1 Hardcore bits

To motivate the construction of pseudorandom generators from one-way permutations, we start with the observation that since $f\colon \{0,1\}^n \to \{0,1\}^n$ is a permutation, the distribution $f(x)$ when x is random is already uniform. To get a pseudorandom generator, it is therefore sufficient to find just one more bit $h(x)$ so that $(f(x), h(x))$ is pseudorandom.

To understand what this $h(x)$ should look like, let us go back to the definition of pseudorandom generator. The definition requires that for every distinguisher D of size s',

$$\left|\Pr_{x\sim\{0,1\}^n}[D(f(x),h(x))=1] - \Pr_{y\sim\{0,1\}^{n+1}}[D(y)=1]\right| \le \varepsilon.$$

Since $f(x)$ is uniformly distributed, we can rewrite this as

$$\left|\Pr_{x\sim\{0,1\}^n}[D(f(x),h(x))=1] - \Pr_{x\sim\{0,1\}^n,b\sim\{0,1\}}[D(f(x),b)=1]\right| \le \varepsilon.$$

This equation says that given $f(x)$, D is just about as likely to output 1 when its second input is $f(x)$ as when it is a completely random bit. Intutively, this means $f(x)$ should give almost no information about the value $h(x)$. Such a function f is called a *hardcore bit* for f.

Definition 1. Let $f\colon \{0,1\}^n \to \{0,1\}^n$ be a function and $h\colon \{0,1\}^n \to \{0,1\}$ be a predicate. We say h is an (s,ε) *hardcore bit* for f if for every circuit P of size s (called a predictor)

$$\Pr_{x\sim\{0,1\}^n}[P(f(x)) = h(x)] \le 1/2 + \varepsilon.$$

We now formalize the discussion that in order to construct a pseudorandom generator, it is sufficient to have a hardcore bit. The proof is unfortunately somewhat technical, and I don't know of a very intuitive way to present it.

Claim 2. *Suppose f is a permutation and h is an (s,ε) hardcore bit for f. Then $(f(x),h(x))$ is an $(s-O(1),\varepsilon)$ pseudorandom generator.*

Proof. Suppose $(f(x),h(x))$ is not an (s,ε) pseudorandom generator. By our previous discussion, there is a circuit D of size s so that

$$\left|\Pr_{x\sim\{0,1\}^n}[D(f(x),h(x))=1] - \Pr_{x\sim\{0,1\}^n,b\sim\{0,1\}}[D(f(x),b)=1]\right| > \varepsilon.$$

Let us remove the absolute value (by possibly adding a negation gate at the output of D) and introduce the shorthand notation $D_x(a) = D(f(x),a)$. With this notation, we have

$$\Pr_{x,b}[D_x(b)=1] - \Pr_{x,b}[D_x(h(x))=1] > \varepsilon.$$

Since $D_x(b)$ is $D_x(h(x))$ with probability $1/2$ and $D_x(\overline{h(x)})$ with the remaining probability, we have

$$\frac{1}{2}\Pr_x[D_x(\overline{h(x)})=1] - \frac{1}{2}\Pr_x[D_x(h(x))=1] > \varepsilon.$$

Using the relation $\mathrm{E}[(-1)^Z] = 1 - 2\Pr[Z = 1]$, true for every random variable Z that takes values -1 and 1, we can rewrite this as

$$\frac{1}{2}\mathrm{E}_x[(-1)^{D_x(h(x))}] - \frac{1}{2}\mathrm{E}_x[(-1)^{D_x(\overline{h(x)})}] > 2\varepsilon.$$

From where we get that

$$\mathrm{E}_{x,b}[(-1)^{D_x(b)+b+h(x)}] > 2\varepsilon.$$

(If $b = h(x)$, we get the first term, and otherwise we get the other term.) This expression says that $D_x(b) + b$ as a random function in x correlates with $h(x)$, so it suggests using the following predictor P:

P: On input $f(x)$, choose a random $b \sim \{0, 1\}$ and output the bit $D(f(x), b) + b$.

