Security Preserving Amplification of Hardness

Oded Goldreich,[*] Technion
Russell Impagliazzo,[†] University of Toronto
Leonid Levin[‡](Lnd@cs.bu.edu), Boston University[§]
Ramarathnam Venkatesan,[¶] Boston University
David Zuckerman,[‖] U.C. Berkeley

Abstract

We consider the task of transforming a weak one-way function (which may be easily inverted on all but a polynomial fraction of the range) into a strong one-way function (which can be easily inverted only on a negligible fraction of the range). The previous known transformation [Yao 82] *does not preserve the security* (i.e., the running-time of the inverting algorithm) *within any polynomial.* Its resulting function $F(x)$ applies the weak one-way function to many small (of length $|x|^{\varepsilon}$, $\varepsilon<1$) pieces of the input. Consequently, the function can be inverted for reasonable input lengths by exhaustive search.

Using random walks on constructive expanders, we transform any regular (e.g., one-to-one) weak one-way function into a strong one, while *preserving security.* The resulting function $F(x)$ applies the weak one-way f to strings of length $\Theta(|x|)$. Our security preserving constructions yield *efficient* pseudo-random generators and signatures based on any regular one-way function.

1 Introduction

A central problem in the foundations of cryptography and other fields is that of relating various basic concepts and primitives such as one-way functions [Diffie, Hellman 76] pseudo random generators [Blum Micali 82, Yao 82] and signature schemes [Goldwasser Micali Yao 83]. Recently, [Hastad Impagliazzo Levin Luby 90] and [Rompel 90], demonstrated the equivalence of these notions[1]. However, the constructions used in these proofs are impractical and yield an equivalence only in a weak sense to be discussed below. We attempt here to present efficient constructions which yield a strong equivalence.

For simplicity, we continue the discussion with respect to one-way permutations (i.e.,

[*]Supported by grant #86-00301 by US-Israel Binational Science Foundation, Jerusalem, Israel.

[†]Supported by CCR-88-13632

[‡]Supported by NSF grant DCR-8607492, MIT and Sun Microsystems.

[§]Dept. of Computer Science, 111 Cummington Street, Boston MA 02215.

[¶]Supported by NSF grant DCR-8607492 and Bell Communications Research.

[‖]Supported by an NSF Graduate Fellowship.

[1]See also [Levin 87, Goldreich Krawczyk Luby 88, Goldreich Levin 89] for constructing pseudo random generators from one-way functions, and [Goldwasser Micali Rivest 84, Goldreich 87, Merkle 87, Bellare Micali 88, Naor Yung 89] for signature schemes.

length preserving 1-1 functions), though our results hold for a broader class.

A polynomial-time computable permutation is called *weakly* one-way if it is "infeasible" to invert it on some polynomial fraction of its range (so the permutation may be easy to invert almost always). Such a permutation is called *strongly* one-way if it is "infeasible" to invert it on all but a negligible fraction of its range. [Yao 82] showed that these two notions are equivalent, *if "infeasible" means non-polynomial time.* For this reason many papers make no distinction between these two notions. However, such an equivalence allows changes in the degree of infeasibility by more than any polynomial. Say, f is hard to invert on $1/n^3$ of the strings of length n. Then $F(x_1, ..., x_{n^4}) = f(x_1) \circ \cdots \circ f(x_{n^4})$ (where $|x_i| = n$) is strongly one-way (proving this is easy, but not trivial, see [Goldreich 89, pp. 20-24]). However, the difficulty of inverting F on inputs of length n^5 is comparable to the difficulty of inverting f on inputs of length n (not n^5)! In this sense the equivalence is weak. This construction allows the resulting function be inverted by exhaustive search over smaller pieces for reasonable input lengths. In practical terms, this means that F is hard to invert only on huge (impractical) inputs.

