0368.4162: Introduction to Cryptography	Ran Canetti
Lecture 4	
24 November 2008	Fall 2008

Scribes: Margarita Vald

Topics for today - Pseudorandom generators:

- Definition
- Range extension
- Construction from OWF
  - Hard core bits

1 Defining PRGs

Definition 1. Two ensembles, $D^1 \stackrel{\text{def}}{=} \left\{D^1_n\right\}_{n\in\mathbb{N}}$ and $D^2 \stackrel{\text{def}}{=} \left\{D^2_n\right\}_{n\in\mathbb{N}}$ are **polynomially indistinguishable** ($D^1 \approx_c D^2$) if for every non-uniform polynomial-time algorithm A and all sufficiently large n's,

$$\left|Prob_{x\leftarrow D^1_n}\left[A\left(1^n,x\right)=1\right]-Prob_{x\leftarrow D^2_n}\left[A\left(1^n,x\right)=1\right]\right|<\nu(n)$$

where $\nu(n)$ is some negligible function.

Definition 2 (Pseudorandom Ensembles)**.** The ensemble $D=\{D_n\}_{n\in\mathbb{N}}$ is called **pseudorandom** if $D\approx_c U$ for the uniform ensemble $U\stackrel{\text{def}}{=}\{U_n\}_{n\in\mathbb{N}}$.

Theorem 3. *There exists a distribution ensemble $D=\{D_n\}_{n\in\mathbb{N}}$ such that D is not statistically close to the uniform ensemble $U=\{U_n\}_{n\in\mathbb{N}}$ and yet, $D\approx_c U$.*

The theorem is proven without computational assumptions, based on the following diagonalization argument: Since there are only exponentially many polynomial size circuits on a bit, it is possible to define a distribution that "fools" all such circuits. For a proof see [4].

Theorem 3 highlights the fact that pseudorandomness in itself is not good enough for encryption purposes. We are interested in pseudorandom ensembles that can be efficiently sampled.

Definition 4. A **pseudorandom generator** (PRG) is a deterministic function $G:\{0,1\}^*\rightarrow\{0,1\}^*$ satisfying the following conditions:

- Efficiency: *G is computable in polynomial-time.*
- Expansion: *There exist a function $l:\mathbb{N}\rightarrow\mathbb{N}$ such that $l(n)>n$ for all $n\in\mathbb{N}$, and $|G(x)|=l(|x|)$ for all $x\in\{0,1\}^*$. ($l(n)$ is called the expansion factor of G.)*
- Pseudorandomness: *The ensemble $\{G(U_n)\}_{n\in\mathbb{N}}$ is pseudorandom.*

A PRG can be viewed as the formalization of the informal notion of a "Stream cipher". That is, a PRG can be thought of as the "security specification" for stream ciphers.

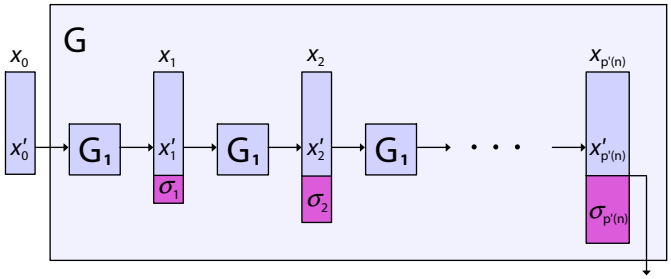

Figure 1: The construction operating on seed $x_0 \in \{0,1\}^n$.

2 Increasing the Expansion Factor

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

Construction *Let G_1 be a deterministic polynomial-time function, mapping strings of length n into strings of length $n+1$, and let $p(n)$ be polynomial. We now formally describe the construction of algorithm G on input $x \in \{0,1\}^n$:*

- *Let $p'(n) = p(n) - n$. Note that this is the amount by which G is supposed to increase the length of its input.*
- *set $x_0 = x$ and $n = |x|$. For i=1 to $p'(n)$, do:*
  - *Let x'_{i-1} denote the first n bits of x_{i-1}, and let σ_{i-1} denote the remaining $i-1$ bits. (When $i = 1$, σ_0 is the empty string.)*
  - *Set $x_i := \left(G_1\left(x'_{i-1}\right), \sigma_{i-1}\right)$*
- *Output $x_{p'(n)}$*

The construction is depicted in Figure 1.

