1 Review from last time:

Definition 1 \( f \) is a one-way function if

1. \( f \) is computable in polynomial time.

2. For all PPT algorithms \( A \), there exists a negligible function \( \epsilon \) such that for all sufficiently large \( n \), we have

\[
Pr_{A,x \leftarrow \{0,1\}^n}[A(f(x)) \in f^{-1}(f(x))] \leq \epsilon(n). \tag{1}
\]

Observation 2 If \( f \) is a one-way permutation, the definition above can be changed by replacing equation (1) with:

\[
Pr_{A,x \leftarrow \{0,1\}^n}[A(f(x)) = x] \leq \epsilon(n). \tag{2}
\]

Theorem 3 One-Way Functions exist iff Efficient Pseudo-Random Generators (PRGs) exist.

Last time we proved that any efficient PRG \( G \) is also a one-way function. Proving the forward direction of the theorem is much more involved. The plan for today is to show instead that if one-way permutations exist, then efficient PRGs exist. The fact that one-way permutations are used instead helps us in two ways. First, we can make use of the definition in Observation 2. Second, if \( f : \{0,1\}^n \rightarrow \{0,1\}^n \) and \( x \) is uniformly chosen in \( \{0,1\}^n \), then the distribution of \( f(x) \) is also uniform in \( \{0,1\}^n \). Before proving our desired result, we will need some prior definitions and theorems.

2 Hardcore bits

Definition 4 The function \( b : \{0,1\}^* \rightarrow \{0,1\} \) is a hard-core predicate for the one-way function \( f \) if for all PPT algorithm \( A \), there is a negligible function \( \epsilon \) such that for all sufficiently large \( n \), we have

\[
Pr_{x \leftarrow \{0,1\}^n}[A(f(x)) = b(x)] \leq \frac{1}{2} + \epsilon(n). \tag{3}
\]

Observation 5 Most commonly, \( b \) is called a hard-core predicate, but in class and hereinafter, we will call \( b \) a hard-core bit.

Theorem 6 If \( b \) is a hard-core bit for the one-way permutation \( f : \{0,1\}^n \rightarrow \{0,1\}^n \), then the function \( G : \{0,1\}^n \rightarrow \{0,1\}^{n+1} \) defined by \( G(x) = f(x)|b(x) \) (concatenation of \( b(x) \) to \( f(x) \)) is a PRG that maps any value \( x \in \{0,1\}^n \) to some value in \( \{0,1\}^{n+1} \) (i.e. one-bit stretch).

Proof First, observe that \( f(x) \) is next-bit unpredictable because if \( x \leftarrow \{0,1\}^n \) is chosen uniformly, then the distribution of \( f(x) \) is also uniform in \( \{0,1\}^n \) implying that knowing the first \( i \) bits of \( f(x) \) does not help in predicting the \( i+1 \) bit with probability better than \( \frac{1}{2} + \frac{1}{n} \) for any \( k \). Second, note that from the definition of hardcore bit, for any PPT algorithm \( A \), inequality (3) is satisfied; this implies that no algorithm can predict \( b(x) \) (the last bit of \( G(x) \)) even if it knows \( f(x) \) (the previous bits of \( G(x) \)). Both points imply that \( G \) is next-bit unpredictable. From a theorem we proved last class, we conclude that \( G \) is a PRG.

The theorem above shows how to obtain one-bit stretch in randomness. We can extend the construction to obtain \( k \) bits of stretch as follows:

Define for any \( j \in \mathbb{Z}_+ \), the function \( f^{(j)} = f \circ f \circ \ldots \circ f \), which is \( f \) composed with itself \( j \) times.
Theorem 7  If $f : \{0, 1\}^l \rightarrow \{0, 1\}^l$ is a one-way function with an efficiently computable hardcore bit $b$, then the function $G : \{0, 1\}^l \rightarrow \{0, 1\}^n$ defined by $G(x) = b(f(x))|b(f(x) \cdot 2)|\ldots|b(f(x))|b(x)$ is a PRG for all $n$, polynomial in $l$ (i.e. $n = P(l)$ for some polynomial $P$).