Several different resources determine the efficiency of a one-way function. Of obvious importance is its running time. However, often the *lengths* of the inputs and outputs needed to achieve this level of security are of equal importance. The bottleneck in cryptographic protocols is often memory or communication costs, rather than computation costs. Consider a private-key encryption system which uses a pseudo-random generator. The two parties agree on a seed for the generator in private, and use the pseudo-random string generated from this seed as a one-time pad to send secret messages over a public channel. Here, private communication is at a premium; computation is only of secondary importance. Even a relatively small increase in the input length can cause dramatic problems here. Just to be immune from attack by exhaustive search, at least 50 bits of input must be used. Using Yao's method on a weak $\frac{1}{n}$ one-way function, 125,000 bits would be required to achieve this minimal level of security. Since fewer bits of message will likely be sent between the parties, they might as well agree on a truly random one-time pad and not use any cryptography.

Another situation with both input and output lengths crucial is in protocols (say, for zero-knowledge) which use one-way permutations for bit commitment. Here, each bit committed to might require sending the value of the one-way function on a random input, and later revealing the input. Since the number of bits committed during the protocol might be large, the communication cost of one commitment is very important. Our constructions of strong one-way functions from weak ones dramatically improves the lengths involved to achieve a given level of security without affecting the running time.

This discussion can be further clarified by specifying a lower bound (called *security*) on the time considered infeasible[2]. [Yao 82]

[2]Polynomial time is an excellent formalization of feasibility: high degree polynomials are rare and unlikely to be the intrinsic complexity of fundamental problems. But super-polynomial time is an unreasonable notion of infeasibility. Say, $k^{(\ln \ln k)/3}$ is quite simple and may be the intrinsic complexity of a fundamental problem (like primality). While super-polynomial, it is less than k^2, for k up to the number of particles in the Universe. Reductions should preserve the security within a polynomial overhead rather than just preserve its super-polynomiality.

transforms a weakly one-way f with security $s(n)$, into a strong one-way function F with security $s(n^\varepsilon)$, $0<\varepsilon<1$, which is smaller than any constant power of s.

Our main result gets a strong one-way permutation with security $s(n)/n^{O(1)}$ out of any weak one-way permutation with comparable security $s(n+O(\log s(n)))$. The result extends to regular one-way functions (defined in Section 5). Our simple techniques yield efficient pseudo random generators and signature schemes from any regular one-way function. Only inefficient constructions were known before.

We use random walks on expanders, as in [Ajtai Komlos Szemeredi 87, Cohen Wigderson 89, Impagliazzo Zuckerman 89], and hence rely on their explicit constructions (see [Margulis 73, Gabber Galil 81, Lubotsky Sarnak Philips 86]). Expanders have found applications in many concrete algorithms. [Levin 88] noticed that expanders yield general results in the theory of computation as well. This paper provides further evidence of this.

2 Results

Definition 1 (one-wayness) : A polynomial time computable function f is $\alpha(n)$-*one-way* (OW) with security $s(n)$ if every randomized algorithm inverts f in time $t(|x|)|x|^{O(1)}$ on fraction $< 1-\alpha+t/s$ of inputs $f(x)$ and internal coin flips. Security is *strict* if the degree of the above polynomial $n^{O(1)}$ is fixed (normally to 0).

Taking the fraction α of hard instances as $n^{-O(1)}, 1/2$, or 1, we get *weak, frequent* or *strong* OW, respectively.

Security reflects the time required to verify by sampling the frequency of hard instances. Since we ignore polynomial factors in the definition of security, the underlying machine model is not crucial.

Permutations are length preserving 1-1 (one-to-one) functions.

We call $s(n)$ and $s(n)^{\Theta(1)}$ *comparable*. We take the security function to be monotone and *smooth*: $s(n+1) = O(s(n))$ or $s(2n) = s(n)^{O(1)}$. Then, $s(n)$ is the same security as $s(n+O(\log n))$ and comparable to $s(\Theta(n))$. These smoothness conditions are unnecessary but natural and simplify the statement of our main results:

Theorem 1 (weak→frequent): *For any $n^{-O(1)}$-OW permutation f, there exists a $1/2$-OW permutation F with the same security.*

Theorem 2 (frequent→strong): *For any $1/2$-OW permutation f with security $s(n) = s'(n+10\log s(n))$, there exists a 1-OW permutation with security $s'(n)$.*

Note that for non-increasing $\log s(n)/\sqrt{n}$, $s(n+O(\log s(n))) = O(s(n))$.