Then $\mathrm{E}[(-1)^{P(f(x))+h(x)}] > 2\varepsilon$, and by using the relation $\mathrm{E}[(-1)^Z] = 2\Pr[Z = 0] - 1$, we get that

$$\Pr[P(f(x)) = h(x)] = \Pr[P(f(x)) + h(x) = 0] = \frac{1}{2} + \frac{1}{2}\mathrm{E}[(-1)^{P(f(x))+h(x)}] > \frac{1}{2} + \varepsilon$$

contradicting the assumption that h is an (s, ε) hardcore bit for f. □

We are now left with the task of constructing a hardcore bit for f. How should we go about this? Without any information about f – apart from the fact that it is a one-way permutation – it makes sense to try and choose some predicate that depends on all the bits of f, for it could be that f is designed in such a way that part of its input gets simply copied in the output, so we do not want to be too dependent on any specific bit. This suggests that perhaps taking the XOR of all the bits of x would be a hardcore bit for f. However, this is incorrect: It turns out that if one-way permutations exist, then there are also one-way permutations where, say, the first output equals the XOR of all the input bits.

2 The Goldreich-Levin theorem

The amazing insight of Goldreich and Levin is that this problem can be avoided by randomizing the hardcore bit. Specifically, they suggest the following construction. Given a one-way permutation $f\colon \{0,1\}^n \to \{0,1\}^n$, it is easy to see that the function $(x, r) \to (f(x), r)$, where $r \in \{0,1\}^n$ is also a one-way permutation. Then

Theorem 3. *(Goldreich-Levin theorem) If f is a (s, ε) one-way permutation, then the predicate $h_S(x, r) = \langle x, r\rangle$ is an $(s \cdot (\varepsilon/n)^{O(1)}, O(\varepsilon))$ hardcore bit for the permutation $(f(x), r)$, where*

$$\langle x, r\rangle = x_1 r_1 + \cdots + x_n r_n \mod 2.$$

The proof of this theorem goes by contradiction. Let $s' = s \cdot (\varepsilon/n)^{O(1)}$ and $\varepsilon' = O(\varepsilon)$ and let's assume that $\langle x, r\rangle$ is not (s', ε') hardcore for $(f(x), r)$. Then there is a predictor P for which

$$\Pr_{x,r}[P(f(x), r) = \langle x, r\rangle] > \frac{1}{2} + \epsilon.$$

It follows that

$$\Pr_x\left[\Pr_r\left[P(f(x),r)=\langle x,r\rangle\right]>\frac{1}{2}+\frac{\epsilon}{2}\right]>\frac{\epsilon}{2}$$

Let S be the set of all x such that

$$\Pr_r\left[P(f(x),r)=\langle x,r\rangle\right]>\frac{1}{2}+\frac{\epsilon}{2} \tag{1}$$

This suggests the following algorithm for inverting $f(x)$ when $x\in S$: On input $z=f(x)$, try to find all x' such that $\Pr_r\left[P(z,r)=\langle x',r\rangle\right]>1/2+\epsilon/2$. Since x satisfies (1), one of these x' must equal x. To find out which one, apply f to all of them and see which one maps to z. Since f is a permutation, if $f(x')=z$ it must be that $x'=x$.

Can we implement this algorithm efficiently? At first, the idea seems unreasonable: It looks like there might be exponentially many x' such that $\Pr_r\left[P(z,r)=\langle x',r\rangle\right]>1/2+\epsilon/2$, so even listing all of them, much less computing them, may take too much time. However, this is not the case, and the search of x' can be carried out in polynomial-time:

Lemma 4. *There is a randomized algorithm $A^?$ which on input ε and given oracle access to $g:\{0,1\}^n\to\{0,1\}$ runs in time $(s/\varepsilon)^{O(1)}$ and with probability $2/3$ outputs a list that contains all x such that*

$$\Pr_r\left[g(r)=\langle x,r\rangle\right]\geq\frac{1}{2}+\varepsilon.$$

The proof of the Goldreich-Levin theorem makes use of the Chebyshev and Chernoff large deviation inequalities, which are stated in the appendix.