Theorem 5. *Let G_1, $p(\cdot)$, and G be as in the construction such that $p(n) > n$. 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 that on input a uniformly distributed n-bit -long string will output a uniformly distributed $(n+1)$-bit-long string. consequently, the indistinguishably 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.

How to formalize this idea? Can we use induction to prove security? This is problematic! Every application of G_1 adds extra information to the adversary. Since G_1 is a PRG, the extra information is small but still at each application of G_1 the error grows. We thus need a way to bound the growth of error and simple induction doesn't give us a way to do that.

The actual proof uses the hybrid technique.

The hybrid technique. The *hybrid technique* is used in many proofs of security and is a basic tool for proving indistinguishably when a basic primitive is applied multiple times. The technique works by defining a series of hybrid distributions that bridge between two "extreme distribution", these being the distributions that we wish to prove indistinguishable. To apply the proof technique, three conditions must hold: First, the extreme hybrids should match the original cases of interest. Second,

it must be possible to translate the capability of distinguishing neighboring hybrids into the capability of breaking some underlying assumption. Finally, the number of hybrids should be polynomial, so the "distinguishing success" is only reduced by a polynomial factor.

We will prove theorem 5 for $p'(n) = 2$ (Homework: prove for any polynomial number of iterations of G_1).

Proof. Let A' be an adversary that distinguishes between $U_{n+2}, G(U_n)$ by margin ϵ. We construct A that distinguishes between $U_{n+1}, G_1(U_n)$ by a margin of $\epsilon/2$.

$\left\{H_n^2\right\} \stackrel{\text{def}}{=} U_{n+2}$	$\left\{H_n^1\right\} \stackrel{\text{def}}{=}$ Hybrid	$\left\{H_n^0\right\} \stackrel{\text{def}}{=} G(U_n)$
$r_1 \cdots r_{n+2}$	$G_1(r_1 \cdots r_n) r_{n+1}$	$G_1(y_1 \cdots y_n) y_{n+1}$, where $y_i = G_1(r_1 \cdots r_n)_i$

Construction of A: (on input $z = z_1 \cdots z_{n+1}$)

- A tosses a coin b
  - If $b = 1$ (we assume that A' can distinguish between $\left\{H_n^1\right\}$ and $\left\{H_n^2\right\}$) we apply A' on $z_1 \cdots z_{n+1}\sigma_1$ where $\sigma_1 \stackrel{R}{\leftarrow} \{0,1\}$ and output A' answer.
  - If $b = 0$ (we assume that A' can distinguish between $\left\{H_n^1\right\}$ and $\left\{H_n^0\right\}$) we apply A' on $G_1(z_1 \cdots z_n) z_{n+1}$ and output A' answer.

$$\begin{aligned} Prob_{z \leftarrow U_{n+1}}[A(z) = 1] &= 1/2\left(Prob_{z \leftarrow U_{n+1}}[A(z) = 1 | b = 1]\right) + \\ &\quad \left(Prob_{z \leftarrow U_{n+1}}[A(z) = 1 | b = 0]\right) \\ &= 1/2\left(Prob_{z \leftarrow \{H_n^2\}}[A'(z) = 1] + Prob_{z \leftarrow \{H_n^1\}}[A'(z) = 1]\right) \end{aligned} \tag{1}$$

$$\begin{aligned} Prob_{z \leftarrow G_1(U_n)}[A(z) = 1] &= 1/2\left(Prob_{z \leftarrow G_1(U_n)}[A(z) = 1 | b = 1]\right) + \\ &\quad \left(Prob_{z \leftarrow G_1(U_n)}[A(z) = 1 | b = 0]\right) \\ &= 1/2\left(Prob_{z \leftarrow \{H_n^1\}}[A'(z) = 1] + Prob_{z \leftarrow \{H_n^0\}}[A'(z) = 1]\right) \end{aligned} \tag{2}$$

$$\begin{aligned} |(1) - (2)| &= \left|1/2\left(Prob_{z \leftarrow U_{n+1}}[A'(z) = 1] - Prob_{z \leftarrow G(U_n)}[A'(z) = 1]\right)\right| \\ &= \epsilon/2 \end{aligned}$$

□

3 Constructing PRGs

There are several approaches to constructing PRGs:

1. We can give a specific construction and simply assume that it is a PRG (recall the shrinking generator described in the last class). We can also try to analyze specific attacks. But we are looking for something better in terms of security guarantees.
2. Alternatively, we can give a construction whose security can proved based on the hardness of a well-known computational problem (e.g., one of the problems mentioned in the last class).