We generalize the above result to regular one-way functions used in [Goldreich Krawczyk Luby 88] for constructing pseudo-random generators. Many known candidates for one-way functions are regular: each point in the range of the function has the same number of inverses.

If a property holds for all except $\leq 1/s$ fraction of instances, we say it holds *almost everywhere* (abbreviated *a.e.*). We relax the definition of regular functions: one-way f is *regular* if there is a polynomial time computable upperbound $m(x)$ such that $|f^{-1}(f(x))| \leq 2^{m(x)}$ a.e. and $|f^{-1}(f(x))| \geq 2^{m(x)}/s(|x|)^{o(1)}$, on a polynomial fraction of hard-instances[3].

[3]There is an optimal (within constant factor) inversion algorithm and hard-instances are those on which it runs slowly. (See [Levin 85])

Let $H_{n,m}$ denote a class of universal hash functions [Carter, Wegman 79] mapping n-bit strings into m-bit strings. It is important for our use that each $h \in H_{n,m}$ has a description (denoted $\overline{h}$) of length $O(n+m)$ and $h(x)$ is polynomial time computable on input $\overline{h}$ and x. An example of such a class is the set of all m-by-n Toeplitz matrices on $GF(2)$ (a Toeplitz matrix $A = \{a_{i,j}\}$ satisfies $a_{i,j} = a_{i+1,j+1}$ and is specified by its first row and first column).

We take our regular functions to be length preserving within a constant factor. Otherwise, one can make them so using hashing as follows: $f(x, \overline{h}) = h(f(x)), \overline{h}$. We call a one-way function f with security s *almost length preserving* if $|f(x)| = |x| + O(\log s(n))$.

Theorem 3 *If a regular $n^{-O(1)}$-OW function of security $s(n) = s'(n + \Theta(\log(s(n)))^2)$ exists, then there is a 1-OW a.e. one-to-one function of security comparable to s'.*

Note that $s(n + O(\log s(n))^2) = s(n)^{O(1)}$, when $\log s(n) < \sqrt{n}$.

Corollaries: Given a regular one-way function, one can construct a pseudo-random generator of comparable security. Similarly using the results of [Naor Yung 89] one can construct an efficient signature scheme.

3 The Construction

We use constructive expanders, i.e. a family of fixed (say, d) degree expander graphs on an exponential size vertex set with a polynomial-time algorithm that on input a node outputs its adjacency list. Also, we require that the ordering of these incident lists induces d (disjoint) perfect matchings of the expander. Such expander families do exist. For simplicity, assume that the expander has 2^n nodes.

These explicit constructions enable us to perform a random walk on the expander. Such random walks rapidly (i.e. within $O(n)$ steps) reach a nearly uniform distribution on the vertices. The key property we need is that the adjacency matrix of the expanders have their second eigenvalue at most cd, for some constant $c < 1$ (see [Alon 86]). If we add a self-loop at each node, the second largest (in absolute value) eigenvalue will be well-separated from the first. We can achieve any constant ratio between the second eigenvalue and first, simply by raising the adjacency matrix to an appropriate power.

To amplify a one-way function f we would like to apply it iteratively many times. This will not help if easy instances for the inverting algorithm keep being mapped to themselves. The idea is to use randomization between successive applications of f. One may try using universal$_2$ hash functions as in [Goldreich Krawcyk Luby 88]. If one applies the *same* hash function between successive applications, the random path induced is not guaranteed to reach uniformly distributed locations and the same holds if a fixed number of random universal$_2$ hash functions are applied[4]. Instead we randomize the arguments to the different iterations of the one-way permutation by taking one random step on an expander. Namely, we associate the domain of the given one-way permutation with the vertex set of the expander. Our construction alternatively applies the one-way permutation and moves at random from the node reached to one of its neighbors. This requires very little randomization. A key observation is that the composition of an expander with any permutation on its vertex set yields an

[4] Proving that in general (i.e. on every d-regular graph), such a path reaches uniform distribution would imply that Rlogspace=Dlogspace [Naor 89].

expander (with the same eigenvalues). Using a Random Walk Lemma and a simulation argument, the construction is showed to amplify the one-wayness of the given permutation while preserving security.

