Increasing the Expansion of Pseudorandom Generators (Extracts from a book on Cryptography)[1]

Oded Goldreich
Department of Computer Science and Applied Mathematics
Weizmann Institute of Science, Rehovot, Israel.

February 24, 1996

Preface: The purpose of the following extract is to provide an accessible source to an unpublished result of Goldreich and Micali (1984) by which the expansion of pseudorandom generators can be increased; see Theorem 3 (below).

3.3 Definitions of Pseudorandom Generators

Pseudorandom ensembles, defined above, can be used instead of uniform ensemble in any efficient application without noticeable degradation in performance (otherwise the efficient application can be transformed into an efficient distinguisher of the supposedly-pseudorandom ensemble from the uniform one). Such a replacement is useful only if we can generate pseudorandom ensembles at a cheaper cost than required to generate a uniform ensemble. The cost of generating an ensemble has several aspects. Standard cost considerations are reflected by the time and space complexities. However, in the context of randomized algorithms, and in particular in the context of generating probability ensembles, a major cost consideration is the quantity and quality of the randomness source used by the algorithm. In particular, in many applications (and especially in cryptography), *it is desirable to generate pseudorandom ensembles using as little randomness as possible.* This leads to the definition of a pseudorandom generator.

3.3.1 Standard Definition of Pseudorandom Generators

Definition 1 (pseudorandom generator - standard definition): *A* pseudorandom generator *is a deterministic polynomial-time algorithm, G, satisfying the following two conditions:*

1. expansion: *there exists a function $l : \mathbb{N} \mapsto \mathbb{N}$ so that $l(n) > n$ for all $n \in \mathbb{N}$, and $|G(s)| = l(|s|)$ for all $s \in \{0,1\}^*$.*
   The function l is called the expansion factor *of G.*

2. pseudorandomness (as above): *the ensemble $\{G(U_n)\}_{n\in\mathbb{N}}$ is pseudorandom.*

Again, we call the input to the generator a *seed*. The expansion condition requires that the algorithm G maps n-bit long seeds into $l(n)$-bit long strings, with $l(n) > n$. The pseudorandomness condition requires that the output distribution, induced by applying algorithm G to a uniformly chosen seed, is polynomial-time indistinguishable from uniform (although it is not statistically close to uniform - see justification in previous subsection).

The above definition says little about the expansion factor $l : \mathbb{N} \mapsto \mathbb{N}$. We merely know that for every n it holds that $l(n) \geq n+1$, that $l(n) \leq \mathrm{poly}(n)$, and that $l(n)$ can be computed in time polynomial in n. Clearly, a pseudorandom generator with expansion factor $l(n) = n+1$ is of little value in practice, since it offers no significant saving in coin tosses. Fortunately, as shown in the subsequent subsection, even pseudorandom generators with such small expansion factor can be used to construct pseudorandom generators with any polynomial expansion factor. Hence, for every two expansion factors, $l_1 : \mathbb{N} \mapsto \mathbb{N}$ and $l_2 : \mathbb{N} \mapsto \mathbb{N}$, that can be computed in poly(n)-time, there exists a pseudorandom generator with expansion factor l_1 if and only if there exists a pseudorandom generator with expansion factor l_2. This statement is proven by using a pseudorandom generator with expansion factor $l_1(n) \stackrel{\text{def}}{=} n+1$ to construct, for every polynomial $p(\cdot)$, a pseudorandom generator with expansion factor $p(n)$. Note that a pseudorandom generator with expansion factor $l_1(n) \stackrel{\text{def}}{=} n+1$ can be derived from any pseudorandom generator.

3.3.2 Increasing the Expansion Factor of Pseudorandom Generators

Given a pseudorandom generator, G_1, with expansion factor $l_1(n) = n+1$, we construct a pseudorandom generator G with polynomial expansion factor, as follows.

Construction 2 *Let G_1 a deterministic polynomial-time algorithm mapping strings of length n into strings of length $n+1$, and let $p(\cdot)$ be a polynomial. Define $G(s) = \sigma_1 \cdots \sigma_{p(|s|)}$, where $s_0 \stackrel{\text{def}}{=} s$, the bit σ_i is the first bit of $G_1(s_{i-1})$, and s_i is the $|s|$-bit long suffix of $G_1(s_{i-1})$, for every $1 \le i \le p(|s|)$. (i.e., $\sigma_i s_i = G_1(s_{i-1})$)*

Hence, on input s, algorithm G applies G_1 for $p(|s|)$ times, each time on a new seed. Applying G_1 to the current seed yields a new seed (for the next iteration) and one extra bit (which is being output immediately). The seed in the first iteration is s itself. The seed in the i^{th} iteration is the $|s|$-long suffix of the string obtained from G_1 in the previous iteration. Algorithm G outputs the concatenation of the "extra bits" obtained in the $p(|s|)$ iterations. Clearly, G is polynomial-time computable and expands inputs of length n into output strings of length $p(n)$.