Example 6. A PRG based on a strong assumption on the hardness of discrete logarithm.

So far we have assumed that, given a prime p and two random elements g, h in Z_p^* it is hard to find an index i s.t. $g^i \equiv h \bmod p$. For the following assumption we need an extra structure: a group G of prime order (in a group of prime order every element is a generator).

By choosing $p = 2 \cdot q + 1$, where q is also a prime, and letting G be the sub-group of order q in Z_p^* we obtain such a group (The primes q above are called "Sophie-Germain"). Let $\{G_n\}_{n\in\mathbb{N}}$ be a sequence of groups, where G_n is of prime order q_n, $q_n \sim 2^n$.

The DDH assumption for $\{\mathbf{G_n}\}_{\mathbf{n}\in\mathbb{N}}$: Let $g, h \overset{R}{\leftarrow} G_n$ and $r, s \overset{R}{\leftarrow} [1, ..., q_n]$ then

$$\{(g, g^r, h, h^r)_n\}_{n\in\mathbb{N}} \approx_c \{(g, g^r, h, h^s)_n\}_{n\in\mathbb{N}}$$

Note that h^r in the l.h.s is deterministically determined by the random elements g, g^r, h

The assumption says that not only it is hard to find the DL of elements but it is also hard to recognize when two elements have the same DL wrt two given random generators.

This stronger assumption is called "Decisional Diffie-Hellman" assumption (for reasons that will became clear later in the course).

Under this assumption, we construct a PRG as follows:

$$GEN_G(g, h, r) = g, g^r, h, h^r$$

This generate an expansion by $log\,|G_n|$ bits.

Claim 7. If the DDH assumption holds for G then GEN_G is a PRG.

3. A third alternative is to construct PRGs based on a general hardness assumption.

Theorem 8. *There exist PRGs iff there exist OWFs [3].*

You can find a sketch of the proof in [4]. However, the construction which transforms an arbitrary OWF into PRG and the proof that this construction indeed yields PRGs, is beyond the scope of this course . In class we'll show a simple derivation:

Theorem 9. *If there exist one way permutations , then there exist PRGs.*

4 Constructing PRGs From One Way Permutations

Let $f : \{0,1\}^n \to \{0,1\}^n$ be a length-preserving one-way permutation. How will we construct PRGs? We have seen that it suffice to construct PRGs that expand by one bit. Our first attempt may be:

$$G(x) = f(x), x_1$$

Taking the first bit or any other bit of x is a bad idea. All we have guaranteed is the hardness to invert f, but not the "pseudorandomness" of each bit separately. Thus we're looking for a way to "extract" pseudorandom bits from OWFs.

Definition 10 (Hard-Core Predicate)**.** A polynomial time computable predicate $B : \{0,1\}^* \to \{0,1\}$ is called a **hard-core** for a function f if for every non-uniform polynomial-time adversary A, exists a negligible function ν, such that:

$$Prob_{x\leftarrow U_n}[A(f(x)) = B(x)] < \frac{1}{2} + \nu(n)$$

An equivalent definition is:

Definition 11. $B : \{0,1\}^* \to \{0,1\}$ is a **hard-core** for f, if for $x \leftarrow U_n$, $b \leftarrow U_1$:

$$(f(x), B(x)) \approx_c (f(x), b)$$

This notion provides a "bridge" between hardness and pseudorandomness: According to Definition 11 the two distributions have a significant statistical distance, yet they are indistinguishable.

Theorem 12. *Let f be a OWP and let B be a hard-core predicate for f. Then the algorithm* Gen *defined by:*

$$\mathrm{Gen}(x) \stackrel{\text{def}}{=} f(x) \cdot B(x)$$

is PRG.

Proof. Following immediately from Definition 11. □

5 OWFs with Hard-Core predicates

Can we construct OWFs with HC predicates?