Let G be an explicit undirected d-regular expander with vertex set $\{0,1\}^n$ with the second largest (in absolute value) eigenvalue $\lambda_2 < d/2$. For example, [Lubotsky Philips Sarnak 86] proposed an expander family with $d = 18$, $\lambda_2 \leq 2\sqrt{17}$ for some graph sizes. Other expander families exist, for graphs of size n^2.

Consider a labeling of the edges incident to each node (using the labels $1, 2, ..., d$) so that the mapping corresponding to each label induces a permutation on the vertex set. Let $g(x, l)$ be the node reachable from x by following the edge labeled l. For any permutation f on $\{0,1\}^n$ and every $k \geq 1$, $x \in \{0,1\}^n$, $\sigma_1, ..., \sigma_k \in \{1, 2, ..., d\}$, let $F_k(x, \sigma_1...\sigma_k) \stackrel{\text{def}}{=} \sigma_1 \circ F_{k-1}(g(f(x), \sigma_1), \sigma_2...\sigma_k)$ (and $F_0(x, \emptyset) \stackrel{\text{def}}{=} x$) and G_f be the expander induced by $g_f(x, l) = g(f(x), l)$.

4 Proofs

Proposition 1 *: Let G_f be an expander as above, $\mu(n) \geq 1/2$ and f be $1 - \mu(n)$-OW with strict security $s(n) = s'(n + k(n)\log_2 d)$. Then, $F_{k(n)}$ is $(1 - \mu^{k/2})$-OW with strict security $s'(n)/k(n)$.*

Proof of Theorems 1 and 2: Theorem 1 follows by applying this proposition iteratively $l = O(\log n)$ times: Use $k(n) = O(1)$ length walks such that each iteration converts a $(1-\mu(n))$-OW function to a $1 - \mu^k$-OW function. Note that the finally resulting function is $1 - \mu^{k^l}$-OW. Each iteration adds $O(1)$ to the input length and multiplies the time of computation and inverting time by $\Theta(1)$.

To prove Theorem 2, apply the proposition with $k(n) = 2\log s(n)$. ■

Proposition 1 is based on the following

Lemma 1 (Random walk): *Let G be an expander graph with $\lambda_2 \leq d/2$. Let W be a subset of measure $\mu \geq 1/2$ of the expander's nodes. Then fraction $\leq \mu^{k/2}$ of random walks of length k on G are contained in W.*

Lemma 1 generalizes the statement appearing in [Ajtai Komlos Szemeredi 86]. A proof of Lemma 1 is given in the Section 6.

The performance of any inversion algorithm $A(\omega, y)$ has two aspects: the running time $T_A(\omega, y)$ and the probability of success in finding an inverse. We now combine these two measures into a single one:

W.l.o.g, now we consider and call *inverters* algorithms $A(\omega, x)$ which for every x, run in expected (over its internal coin flips ω) polynomial time. (We assume this includes the time required to compute f on the result to check that the inversion is correct.) Any algorithm can be modified to satisfy this requirement. For this purpose, A can use its power of flipping coins to abort its computation so that for each x it runs 2^t steps with probability, say, $2^{-t}/t^2$. This will decrease the probability of success nearly proportionally to A's original running time.

Then probability of inversion by this algorithm accounts for both running time and probability of success: A runs t steps only with probability $n^{-O(1)}/t$ and the following are equivalent:

1. f is $1 - \mu$-OW with security s.

2. Every inverter A has a set H of measure $1 - \mu$, s.t. the fraction of $\{(\omega, x) : f(A(\omega, x)) = x \in H\}$, is $< 1/s$.

We will now use the second form of this definition and for strict security require a fixed degree (e.g., 0) of the polynomial expected running time of the inverters.

Now let H^* be the set of paths (x,p) which intersect with H. By lemma 1, the measure of H^* is $1-\mu^{k/2}$.

