G(s) = G(\bar{r}, h, x) = (\bar{r}, h, b_0, \ldots, b_n),$$

where $i = 0, \ldots, n$, b_i is the inner product modulo 2 of r_i and $F_0^{(i)}(h, x)$. (We denote $F_0^{(0)}(h, x) = x$.)

We claim that this generator is pseudorandom. This is proved by noting that the output string is unpredictable. This is true for the $\bar{r}$ and h part as they were chosen as truly random strings. For the other bits, this is guaranteed by the Goldreich–Levin result and the fact that F_0 is hard to invert for $n + 1$ iterations (even when h is given to the inverter). □

2.2. Main ideas. We prove the Main Theorem by transforming any regular and (strongly) one-way function f 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 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 that 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 a single 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 that 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:

$$F(h_0, \ldots, h_{t(n)-1}, i, x) = (h_0, \ldots, h_{t(n)-1}, i^+, h_i(f(x))),$$

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 $t(n) > n$.

Let $h_0, h_1, \ldots, h_{t(n)-1}$ be $t(n)$ functions from the set $H(n)$. For $r = 1, \ldots, t(n)$, let g_r be the function $g_r = fh_{r-1}fh_{r-2}f \ldots h_0 f$ acting on strings of length n, let $G_r(n)$ be the set of all 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 $r = 1, \ldots, t(n)$. 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, over $h_0, h_1, \ldots, h_{t(n)-1} \in_R H(n)$, that a specific set of n balls is mapped after $t(n)$ iterations to the same cell is bounded above by $p(n) = [(n \cdot t(n))/M]^{n-1}$.*

Proof. Let $B = \{b_1, b_2, \ldots, b_n\}$ 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 $h_0, h_1, \ldots, h_{t(n)-1}$, 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 $h_0, h_1, \ldots, h_{t(n)-1}$ map the balls of B to the same cell, then $b_1, b_2, \ldots, b_n$ 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, which we call the *join 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 since $(t(n)/M$ 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, §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 phenomenon 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 probability that $\chi_A = 1$, for any subset $A \subseteq S$ of size n. Let $q(n')$ be 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 $\binom{|S|}{n} \cdot p(n)/\binom{n'}{n}$. □

Remark. Lemma 4 is useful in situations when the ratio $p(n)/p(n')$ is smaller than $(|S| - n \quad n' - n)$. 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 Lemmas 3 and 4, we get the following theorem.

THEOREM 5. *Consider the game played for $t(n)$ iterations. Then, the probability that there are $4t(n) \cdot n^2 + n$ 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. (We also use the fact that $M \leq 2^n$ and the binomial bound $(n' \quad n) \geq (n'/n - 1)^n$.) □

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 $g = fh_{t(n)-1}f \ldots h_0 f$. The functions $h = h_j$ are hash functions used to map the range of f to the domain of f. We let $h_0, \ldots, h_{t(n)-1}$ 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 §2.4; and thus, Theorem 5 can be applied to obtain the next lemma. (Notice that $g \in_R G(n)$ means choosing $t(n)$ functions $h_0, \ldots, h_{t(n)-1} \in_R H(n)$ and putting $g = fh_{t(n)-1}f \ldots h_0 f$.)

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 $g \in_R G(n)$.

Note. The constant c_0 depends on the degree of $t(n)$. More precisely, we need $n^{c_0} \geq 4t(n) \cdot n^2 + n$ (see Thm. 5).

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 Lemma 6, $G'(n)$ 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 $r = 1, \ldots, t(n)$ 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 $r = 1, \ldots, t(n)$,*

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

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 $r = 1, \ldots, t(n)$ 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 nonnegligible probability. By assumption there is a nonnegligible 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 nonnegligible 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 techniques 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 $\hat{f}(xy) = f(x)$, where $|x| = n$, $|y| = n' - n$.

For the applications in §3, and possibly for other cases, the following extension (referred to as *semiregular*) is useful. Let $\{f_x\}_{x\in\{0,1\}^*}$ be a family of regular functions, then our construction can still be 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

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

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 $k = \lfloor \log_2(m(n)) \rfloor$ 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) $hist_f(n, b(n)) \geq \frac{2^n}{p(n)}$

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

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. Details follow.

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 §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 below 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 $n + 1$ (see §2.1) to a quantity that is greater than $c \cdot p(n) \cdot n \cdot \log n$, for any constant c. Choosing this way the function $t(n)$ in the definition of F in §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 §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 $t(n) \cdot 2^n/(n \cdot p(n))^2$. Choosing $t(n) = \frac{1}{2}p(n)n^2$, this weight will be at most $2^n/(2p(n))$. But then one-half of the weight in the $b(n)$th level remains concentrated in balls that were not affected by the big balls. In other words, we get that the function F so constructed is one-way for $t(n)$ iterations on $1/(2p(n))$ of the input strings. Applying Yao's construction, as explained above, we get a function F, which satisfies the criterion of Lemma 1 and is then suitable for the construction of pseudorandom generators.