Theorem 13 (Blum-Micali 82)**.** *The following predicate is hard-core for the DL function:*

$$B(x) = 1 \iff x > {}^{p}/_{2}$$

This gives a construction of a PRG based on the DL assumption. However, we want to construct more general PRGs. We don't know how to show that any OWF has an HC bit, but we will show something almost as good:

Theorem 14 (Goldreich-Levin 87)**.** *Let f be OWF. Let f' be defined by: $f'(x,r) \stackrel{\text{def}}{=} (f(x), r)$, where $|x| = |r|$ .Then the predicate*

$$B(x) = \langle x, r \rangle = \sum_{i=1}^{n} x_i r_i (\mathrm{mod}\, 2)$$

is hard-core for f'.

By theorem 14, given any OWF f we can modify f to OWF f' with hard-core predicate. Furthermore, if f is a OWP, then so is f'.

6 Markov, Chernoff, Hoefding, Chebyshev

Markov, Chernoff, Hoefding, Chebyshev are fundamental inequalities from probability theory. These inequalities bound the "tail probabilities", namely the probability that random variable, or a sum of random variables, turns out to be far from the expectation. We take the opportunity to review them here, since we will use them to prove theorem 14.

6.1 Markov inequality

Theorem 15. *Let X be a non-negative random variable, and $v > 0$, then:*

$$Prob[X \geq v] \leq \frac{E[x]}{v}$$

Proof. We have

$$E[X] = \sum_{x \geq 0} Prob[X = x] \cdot x \geq \sum_{0 \leq x < v} Prob[X = x] \cdot 0 + \sum_{x \geq v} Prob[X = x] \cdot v = Prob[X \geq v] \cdot v$$

□

6.2 Chernoff, Hoefding inequality

Theorem 16 (Chernoff Bound[1])**.** *Let $X_1, \ldots, X_n$ be n independent random variables with expectation μ such that $\forall j, X_j \in \{0,1\}$. Then for any $\epsilon > 0$,*

$$Prob\left(\left|\frac{\sum_{i=1}^n X_i}{n} - \mu\right| > \epsilon\right) \leq 2e^{\frac{-\epsilon^2 n}{2\mu(1-\mu)}}$$

Remark 17. Chernoff proved this theorem for $0-1$ random variables, while Hoefding deals with general domains.

6.3 Chebyshev inequality

Sometimes we can't guarantee full independence of the random variables, but only limited independence.

Definition 18. A sequence of random variables $X_1, \ldots, X_n$ is **pairwise independent** if for any i, j, a, b,

$$Prob\left[X_i = a \wedge X_j = b\right] = Prob\left[X_i = a\right] Prob\left[X_i = b\right]$$

Theorem 19 (Chebyshev bound)**.** *Let $X_1, \ldots, X_n$ be pairwise independent random variables with expectation μ and variance σ^2. Then, for any $\epsilon > 0$:*

$$Prob\left(\left|\frac{\sum_{i=1}^n X_i}{n} - \mu\right| > \epsilon\right) \leq \frac{\sigma^2}{\epsilon^2 n}$$

Proof.

$$\begin{aligned}
Prob\left(\left|\frac{\sum_{i=1}^n X_i}{n} - \mu\right| > \epsilon\right) &\leq Prob\left(\left(\frac{\sum_{i=1}^n X_i}{n} - \mu\right)^2 > \epsilon^2\right) \\
&\leq \frac{E\left[\left(\frac{\sum_{i=1}^n X_i}{n} - \mu\right)^2\right]}{\epsilon^2} \quad (by\ Markov) \\
&= \frac{Var\left[\frac{\sum_{i=1}^n X_i}{n}\right]}{\epsilon^2} = \frac{\frac{1}{n^2}\sum_{i=1}^n Var\left[X_i\right]}{\epsilon^2} \\
&= \frac{\frac{1}{n^2}\sum_{i=1}^n \sigma^2}{\epsilon^2} = \frac{\sigma^2}{\epsilon^2 n}
\end{aligned}$$

□

7 Proof of the Goldreich-Levin theorem

Proof. By reduction: Let A' be an algorithm that on input $\left(1^{|x|}, f(x), r\right)$ predicts $\sigma = \langle x, r\rangle$ with probability e'. We will construct an algorithm A that, using A', inverts f with probability e. That is, given y, A will output x s.t $f(x) = y$ with probability e. A will use y only to run A' on, and will concentrate on trying to reconstruct x from the "noisy information" provided by A'.