Proof of Proposition 1 : For contradiction, let $A(\omega,y,p)$ be an inverter for F. Then an Inverter a for f is as follows: To invert at a given point x choose at random $i \in \{1 \cdots k\}$, and a random walk $p=\sigma_1 \cdots \sigma_k$ of length k. Compute the path (y,p) so that x is in position i in the path (and y is the last point). Compute $z=A(y,p)$. Tracing the path from z compute $f^{-1}(x)$.

We show that H^* a hard set for A, if H is a hard set for a. Indeed, let A succeed in inverting with probability k/s' for instances in H^*. With $1/k$ chance i will point to a hard point in the path. So, a succeeds with probability $1/s'$ for instances in H. ■

Proof of Theorem 3 : By Lemma 3, we can assume w.l.o.g. that f is a.e. 1-1. We need to modify the construction of Section 3 in two respects.

First if $|f(x)| \geq 2|x|$ modify f according to Lemma 3 to make it almost length preserving. Second make $\sigma_i \in \{1,\ldots,d\}^{\log s(n)}$ rather than $\sigma_i \in \{1,\ldots,d\}$ in the definition of F, so that $\log s(n)$ random expander steps are taken between every two applications of f (instead of one random step).

This gives a modification of Proposition 1, where the argument to F has length $n+\Theta(k(n)\cdot\log s(n))$ (instead of $n+\Theta(k(n))$).

Now Theorem 3 follows by iterating the modified Proposition 1 similar to derivation of Theorems 1 and 2. ■

It seems redundant to increase the length at every iteration of f. Also it seems possible to use $O(|\log\mu|)$ rather than $O(\log s)$ random walks in these constructions. We shall explore this in the journal version.

5 Regular functions

Lemma 2 (regular → a.e. $1-1$) *For any regular $\alpha(n)$-one-way function f there exists an almost everywhere 1-1, $\alpha(n)$-one-way function g with comparable security.*

Proof Outline: Let $g(x,\overline{h})=h(x)\circ\overline{h}\circ f(x)$ where $h \in H_{n,m(n)+\varepsilon\log s}$. Clearly, g is a.e. 1-1. Using a simulation argument the security can be shown to be as stated. ■

Lemma 3 *Let f be an a.e. 1-1 α-OW function with security $s(n)$ and $|f(x)|=\Theta(|x|)$. Then there is an almost length preserving a.e. 1-1 α-OW function with security $s(\Theta(n))$.*

Proof outline: Let $g(x,\overline{h})=h(f(x))\circ\overline{h}$, where $h \in H_{|f(x)|,n+\log s}$. ■

Efficient construction of pseudorandom generators: W.l.o.g. let f be a.e. 1-1, strong one-way of security $s(n)$ and $|f(x)|=O(|x|)$. The following pseudo-random generator has security $s^{\Theta(1)}$ and produces $\Theta(\log s)$ extra bits per evaluation: $f'(x,\overline{h}_1,\overline{h}_2) = h_1(x), h_2(f(x)), \overline{h}_1, \overline{h}_2$, where $|x|=n$, $0<\varepsilon<1$, $h_1 \in H_{n,2\varepsilon\log s}$, $h_2 \in H_{|f(x)|,n-\varepsilon\log s}$. This generates a pseudorandom string, slightly longer than the random seed. Construction of a pseudo-random generator which doubles the length of the seed from one which is just length increasing, and the pseudorandom function construction of [Goldreich Goldwasser Micali 86] are then used. These constructions preserve security.

6 Proof of Lemma 1

Modify the the adjacency matrix of G by dividing the entries by d (the degree) and let A be the resulting N-by-N matrix. Let $\lambda_1 \geq \lambda_2 \ldots \geq \lambda_n$ and $e_1, e_2, \ldots, e_n$ be its eigenvalues and corresponding eigenvectors. By Perron-Frobenius Theorem, non-negative A has $\lambda_1=1$. By our assumptions on A, $1/2 \geq \lambda_2 \geq |\lambda_n|$. The space V_0 spanned by $e_2, \ldots, e_N$ is orthogonal to the space V_1 spanned by $e_1=(1,1,\ldots,1)$, and A leaves invariant these subspaces of R^N. Any $X \in R^N$ can be written as $X=X_0+X_1$ where $X_0 \in V_0$ and $X_1 \in V_1$. Let $\|X\|$ be the L^2 (Euclidean) norm and $\|X\|_1$ the L^1 norm (sum of the absolute values). It is well known that $\|AX_0\| \leq \lambda_2 \|X_0\|$.