Proof  We will assume the opposite, which is that $G$ is not a PRG. Then $G$ is next-bit predictable. This implies there exists a PPT algorithm $P$ that can predict bit $i$ of the output of $G$ for some $i$, i.e.

$$Pr_{x \sim \{0, 1\}^l}[P(b(f(0^n-1)(x))|b(f(0^n-2)(x))|b(0^n-1)(x)) = b(f(0^n-i)(x))] \geq \frac{1}{2} + \frac{1}{n^k}$$

for some constant $k$. After setting $y = f(0^n-1)(x)$, notice that because $f$ is a permutation (and so is $f(0^n-i)$), then $y$ is uniform in $\{0, 1\}^l$ if $x$ is. Then we can rewrite this equation as

$$Pr_{y \sim \{0, 1\}^l}[P(b(f(i-1)(y))|b(f(i-2)(y))|b(f(y)) = b(y)) \geq \frac{1}{2} + \frac{1}{n^k}.$$ 

Having (5), we will construct a PPT algorithm $P'$, such that $Pr_{y \sim \{0, 1\}^l}[P'(f(y)) = b(y)] \geq \frac{1}{2} + \frac{1}{n^k}$, contradicting the fact that $b$ is a hardcore bit of $f$. Algorithm $P'$, on an input $x$, will compute $f(i)(x)$ for $1 \leq j \leq i - 2$. Then $P'$ computes $b(f(i)(x))$ for all $0 \leq j \leq i - 2$, obtains the concatenation $z = b(f(i-2)(x))|b(f(i-3)(x))|\ldots|b(f(2)(x))|b(f(x))|b(x)$, applies algorithm $P$ to $z$ and finally outputs the result of $P(z)$. Note the following two points. First, if $x = f(y)$, then it is clear from (5) that the probability that $P'$ succeeds is $\frac{1}{2} + \frac{1}{n^k}$. Second, because $b$ is efficiently computable, then $P'$ is a PPT algorithm. Both points imply that $P'$ successfully computes $b(y)$ from input $f(y)$ with at least probability $\frac{1}{2} + \frac{1}{n^k}$, i.e. $b$ is not a hardcore bit (\(\implies\)). Hence $G$ is a PRG. ■

The above theorem shows how to construct a PRG from a hardcore bit for a one-way function, but we are not even sure a hardcore bit exists. In the next section, we show that for any one-way permutation $f$, we can construct a one-way permutation $f'$ from $f$, and a hardcore bit $b$ for $f'$.

3 Goldreich-Levin Theorem

Theorem 8 (Goldreich-Levin) If $f$ is a one-way function, then $b : \{0, 1\}^* \rightarrow \{0, 1\}$, defined by $b(x, r) = \langle x, r \rangle$, is a hardcore bit for the one-way function $f'$ defined by $f'(x, r) = \langle f(x), r \rangle$, with $|x| = |r|$.

As we said before, the proof of this theorem is quite involved. In lecture, we saw the proof for the case of a one-way permutation $f : \{0, 1\}^l \rightarrow \{0, 1\}^l$. Also, we made the simplifying assumption that $f$ is a one-way permutation in the circuit complexity model. The proof will go by contradiction by assuming there is a PPT algorithm $A$ that can predict $b(x, r)$ from $f'(x, r)$. From our last assumption, we can assume $A$ is a deterministic algorithm.

Finally, before starting with the proof, convince yourself that if $f : \{0, 1\}^l \rightarrow \{0, 1\}^l$ is a one-way permutation, then $f' : \{0, 1\}^{2l} \rightarrow \{0, 1\}^{2l}$, defined as in the theorem for $|x| = |r|$, is also a one-way permutation. It is clear that $f'$ is a permutation of $\{0, 1\}^{2n}$ if $f$ is a permutation of $\{0, 1\}^n$. It is also true that $f'$ is a one-way function if $f$ is one-way. This is an easy exercise (prove that if there is a PPT algorithm that inverts $f'$ with non-negligible probability, then one can construct a PPT algorithm that inverts $f$ with non-negligible probability).

Proof (Simplified Version) We assume the opposite, i.e. there is a poly-time deterministic algorithm $A$ such that $Pr_{x, r}[A(f(x), r) = b(x, r) = \langle x, r \rangle] \geq \frac{1}{2} + \epsilon$ for some $\epsilon = \epsilon(l) \geq \frac{1}{n^k}$ where $k$ is a constant and $l = |x| = |r|$ is the number of bits of $x$ and $r$. 