Theorem 3 *Let G_1, $p(\cdot)$, and G be as in Construction 2 (above). Then, if G_1 is a pseudorandom generator then so is G.*

Intuitively, the pseudorandomness of G follows from that of G_1 by replacing each application of G_1 by a random process which on input s outputs σs, where σ is uniformly chosen in $\{0,1\}$. Loosely speaking, the indistinguishability of a single application of the random process from a single application of G_1 implies that polynomially many applications of the random process are indistinguishable from polynomially many applications of G_1. The actual proof uses the hybrid technique.

Proof: The proof is by a "reducibility argument" . Suppose, to the contradiction, that G is not a pseudorandom generator. It follows that the ensembles $\{G(U_n)\}_{n\in\mathbb{N}}$ and $\{U_{p(n)}\}_{n\in\mathbb{N}}$ are not polynomial-time indistinguishable. We will show that it follows that the ensembles $\{G_1(U_n)\}_{n\in\mathbb{N}}$ and $\{U_{n+1}\}_{n\in\mathbb{N}}$ are not polynomial-time indistinguishable, in contradiction to the hypothesis that G_1 is a pseudorandom generator with expansion factor $l_1(n) = n+1$. The implication is proven, using the hybrid technique.

For every k, $0 \le k \le p(n)$, we define a hybrid $H^k_{p(n)}$ as follows. First we define, for every k, a function $g^k_n : \{0,1\}^n \mapsto \{0,1\}^k$ by letting $g^0_n(x) \stackrel{\text{def}}{=} \lambda$ (the empty string) and $g^{k+1}_n(x) = \sigma g^k_n(y)$, where σ is the first bit of $G_1(x)$ and y is the n-bit long suffix of $G_1(x)$ (i.e., $\sigma y = G_1(x)$). Namely, for every $k \le p(|x|)$, the string $g^k_n(x)$ equals the k-bit long prefix of $G(x)$. Define the random variable $H^k_{p(n)}$ resulting by concatenating a uniformly chosen k-bit long string and the random variable $g^{p(n)-k}(U_n)$. Namely

$$H^k_{p(n)} \stackrel{\text{def}}{=} U^{(1)}_k g^{p(n)-k}(U^{(2)}_n)$$

where $U^{(1)}_k$ and $U^{(2)}_n$ are independent random variables (the first uniformly distributed over $\{0,1\}^k$ and the second uniformly distributed over $\{0,1\}^n$). Intuitively, the hybrid $H^k_{p(n)}$ consists of the k-bit long prefix of $U_{p(n)}$ and the $(p(n)-k)$-bit long suffix of $G(X_n)$, where X_n is obtained from U_n by applying G_1 for k times each time to the n-bit long suffix of the previous result. However, the later way of looking at the hybrids is less convenient for our purposes.

At this point it is clear that $H^0_{p(n)}$ equals $G(U_n)$, whereas $H^{p(n)}_{p(n)}$ equals $U_{p(n)}$. It follows that if an algorithm D can distinguish the extreme hybrids then D can also distinguish two neighbouring hybrids, since the total number of hybrids is polynomial in n and a non-negligible gap between the extreme hybrids translates into a non-negligible gap between some neighbouring hybrids. The punch-line is that, using the structure of neighbouring hybrids, algorithm D can be easily modified to distinguish the ensembles $\{G_1(U_n)\}_{n\in\mathbb{N}}$ and $\{U_{n+1}\}_{n\in\mathbb{N}}$. Details follow.

The core of the argument is the way in which the distinguishability of neighbouring hybrids relates to the distinguishability of $G(U_n)$ from U_{n+1}. As stated, this relation stems from the structure of neighbouring hybrids. Let us, thus, take a closer look at the hybrids $H^k_{p(n)}$ and $H^{k+1}_{p(n)}$, for some $0 \le k \le p(n)-1$. To this end, define a function $f^m : \{0,1\}^{n+1} \mapsto \{0,1\}^m$ by letting $f^0(z) \stackrel{\text{def}}{=} \lambda$ and $f^{m+1}(z) \stackrel{\text{def}}{=} \sigma g^m(y)$, where $z = \sigma y$ with $\sigma \in \{0,1\}$.

Claim 3.1:

1. $H^k_{p(n)} = U^{(1)}_k f^{p(n)-k}(X_{n+1})$, where $X_{n+1} = G_1(U^{(2)}_n)$.
2. $H^{k+1}_{p(n)} = U^{(1)}_k f^{p(n)-k}(Y_{n+1})$, where $Y_{n+1} = U^{(3)}_{n+1}$.