3 Proof of the Goldreich-Levin theorem

We will start by proving a much weaker statement than what is required, and strenghten it in stages to derive the proof of the theorem. Our goal is to design an algorithm A that outputs all x such that

$$\Pr_r\left[g(r)=\langle x,r\rangle\right]\geq p.$$

where $p=1/2+\epsilon$. Let us however start with the case $p=1$.

Case $p=1$. In this case, A can evaluate $g(r)=\langle x,r\rangle$ for every r and wants to "recover" x. It is not hard to see that g uniquely determines x, and the ith bit of x is given by $x_i=\langle x,e_i\rangle=g(e_i)$, where e_i is the string that has 1 in the ith coordinate and 0 everywhere else. So in this way we can recover x bit by bit.

Case $p=1-\frac{1}{6n}$. Now we need to work a bit harder, since it might be the case that $g(e_i)\neq\langle x,e_i\rangle$, so querying g at e_i might be misleading. But we can deduce the value $\langle x,e_i\rangle$ by querying g at two random but correlated points: We know that for every r, $x_i=\langle x,e_i\rangle=\langle x,r\rangle+\langle x,r+e_i\rangle$. Moreover, for a random r, the strings r and $r+e_i$ are both uniformly random in $\{0,1\}^n$. So, if we choose a random r and compute $g(r)+g(e_i+r)$, we have that

$$\Pr_r[g(r)+g(e_i+r)\neq x_i]\leq\Pr_r[g(r)\neq\langle x,r\rangle]+\Pr_r[g(e_i+r)\neq\langle x,e_i+r\rangle]<\frac{1}{6n}+\frac{1}{6n}<\frac{1}{3n}$$

By taking a union bound, we have that $\Pr_r[\exists i : g(e_i + r) + g(r) \neq x_i] < \frac{1}{3}$. So with probability 2/3 this randomized algorithm recovers all the bits of x.

Case $p = \frac{3}{4} + \epsilon$. In the above algorithm, we now have that

$$\Pr_r[g(r) + g(e_i + r) \neq x_i] \leq \Pr_r[g(r) \neq \langle x, r\rangle] + \Pr_r[g(e_i + r) \neq \langle x, e_i + r\rangle] < 2(1/4 - \epsilon) < 1/2 - 2\epsilon.$$

We cannot take a union bound of n such events anymore. However, by repeating this procedure several times, we can increase its success probability from $1/2 - 2\epsilon$ to $1/3n$, and then take the union bound.

In particular, consider the following algorithm: For each $1 \leq i \leq n$, compute the value $g(r)+g(e_i+r)$ for $t = O(\log n/\epsilon^2)$ independent values of r and let x_i equal the majority of the answers. Each trial gives the correct value for x_i with probability $1/2 - 2\epsilon$ and the trials are independent, so by the Chernoff bound the probability that a majority of the trials fails is at most $\exp((2\epsilon)^2 t/2) < 1/3n$.

Case $p = 1/2 + \epsilon$. We now give the proof of the theorem. It turns out that an interesting phenomenon happens when we try to take $p \leq 3/4$. By the analysis of the previous case, it follows that when $p > 3/4$, there is a unique x such that $\Pr[g(r) = \langle x, r\rangle] \geq p$. However, when $p \leq 3/4$, there may be two or more such xs. So we must introduce a way into the algorithm to disambiguate between the different solutions.

There is also an evident (and related) problem with the above analysis: If we attempt to use the same algorithm, we would get that $\Pr[g(r) + g(e_i + r) \neq x_i] < 1 - 2\varepsilon$, so it appears that we do not obtain any information about the value x_i.