Further Remarks.

1. The denominator in condition (ii) of Definition 5 can be substituted by any function growing faster than $c \cdot p^2(n) \cdot n$, for any constant c. This follows from the above analysis and the fact that the construction of a hard-core predicate in [GL] allows extracting $\log n$ secure bits with each application of the one-way function.

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

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

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 (PRGs) based on the assumption that one of the following computational problems is "hard on a nonnegligible 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 nonnegligible 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; 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 nonnegligible 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 semiregular function (see §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

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

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 semiregular. The one-wayness of the function follows from IGF (using Rabin's argument [R]). Using an extension of Theorem 2 (see §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, Chap. 1, p. 34]. The same argument can be used to show (for every $\epsilon > 0$) that if $k/n < 1 - H_2((1+\epsilon)\cdot d/n)$, 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 let $e \in E_t^n \subseteq \{0,1\}^n$ 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 code word of x (i.e., $C(x)$ is the vector resulting by the matrix product xC). One can easily verify that f just defined is semiregular (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 code word 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

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

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 §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 $a_0, a_1, \ldots, a_n$ of length n-bit each.

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

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

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

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 $0^n f(0^n) \in \bar{f}^{-1}(0^n f(0^n))$.

Acknowledgments. 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 the infinite number of 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 J. Comput., 17 (1988), pp. 194–209.

[BBS] L. Blum, M. Blum, and M. Shub, *A simple secure unpredictable pseudo-random number generator*, SIAM J. Comput., 15 (1986), pp. 364–383.

[BM] M. Blum and S. Micali, *How to generate cryptographically strong sequences of pseudo-random bits*, SIAM J. Comput., 13 (1986), pp. 850–864.

[CW] J. Carter and M. Wegman, *Universal classes of hash functions*, JCSS, 18 (1979), pp. 143–154.

[CG] B. Chor and O. Goldreich, *On the power of two-point sampling*, J. Complexity, 5 (1989), pp. 96–106.

[CGG] B. Chor, O. Goldreich, and S. Goldwasser, *The bit security of modular squaring given partial factorization of the modulos*, Adv. Cryptology – Crypto 85 Proceedings, H. C. Williams, ed., Lecture Notes in Computer Science, 218, Springer-Verlag, 1985, pp. 448–457.

[DH] W. Diffie and M. E. Hellman, *New directions in cryptography*, IEEE Trans. Inform. Theory, IT-22, Nov. 1976, pp. 644–654.

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

[GGM] O. Goldreich, S. Goldwasser, and S. Micali, *How to construct random functions*, J. Assoc. Comput. Mach., 33 (1986), pp. 792–807.

[GKL] O. Goldreich, H. Kawczyk, and M. Luby, *On the existence of pseudorandom generators*, Proc. 29th IEEE Symposium on Foundations of Computer Science, 1988, pp. 12–24.

[GL] O. Goldreich and L. A. Levin, *A hard-core predicate for any one-way function*, Proc. 21st Symposium on Theory of Computing, 1989, pp. 25–32.

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

[GM] S. Goldwasser and S. Micali, *Probabilistic encryption*, JCSS, 28 (1984), pp. 270–299.

[H] J. Hastad, *Pseudo-random generators under uniform assumptions*, Proc. 22nd Symposium on Theory of Computing, 1990, pp. 395–404.

[ILL] R. Impagliazzo, L. A. Levin, and M. G. Luby, *Pseudo-random generation from one-way functions*, Proc. 21st Symposium on Theory of Computing, 1989, pp. 12–24.

[J] A. Joffe, *On a set of almost deterministic k-independent random variables*, Ann. Probab., 2 (1974), pp. 161–162.

[L] L. A. Levin, *One-way function and pseudorandom generators*, Combinatorica, 7 (1987), pp. 357–363; a preliminary version appeared in Proc. 17th Symposium on Theory of Computing, 1985, pp. 363–365.

[L2] ———, *Homogeneous measures and polynomial time invariants*, Proc. 29th IEEE Symposium on Foundations of Computer Science, 1988, pp. 36–41.

[Lu] M. Luby, *A simple parallel algorithm for the maximal independent set problem*, SIAM J. Comput., 15 (1986), pp. 1036–1054.

[LR] M. Luby and C. Rackoff, *How to construct pseudorandom permutations from pseudorandom functions*, SIAM J. Comput., 17 (1988), pp. 373–386.

[McS] F. J. McWilliams and N. J. A. Sloane, *The Theory of Error Correcting Codes*, North-Holland Publishing Company, Amsterdam, 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, 21 (1978), pp. 120–126.

[S] A. SHAMIR, *On the generation of cryptographically strong pseudorandom sequences*, ACM Trans. Comput. Systems, 1 (1983), pp. 38–44.

[Y] A. C. YAO, *Theory and applications of trapdoor functions*, Proc. of 23rd IEEE Symposium on Foundation of Computer Science, 1982, pp. 80–91.