Let P be a projection matrix such that $PX=Y$ with $Y_i=X_i$, if $i \in W$ and zero otherwise. Then $P^2=P$ and $\|PY\| \leq \|Y\|$ (for every Y).

We show below that

$$\|PAX\| \leq \mu^{1/2} \|X\|.$$

From this it follows that if $Y=(PA)^{2k}X$ then $\|Y\| \leq \mu^k \|X\|$. Note that for $X = e_1/N$, the ith component of Y is the probability that after $2k$ steps, the random walk ends up at i without ever leaving W. Thus the total probability that the random walk never visits an element outside of W in $2k$ steps is $\|Y\|_1 \leq \sqrt{N}\|Y\| \leq \sqrt{N}\mu^k \|e_1/N\| = \mu^k$.

It is left to show that $\|PAX\| \leq \mu^{1/2}\|X\|$. To do this, let $X = X_0 + X_1$, so $\|X\| = (\|X_0\|^2 + \|X_1\|^2)^{1/2}$. The basic idea is that A will reduce the norm of X_0, and P will reduce the norm of X_1. However, PX_1 is no longer perpendicular to X_0 (or AX_0), so we must bound the angle between these vectors.

Using $P^2 = P$ and $AX_1 = X_1$, $\|PAX\| = \|P(AX_0 + PX_1)\| \leq \|AX_0 + PX_1\|$. Observe that AX_0 and X_1 are perpendicular, and let θ denote the angle between X_1 and PX_1. Then

$$\cos\theta = \frac{X_1 \cdot PX_1}{\|X_1\|\|PX_1\|} = \frac{\mu N}{\sqrt{N}\sqrt{\mu N}} = \sqrt{\mu}.$$

Now if we put the vector PX_1 at the tail of the vector AX_0 in order to form the sum $AX_0 + PX_1$, then the angle between these vectors (where they meet) is at most $\pi/2+\theta$. Therefore, using the law of cosines and that $-\cos(\pi/2+\theta) = \sin\theta = \sqrt{1-\cos^2\theta} = \sqrt{1-\mu}$, we have $\|AX_0 + PX_1\|^2 \leq$

$$\|AX_0\|^2 + \|PX_1\|^2 - 2\|AX_0\|\|PX_1\|\cos(\pi/2+\theta)$$

$$= \|AX_0\|^2 + \|PX_1\|^2 + 2\|AX_0\|\|PX_1\|\sqrt{1-\mu} \leq$$

$$\lambda_2^2\|X_0\|^2 + \mu^2\|X_1\|^2 + 2\lambda_2\mu\sqrt{1-\mu}\|X_0\|\|X_1\|.$$

But because the geometric mean is always at most the arithmetic mean for non-negative numbers, $2\lambda_2\|X_0\|\mu\sqrt{1-\mu}\|X_1\| \leq \lambda_2^2\|X_0\|^2 + \mu^2(1-\mu)\|X_1\|^2$. Substituting in above and using $\mu^2 + \mu^2(1-\mu) \leq \mu$, we deduce

$$\|AX_0 + PX_1\|^2 \leq 2\lambda_2^2\|X_0\|^2 + \mu\|X_1\|^2$$

$$\leq \max(2\lambda_2^2, \mu)\|X\|^2.$$

Using $\lambda_2 \leq 1/2$ and $\mu \geq 1/2$ yields $\|PAX\| \leq \mu^{1/2}\|X\|$ and the lemma.