[1] A proof is available in this link: `http://people.csail.mit.edu/ronitt/COURSE/S07/lec25.pdf`

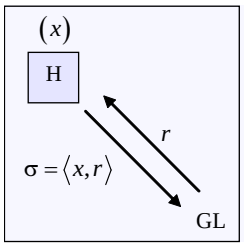

Figure 2: The probabilistic (oracle) algorithm GL

Lemma 20. *(See Figure 2 on page 7) There exists a probabilistic (oracle) algorithm GL such that for any* $h : \{0,1\}^n \to \{0,1\}$*,* $x \in \{0,1\}^n$*, and* $\epsilon > 0$*, if* $h(r)$ *agrees with* $\sigma = \langle x, r\rangle$ *in* $\geq \frac{1}{2} + \epsilon$ *fraction of* r*'s , then:*

$$Prob\left[GL^h(n,\epsilon) = x\right] \geq \Omega\left(\epsilon^2/n\right)$$

where $GL^h(n,\epsilon)$ *runs in time poly*$(n, 1/\epsilon)$ *. (Here* GL^h *means the algorithm* GL *has oracle access to* h*.)*

Remark 21. Note that x is fixed, and h relates to the same x always.

Proof of theorem 14 from the lemma:

Intuitively, the main difference between the setting of the lemma and that of the theorem is that in the former setting the value x is well-defined, and unique, whereas in the latter setting A' may refer to a different preimage of $f(x)$ in each answer. We will show that this this difference can be overcome.

Assume, in contradiction to the conclusion that $\langle x, r\rangle$ isn't HC predicate, that there exist an A' such that:

$$Prob_{x,r}\left[A'\left(1^{|x|}, f'(x,r)\right) = \langle x, r\rangle\right] \geq 1/2 + \epsilon$$

Where $x, r \overset{R}{\leftarrow} \{0,1\}^n$, and $f'(x,r) = (f(x), r)$, and ϵ is non-negligible. Now we define an algorithm $A(y)$ as follows:

- Output $GL^{A'\left(1^{|x|}, y, \cdot r\right)}(n, \epsilon/2)$

Claim 22. For at least a fraction $\epsilon/2$ of the x's we have:

$$p(x) = Prob_r\left[A'\left(1^{|x|}, f(x), r\right) = \langle x, r\rangle\right] \geq 1/2 + \epsilon/2 \tag{3}$$

Proof. Consider the set:

$$S_n \overset{\text{def}}{=} \{x | p(x) \geq 1/2 + \epsilon/2\}$$

Assume by the way of contradiction that: $|S_n| = \delta < \epsilon/2$. Then:

$$\begin{aligned} Prob_{x,r}\left[A'\left(1^{|x|}, f'(x,r)\right) = \langle x, r\rangle\right] &= Prob\left[x \in S_n\right] \cdot Prob_r\left[A'\left(1^{|x|}, f(x), r\right) = \langle x, r\rangle \,|\, x \in S_n\right] + \\ &\quad Prob\left[x \notin S_n\right] \cdot Prob_r\left[A'\left(1^{|x|}, f(x), r\right) = \langle x, r\rangle \,|\, x \notin S_n\right] \\ &\leq \delta \cdot 1 + (1-\delta) \cdot (1/2 + \epsilon/2) \\ &< \delta \cdot 1 + 1 \cdot (1/2 + \epsilon/2) < 1/2 + \epsilon \end{aligned}$$

□

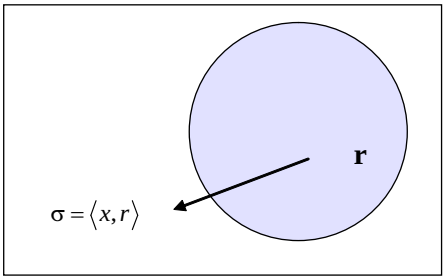

Figure 3: $h(r)$ agrees with $\sigma = \langle x, r \rangle$ in the entire domain r_n

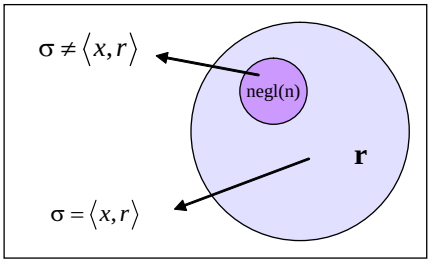

Figure 4: $h(r)$ doesn't agree with $\sigma = \langle x, r \rangle$ with negligible probability over r_n