Proof:

1. By definition of the functions g^m and f^m, we have $g^m(x) = f^m(G_1(x))$. Using the definition of the hybrid $H^k_{p(n)}$, it follows that

$$H^k_{p(n)} = U^{(1)}_k g^{p(n)-k}(U^{(2)}_n) = U^{(1)}_k f^{p(n)-k}(G_1(U^{(2)}_n))$$

2. On the other hand, by definition $f^{m+1}(\sigma y) = \sigma g^m(y)$, and using the definition of the hybrid $H^{k+1}_{p(n)}$, we get

$$H^{k+1}_{p(n)} = U^{(1)}_{k+1} g^{p(n)-k-1}(U^{(2)}_n) = U^{(1)}_k f^{p(n)-k}(U^{(3)}_{n+1})$$

□

Hence distinguishing $G_1(U_n)$ from U_{n+1} is reduced to distinguishing the neighbouring hybrids (i.e. $H^k_{p(n)}$ and $H^{k+1}_{p(n)}$), by applying $f^{p(n)-k}$ to the input, padding the outcome (in front of) by a uniformly chosen string of length k, and applying the hybrid-distinguisher to the resulting string. Further details follow.

We assume, to the contrary of the theorem, that G is not a pseudorandom generators. Suppose that D is a probabilistic polynomial-time algorithm so that for some polynomial $q(\cdot)$ and for infinitely many n's it holds that

$$\Delta(n) \stackrel{\text{def}}{=} |\text{Prob}(D(G(U_n)=1) - \text{Prob}(D(U_{p(n)})=1)| > \frac{1}{q(n)}$$

We derive a contradiction by constructing a probabilistic polynomial-time algorithm, D', that distinguishes $G_1(U_n)$ from U_{n+1}.

Algorithm D' uses algorithm D as a subroutine. On input $\alpha \in \{0,1\}^{n+1}$, algorithm D' operates as follows. First, D' selects an integer k uniformly in the set $\{0, 1, ..., p(n)-1\}$, next D' selects β uniformly in $\{0,1\}^k$, and finally D' halts with output $D(\beta f^{p(n)-k}(\alpha))$, where $f^{p(n)-k}$ is as defined above.

Clearly, D' can be implemented in probabilistic polynomial-time (in particular $f^{p(n)-k}$ is computed by applying G_1 polynomially many times). It is left to analyze the performance of D' on each of the distributions $G_1(U_n)$ and U_{n+1}.

Claim 3.2:

$$\text{Prob}(D'(G(U_n))=1) = \frac{1}{p(n)} \sum_{k=0}^{p(n)-1} \text{Prob}(D(H^k_{p(n)})=1)$$

and

$$\text{Prob}(D'(U_{n+1})=1) = \frac{1}{p(n)} \sum_{k=0}^{p(n)-1} \text{Prob}(D(H^{k+1}_{p(n)})=1)$$

Proof: By construction of D' we get, for every $\alpha \in \{0,1\}^{n+1}$,

$$\text{Prob}(D'(\alpha)=1) = \frac{1}{p(n)} \sum_{k=0}^{p(n)-1} \text{Prob}(D(U_k f^{p(n)-k}(\alpha))=1)$$

Using Claim 3.1, our claim follows. □

Let $d^k(n)$ denote the probability that D outputs 1 on input taken from the hybrid $H^k_{p(n)}$ (i.e., $d^k(n) \stackrel{\text{def}}{=} \text{Prob}(D(H^k_{p(n)}=1))$. Recall that $H^0_{p(n)}$ equals $G(U_n)$, whereas $H^{p(n)}_{p(n)}$ equals $U_{p(n)}$. Hence, $d^0(n) = \text{Prob}(D(G(U_n)) = 1)$, $d^{p(n)}(n) = \text{Prob}(D(U_{p(n)}) = 1)$, and $\Delta(n) = |d^0(n) - d^{p(n)}(n)|$. Combining these facts with Claim 3.2, we get,

$$\begin{aligned} |\text{Prob}(D'(G_1(U_n))=1) - \text{Prob}(D'(U_{n+1})=1)| &= \frac{1}{p(n)} \cdot |\sum_{k=0}^{p(n)-1} d^k(n) - d^{k+1}(n)| \\ &= \frac{|d^0(n) - d^{p(n)}(n)|}{p(n)} \\ &= \frac{\Delta(n)}{p(n)} \end{aligned}$$

Recall that by our (contradiction) hypothesis $\Delta(n) > \frac{1}{q(n)}$, for infinitely many n's. Contradiction to the pseudorandomness of G_1 follows. ■