2
Let us define $h_x(r) = A(f(x), r)$ and call good to a value $x$ if $Pr_r[h_x(r) = \langle x, r \rangle] \geq \frac{1}{2} + \frac{\epsilon}{2}$. We claim that there are at least $\epsilon/2$ good values of $x$. In fact, assume this is not the case, so there are at most $\epsilon/2$ good values of $x$. Observe that for bad values of $x$, the probability that $A$ guesses $b(x, r)$ correctly is at most $\frac{1}{2} + \frac{\epsilon}{2}$. Therefore

$$Pr_{x,r}[A(f(x), r) = \langle x, r \rangle] = Pr_x[x \text{ is good }]Pr_r[A(f(x), r) = \langle x, r \rangle] + Pr_x[x \text{ is bad }]Pr_r[A(f(x), r) = \langle x, r \rangle]$$

$$< \frac{\epsilon}{2} \times 1 + 1 \times \left( \frac{1}{2} + \frac{\epsilon}{2} \right)$$

$$= \frac{1}{2} + \epsilon$$

which is a contradiction with our initial assumption. Therefore, there are at least $\frac{\epsilon}{2}$ good values of $x$, as desired.

Our goal now is to obtain a PPT algorithm $B$ that inverts $f$ for a non-negligible fraction of the inputs, therefore proving that $f$ is not a one-way function. In fact, we will construct $B$ so that it outputs $x$ on input $z = f(x)$ if $x$ is good. Consider the function $h : \{0,1\}^l \rightarrow \{0,1\}$, defined by $h(r) = A(z, r)$. To proceed, we translate the functions we are working with to the Boolean analysis notation (i.e. bit 1 becomes $-1$ and bit 0 becomes $+1$). Observe that if $S_x \subseteq \{1\}$ is the set that defines $x$ ($j \in S_x$ iff the $j$ bit of $x$ is 1), then $(x, r)$ becomes $\chi_{S_x}(r)$ in Boolean notation. Therefore, if $x$ is good, we have that $Pr_r[h(r) = \chi_{S_x}(r)] \geq \frac{1}{2} + \frac{\epsilon}{2}$, or $h(S_x) \geq \epsilon$. This makes it simple to construct PPT $B$. In fact, $B$ first runs the Goldreich-Levin algorithm to find all the heavy Fourier coefficients of $h$, the ones for which $h(S) > \frac{\epsilon}{2}$. For those sets, we have that $Pr_r[h(r) = \chi_{S}(r)] > \frac{1}{2} + \frac{\epsilon}{2}$. Thus, if $z = f(x)$ with good $x$, then $S_x$ is among the sets outputted by the Goldreich-Levin algorithm with high probability. Then $B$ can compute $f(x)$ for all $x$ for which $S_x$ was outputted by the Goldreich-Levin algorithm and output a particular $x_0$ if $f(x_0) = z$. Otherwise, $B$ just outputs a random value.

The probability that $B$ succeeds on $z = f(x)$, for good $x$, can be made at least $1/2$ if we set the confidence parameter $\delta = 1/2$ in Goldreich-Levin. Note that since at least $\frac{\epsilon}{2} \geq \frac{1}{2\sqrt{n}}$ fraction of the inputs $x$ are good, then $B$ succeeds with probability at least $\frac{1}{2} + \frac{\epsilon}{2}$ for a random $z = f(x)$. This shows that $f$ is not a one-way permutation ($\Rightarrow \Leftarrow$). Hence, we conclude that the theorem is true.

**Observation 9** It may not be clear that $B$ runs in polynomial time, because we do not know how many heavy coefficients $h(S)$ there are. However, remember that there are at most $\text{poly}(\frac{1}{\epsilon})$ of these coefficients, and since $\epsilon \geq \frac{1}{2\sqrt{n}}$, then this is also polynomial, as desired.

### 4 For next lecture

Next lecture, we will study the Nisan Pseudorandom Generator. As a warm-up, you might want to think of the following definition and try to prove the next theorem.

**Definition 10** A collection of subsets $S_1, S_2, \ldots, S_m \subseteq [d] = \{1,2,\ldots,d\}$ is a $(l, a)$-design if

- $|S_i| = l$ for all $1 \leq i \leq m$.
- $|S_i \cap S_j| \leq a$ for all $1 \leq i \neq j \leq m$.

**Theorem 11** There exists a $(l, a)$-design with $a = \gamma \log m$ and $d = O(l^2/a)$ for some $m \in \mathbb{Z}_+$ and all $\gamma > 0$. 

3