This means that for $\geq \epsilon/2$ fraction of x's, the function $h(r) = A'\left(1^{|x|}, f(x), r\right)$ agrees with $\langle x, r \rangle$ in $\geq 1/2 + \epsilon/2$ fractions of inputs. We know that for those x's,

$$Prob\left[GL^{A'\left(1^{|x|}, f(x), r\right)}(n, \epsilon/2) = x\right] \geq \Omega\left(\epsilon^2/n\right).$$

Accordingly, we have $Prob_{x \leftarrow X}\left[A(f(x)) = x\right] \geq (\epsilon/2) \cdot \Omega\left(\epsilon^2/n\right)$, which is non-negligible because ϵ is non-negligible. This contradicts the one-wayness of f. □

We turn to proving lemma 20; But first we'll do a number of warmups.

7.1 Warmup 1

Suppose $h(r)$ agrees with $\langle x, r \rangle$ everywhere. see Figure 3 on page 8.

Note $\langle x, e_i \rangle = x_i$, where $e_i =$ is a vector with all entries being 0 except for the i-th entry which is 1. Now define an algorithm as follows:

- Set $z_i = h(e_i)$ for $i \in [n]$.
- Output $z_i, \ldots, z_n$.

Clearly, the above algorithm always output x.

7.2 Warmup 2

Consider agreement $1 - \text{negl}(n)$, instead of everywhere. That is, let $E_x = \{r | h(r) = \langle x, r \rangle\}$ then $\frac{|E_x|}{2^n} \geq 1 - \text{negl}(n)$. see Figure 4 on page 8.

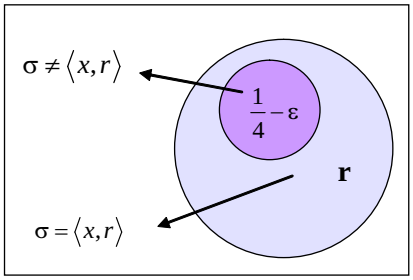

Figure 5: $h(r)$ doesn't agree with $\sigma = \langle x, r \rangle$ with probability $1/4 - \epsilon$ over r_n

Now this technique doesn't work since h can be mistaken on some or all the e_i vectors. The key idea is random self reducibility: we can reduce computing $\langle x, r \rangle$ at specific points (e.g. $e_1, \ldots, e_n$) to computing it at random points (where h will be correct with high probability).

Specifically, $\langle x, e_i \rangle = \langle x, r \rangle \oplus \langle x, r \oplus e_i \rangle$, and if r is random then so is $r \oplus e_i$. Now define an algorithm as follows:

- Choose $r \leftarrow \{0,1\}^n$.
- Set $z_i = h(r) \oplus h(r \oplus e_i)$ for $i \in [n]$.
- Output $z_1, \ldots, z_n$.

Since $h(r)$ (resp. $h(r \oplus e_i)$) agrees with $\langle x, r \rangle$ (resp. $\langle x, r \oplus e_i \rangle$) with probability $1 - \text{negl}(n)$, using union bound[2] we conclude that $Pr_r[z \neq x] \leq (n+1) \cdot \text{negl}(n) = \text{negl}(n)$.

7.3 Warmup 3

Consider agreement $3/4 + \epsilon$, where ϵ is non-negligible. That is, $\frac{|E_x|}{2^n} \geq \frac{3}{4} + poly(n)$. see Figure 5 on page 9.

Note that the error for each coordinate i using the previous scheme is:

$$\begin{aligned} Prob_r[h(r) \oplus h(r \oplus e_i) \neq x_i] &\leq Prob_r[h(r) \neq \langle x, r \rangle] + Prob_r[h(r \oplus e_i) \neq \langle x, r \oplus e_i \rangle] \\ &\leq (1/4 - \epsilon) + (1/4 - \epsilon) = 1/2 - 2\epsilon \end{aligned}$$

Therefore we cannot use a simple union bound as in the previous scheme, we will need to reduce the error for each coordinate i.