Acknowledgements

Oded Goldreich would like to thank Amir Herzberg for collaboration in the early stages of this work, and Nati Linial, David Peleg, Eli Upfal and Avi Wigderson for discussions concerning random walks on expanders. R.Venkatesan thanks Raf Ostrovsky for valuable discussions.

References

[1] N. Alon. Eigenvalues and Expanders. *Combinatorica*, 1986.

[2] M. Ajtai, J. Komlos, E. Szemeredi. Deterministic Simulation in LOGSPACE, *STOC* 87

[3] Blum, M. Micali, S. How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits. *SIAM J. on Computing* 5:850-864 (1984). Early version in *FOCS 1982*.

[4] Bellare, M., S. Micali.How to Sign Given any Trapdoor Function *STOC* 1988.

[5] S. Ben-David, B. Chor, O. Goldreich, M. Luby, On the Theory of Average Case Complexity. *Proc. 21st STOC*, 1989.

[6] J. Carter and M. Wegman. Universal Classes of Hash Functions. *JCSS* 18:143-154, 1979.

[7] A. Cohen, A. Wigderson, Dispensers, Deterministic Amplification, and Weak Random Sources. *30th FOCS*, 1989.

[8] W. Diffie, M. E. Hellman. New Directions in Cryptography. *IEEE transactions on Info. Theory*, IT-22: 644-654, 1976.

[9] O. Gabber, Z. Galil. Explicit Constructions of Linear Size Superconcentrators. *JCSS* 22: 407-420, 1981

[10] O. Goldreich, Two Remarks concerning the Goldwasser-Micali-Rivest Signature Scheme. *Crypto86*, proceedings, Springer-Verlag, Lecture Notes in Computer Science 263:104-110, 1987.

[11] O. Goldreich, *Foundations of Cryptography - Class notes*, Computer Science Dept., Technion, Haifa, Israel, 1989.

[12] O. Goldreich, H. Krawczyk, M. Luby, On the Existence of Pseudorandom Generators. *FOCS 88*.

[13] O.Goldreich, L.A. Levin, Hard-core Predicate for any One-way Function. *STOC 89*.

[14] O. Goldreich, S. Goldwasser, S. Micali, How to Construct Random Functions. *J. ACM*, 33/4: 792-807, 1986.

[15] S. Goldwasser, S. Micali, and R.L. Rivest, A Digital Signature Scheme Secure Against Adaptive Chosen Message Attacks. *SIAM J on Comput.*17(2):281-308, 1988.

[16] S. Goldwasser, S. Micali, A.C. Yao, Strong Signature Schemes. *15th STOC*, 1983.

[17] J. Hastad, R. Impagliazzo, L. Levin, M. Luby. Pseudo-Random Generators from Any One-way Function. To be published. Preliminary versions in STOC 1989 pp. 12-24, and 1990 pp. 395-404.

[18] R. Impagliazzo, and D. Zuckerman, How to Recycle Random Bits. 30th FOCS, 1989.

[19] R.M. Karp, N. Pippinger, and M. Sipser, A Time-Randomness Tradeoff. *AMS Conf. on Probabilistic Compuational Complexity*, Durham, 1985.

[20] A. Lubotzky, R. Phillips, P. Sarnak, Explicit Expanders and the Ramanujan Conjectures. *STOC 86*.

[21] L.A. Levin, One-Way Function and Pseudorandom Generators, *Combinatorica* 7(4):357-363, 1987. Early version in *17th STOC*, 1985.

[22] L.A. Levin, Homogeneous Measures and Polynomial Time Invariants, FOCS 88.

[23] G.A. Margulis, Explicit Construction of Concentrators. *Probl. of Inf. Transm.,* 9(4), (1973).

[24] R. Merkle, A Certified Digital Signature. unpublished manuscript, 1987.

[25] M. Naor, M. Yung. Universal One-Way Hash Functions and their Cryptographic Applications. *21st STOC*, 1989.

[26] N. Nisan. Pseudo random generators for space-bounded computation. *STOC 90.*

[27] J. Rompel. One-way functions are necessary and sufficient for secure signatures. *STOC* 90

[28] A. C. Yao, Theory and Applications of Trapdoor Functions. *FOCS 1982*.