However, suppose that someone could tell us the values $h_r = \langle x, r\rangle$ needed by the algorithm, so we wouldn't have to query g to get them and make potential mistakes. Then we would have

$$\Pr_r[h_r + g(e_i + r) \neq x_i] \leq \Pr_r[g(e_i + r) \neq \langle x, e_i + r\rangle] < 1/2 - \epsilon. \tag{2}$$

so the previous algorithm would work – provided that we knew the values $h_r = \langle x, r\rangle$.

How can we get hold of the values h_r? One possibility is to simply guess them, and think of each possible guess as giving a candidate value for x. So to obtain a list of all x, one can simply go through all the choices for h_r. How many such choices are there? For each i, the algorithm uses $O(\log n/\epsilon^2)$ choices of r, and there are n possible values of i, so we need to guess $O(n \log n/\epsilon^2)$ different values h_r. It looks like going through all the choices would take time exponential in n!

One place in the algorithm where we can save immediately is this: Instead of using independent choices of r for the different coordinates i, we can in fact make the same choices. In the end, our analysis works by taking a union bound over i, so it does not matter if the randomness used for different coordinates is the same. This will reduce the number of random strings r needed by the algorithm to $O(\log n/\epsilon^2)$, so the number of possible choices for h_r becomes $2^{O(\log n/\epsilon^2)} = n^{O(1/\epsilon^2)}$. This is too large, as we usually think of ϵ as being inverse polynomial in n.

To further improve the algorithm, we introduce additional correlations among the rs. To amplify the success probability of (2) from $1/2 + \epsilon$ to $1 - 1/3n$, it is not really necessary that the rs are independent. It turns out that we can choose them in a dependent way so that we only need to guess the value h_r for a very small number of r, and this will automatically yield guesses for the other rs.

We choose the rs from the following distribution: First, choose a "basis" $r_1, \ldots, r_m \sim \{0,1\}^n$ independently at random, where $m = \log_2 O(n/\epsilon^2)$. Then, for every subset $S \subseteq \{1, \ldots, m\}$, set $r_S = \sum_{j \in S} r_j$. Notice that guesses h_j for the values $\langle x, r_j \rangle$ automatically yield guesses h_S for the values $\langle x, r_S \rangle$ via the formula $\langle x, r_S \rangle = \sum_{j \in S} \langle x, r_j \rangle$.

We can now give the algorithm A from the theorem:

A^g: Choose $r_1, \ldots, r_m$ independently at random from $\{0,1\}^n$.
  For every choice of values $h_1, \ldots, h_m \in \{0,1\}$:
    For every $1 \leq i \leq n$:
      For every $S \subseteq \{1, \ldots, m\}, S \neq \varnothing$:
        Set $r_S = \sum_{j \in S} r_i$ and $h_S = \sum_{j \in S} h_j$.
        Compute $a_{i,S} = h_S + g(e_i + r_S)$.
      Set $x_i = \text{majority}_S(a_{i,S})$.
    Output $x = x_1 \ldots x_n$.

We choose $m = \log_2(6n/\epsilon^2)$, so the number of possible choices for $h_1, \ldots, h_m$ is $6n/\epsilon^2$ and the running time of the algorithm is polynomial in n and $1/\epsilon$.

Claim 5. *For every x such that $\Pr_r[g(r) = \langle x, r \rangle] \geq 1/2 + \epsilon$, with probability $2/3$ over the choice of $r_1, \ldots, r_m$, A^g outputs x.*

This claim almost proves the Goldreich-Levin theorem. The only difference is that it only guarantees each x satisfying the condition will appear in the list with probability $2/3$, while the theorem says that the list contains all such x with probability $2/3$. To take care of this, we run the algorithm n times and take the union of all the lists output by it. Then each such x will appear in the list with probability $1 - 3^{-n}$, so by a union bound the list will contain all such x with probability $2/3$.