Now define an algorithm as follows:

- Choose $r_1, \ldots, r_t \leftarrow \{0,1\}^n$ $\left(t = O\left(\frac{1}{\epsilon^2} logn\right)\right)$.
- Set $z_i = maj_j\{\zeta_j\}$ for $i \in [n]$ and $\zeta_j = h(r_j) \oplus h(r_j \oplus e_i)$.
- Output $z_1, \ldots, z_n$.

Note by Chernoff bound[3], $Prob_r[z_i \neq x_i] \leq 1/2n$. So by union bound, we know $Prob_r[z \neq x] \leq n \cdot (1/2n) = 1/2$.

However, when $\frac{|E_x|}{2^n}$ is below or equal 3/4 the above algorithm does not work anymore and new ideas are needed.

[2]The Union bound states that $Pr[A \vee B] \leq Pr[A] + Pr[B]$, for any two (possibly dependent) events A, B.

[3]Chernoff's bound states that the probability that the average of t independent experiments is "γ -far" from its expectation is of the order $2^{-O(\gamma^2 t)}$. More formally, let Z_j be the indicator variable which is 1 if $\zeta_j = x_i$, where j = 1, . . . , t. Then all Z_j are independent and $E[Z_j] \geq \frac{1}{2} + 2 \cdot \epsilon$. Then, if $Z = \sum_{i=1}^{t} Z_j$, we have $E[Z] \geq t \cdot (\frac{1}{2} + 2 \cdot \epsilon)$, and $Pr[Z < t/2] \leq 2^{-O(\gamma^2 t)}$.

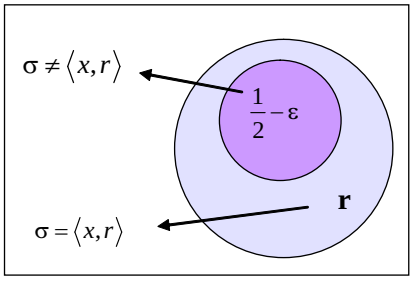

Figure 6: General case: $h(r)$ doesn't agree with $\sigma = \langle x, r \rangle$ with probability $1/2 - \epsilon$ over r_n

7.4 Warmup 4

Consider agreement $1/2 + \epsilon$, where $0 < \epsilon < \frac{1}{4}$ is a constant.

Here the idea is to choose at random $r_1, \ldots, r_t$ independent values such that for $j \in [t]$ we can $simultaneously$ guess $\langle x, r_j \rangle$ for all j with significant probability. Details follow:

- Choose $r_1, \ldots, r_t$, $(t = O(logn))$
- Choose $\sigma_1, \ldots, \sigma_k \leftarrow \{0,1\}$ (hope $\sigma_i = \langle x, r_i \rangle$ for all $i \in [t]$, which happens with probability 2^{-t} where t is logarithmic in n).
- Set $z_i = maj_j \{\zeta_j\}$ for $i \in [n]$ and $\zeta_j = \sigma_j \oplus h(r_j \oplus e_i)$.
- Output $z_1, \ldots, z_n$.

There are $O(logn)$ guesses, so the probability of guessing all the σ's right is good enough. (i.e. polynomial.)

Note by Chernoff bound $Prob_r[z_i \neq x_i] \leq 1/2n$. So by union bound, we know $Prob_r[z \neq x] \leq n \cdot (1/2n) = 1/2$.

Overall, we know $Pr[z = x] \geq \frac{1}{poly(n)} \cdot (1/2) = \frac{1}{poly(n)}$, where $\frac{1}{poly(n)}$ is the probability of guessing all the σ's right.

7.5 General Case

Proof. Consider agreement $1/2 + \epsilon$, where $\epsilon > 0$ is a polynomial fraction. see Figure 6 on page 10.

Here the idea is to choose $r_1, \ldots, r_t$ so that for $j \in [t]$ we can $simultaneously$ guess $\langle x, r_j \rangle$ for all j with probability $\Omega(\epsilon^2/n)$ (Note that this is much larger than 2^{-t}, which is what random independent guesses would give). Specifically, the strategy is to choose $r_1, \ldots, r_t$ from a random log t-dimensional subspace, where $t = O(n/\epsilon^2)$. Details follow:

- Choose $s_1, \ldots, s_k \leftarrow \{0,1\}^n$ $(k = log(t+1), t = O(n/\epsilon^2))$.
- Choose $\sigma_1, \ldots, \sigma_k \leftarrow \{0,1\}$ (hope $\sigma_i = \langle x, s_i \rangle$ for $i \in [k]$, which happens with probability $2^{-k} = \Omega(\epsilon^2/n)$).
- $\forall w \subseteq \{1, \cdots, k\}, w \neq \emptyset$, compute $r_w = \oplus_{j \in w} s_j$ and $\rho_w = \oplus_{j \in w} \sigma_j$ (if the hope occurs, then $\rho_w = \langle x, r_w \rangle$).
- Set $z_i = maj_w \{\rho_w \oplus h(r_w \oplus e_i)\}$ for $i \in [n]$ and output $z_1, \ldots, z_n$.

Note:

1. If for all i, $\sigma_i = \langle x, S_i \rangle$ then for all w, $\rho_w = \oplus_{j \in w} \sigma_j$.

2. The sequence $\{r_w\}_{w \subseteq \{1,\cdots,k\}}$ is pairwise independent. Consequently, for each i, the sequence $\{h(r_w \oplus e_i)\}_{w \subseteq \{1,\cdots,k\}}$ is a sequence of pairwise independent random vectors.

This means we can apply Chebyshev's inequality: Let X_w be the indicator variable which is 1 if $\rho_w \oplus h(r_w \oplus e_i) = x_i$, where $w \subseteq \{1,\cdots,k\}$. then all X_w's are pairwise independent and $E[X_w] \geq \frac{1}{2} + \epsilon$ and $\sigma_w^2 \geq \left(\frac{1}{2}\right)^2 - \epsilon^2$. Then, if $X = \sum_{w \subseteq \{1,\cdots,k\}} X_w$, we have $E[X] \geq t \cdot (\frac{1}{2} + \epsilon)$. If the hope occurs, by pairwise independence we know $Prob[z_i \neq x_i] = Prob[X < t/2] \leq \frac{\left(\frac{1}{2}\right)^2 - \epsilon^2}{t \cdot \epsilon^2} \leq \frac{1}{t \cdot \epsilon^2} \leq \frac{1}{2 \cdot n}$ and hence by union bound $Pr[z \neq x] \leq 1/2$.

Overall, we know $Pr[z = x] \geq \cdot \Omega\left(\epsilon^2/n\right)(1/2) = \Omega\left(\epsilon^2/n\right)$, which establishes the theorem. Therefore, as long as we have polynomially many samples ($O\left(n/\epsilon^2\right)$ pairwise independent samples), we are done. □

8 Reflections on GL

We can look at the result of the Goldreich-Levin Theorem in a number of ways:

Convertings unpredictability into pseudorandomness: This viewpoint corresponds to what we were doing here. specifically, if we cannot predict x with probability $\geq \epsilon$, then we cannot compute $\langle x, r \rangle$ for a random r with probability $> 1/2 + \epsilon$.

Learning: We can also view the Goldreich-Levin Theorem as learning a noisy linear function. Specifically, consider an oracle for a linear function which may sometimes return wrong answers, and set the goal to be learning the real function.
On the other hand, we can also view the theorem as learning the large Fourier coefficients of a Boolean function, with the underlying (Fourier) transform being a decomposition of a Boolean function into linear functions. This subroutine turn out to be very useful for learning a number of other natural classes of functions (e.g. DNF formula).

Coding theory: We can also relate the theorem to error-correcting codes, viewing $\langle x, \cdot \rangle$ as an encoding of x (of length 2^n - this is called a $Hadamard$ core), and view the wrong answers from the aforementioned oracle as transmission errors. Then we see that the Goldreich-Levin Theorem gives us kind of very fast decoding algorithm, that runs in time poly logarithmic in the length of the codeword.

References

[1] Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudo-random bits. SIAM J. Comput., 13(4):850–864, 1984.

[2] O. Goldreich and L. A. Levin. A hard-core predicate for all one-way functions. In STOC '89: Proceedings of the twenty-first annual ACM symposium on Theory of computing, pages 25–32, New York, NY, USA, 1989. ACM Press.

[3] Johan Hastad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudo- random generator from any one-way function. SIAM J. Comput., 28(4):1364–1396, 1999.

[4] Oded Goldreich. Foundations of Cryptography Volume I Basic Tools. Cambridge University press, 2006.