Proof. We will show that A^g outputs x when $h_i = \langle x, r_j \rangle$ for all $1 \leq j \leq m$, which also implies $h_S = \langle x, r_S \rangle$ for every $S \subseteq \{1, \ldots, m\}$. Let us fix this choice for h_i. As before, is enough to show that for all i,

$$\Pr_r\big[\text{majority}_S(h_S + g(e_i + r_S)) = x_i\big] > 1 - \frac{1}{3n}.$$

Let

$$Y_S = \begin{cases} 1, & \text{if } h_S + g(e_i + r_S) = x_i, \\ 0, & \text{otherwise.} \end{cases}$$

Then for every S,

$$\Pr[Y_S = 1] = \Pr[h_S + g(e_i + r_S) = x_i] = \Pr[g(e_i + r_S) = \langle x, e_i + r_S \rangle] \geq \frac{1}{2} + \epsilon$$

The main observation here is that the random variables Y_S are pairwise independent, since the variables r_S are pairwise independent and Y_S is determined by r_S. We can therefore use Chebyshev's inequality to obtain a deviation bound on $Y = \sum_{S \subseteq \{0,1\}^m} Y_S$. Let us assume for simplicity that $\Pr[Y_S = 1] = 1/2 + \epsilon$. Then

$$\mathrm{E}[Y] = \sum_{S \subseteq \{0,1\}^m} \mathrm{E}[Y_S] = (1/2 + \epsilon) \cdot 2^m \qquad \text{and} \qquad \mathrm{Var}[Y] = \sum_{S \subseteq \{0,1\}^m} \mathrm{Var}[Y_S] \leq 2^m.$$

By Chebyshev's inequality, we have

$$\begin{aligned}\Pr[x_i \neq \text{majority}_S(a_{i,S})] &\leq \Pr\big[Y < \mathrm{E}[Y] - \epsilon \cdot 2^m\big] \\ &\leq \Pr\Big[Y < \mathrm{E}[Y] - \epsilon \cdot 2^{m/2} \cdot \sqrt{\mathrm{Var}[Y]}\Big] \\ &\leq \frac{1}{(\epsilon \cdot 2^{m/2})^2} < \frac{1}{3n}.\end{aligned}$$

□

A Large deviation inequalities

Recall the variance of a random variable X is given by the formula

$$\mathrm{Var}[X] = \mathrm{E}[X^2] - \mathrm{E}[X]^2 = \mathrm{E}[(X - \mathrm{E}[X])^2].$$

Let $X_1, \ldots, X_m$. Large deviation inequalities give tail bounds on the distribution of $X = X_1 + \cdots + X_m$. Recall that $X_1, \ldots, X_m$ are pairwise independent if for every pair $i \neq j$, X_i and X_j are statistically independent. If $X_1, \ldots, X_m$ are pairwise independent, then

$$\mathrm{Var}[X] = \mathrm{Var}[X_1] + \cdots + \mathrm{Var}[X_m].$$

However, this is not true in general. Chebyshev's inequality gives deviation bounds on the random variable X. It always holds, but it is most useful when X is a sum of pairwise independent random variables (or almost so):

Theorem 6 (Chebyshev inequality)**.** *For every random variable X*

$$\Pr\big[|X - \mathrm{E}[X]| \geq t\sqrt{\mathrm{Var}[X]}\big] \leq \frac{1}{t^2}.$$

The Chernoff inequality gives a sharper bound, but under the restriction that the X_i are independent and take $0, 1$ values:

Theorem 7 (Chernoff bound)**.** *Let $X_1, \ldots, X_n$ be independent identically distributed random variables that take value 0 or 1, and $X = X_1 + \cdots + X_n$. Then*

$$\Pr\big[X \geq \mathrm{E}[X] + \varepsilon n\big] \leq e^{-2\varepsilon^2 n}.$$

There are even sharper versions of this inequality, and also a corresponding bound on the left tail of X, but we won't need them here.