CSA E0 235: Cryptography February 22,2016

Scribe for Lecture 13

Instructor: Arpita Patra *Submitted by: Ishan Agarwal*

1 Recall of Previous Lectures

In the last lecture we defined one way functions and hard core predicates. We will recall these definitions here. We also realized the non-triviality of finding them. We showed, in the last lecture, the Goldreich Levin theorem that states that if there exists an OWF[i] then there exists one with a hard-core predicate.

2 Aim of this Lecture

We also outlined a road map for future progress; which was to show that **OWF with HCP implies existence of a *minimally expanding*[ii] PRG which in turn implies the existence of a PRG with any given polynomial expansion factor**. These two implications shall be shown in this lecture. This would enable us to finally conclude that PRFs exist just from the assumption of OWFs existing, using the previously proven result that length doubling PRG implies PRF exists. Thus there are two basic goals for this lecture:

- Prove: OWF with HCP $\Rightarrow$ PRG with minimal expansion factor
- Prove: PRG with minimal expansion factor $\Rightarrow$ PRG with *any given poly* expansion factor

Showing these two results would allow us to complete our chain of reasoning.

3 Pseudo Random Generators (PRGs)

A pseudo random generator $\mathbf{G}$ is a deterministic algorithm for transforming a short, uniform string called the seed into a longer, uniform looking output string. It should also do this within polynomial time. The basic criterion is that a good pseudo random generator should be such that for any distinguisher $\mathcal{D}$, the probability that $\mathcal{D}$ returns 1 when given the output of the pseudo random generator should be close to the probability that $\mathcal{D}$ returns 1 when given a uniform string of the same length. Let $l(n)$ be a polynomial in n and $\mathbf{G}$ be a deterministic polynomial time algorithm that is such that $\forall n$ and $\forall s \in_R \{0,1\}^n$, $\mathbf{G}$ output a string of length $l(n)$. $\mathbf{G}$ is said to be a PRG if the following conditions hold:

- **Expansion:** $l(n)$ should be of length $>$ length of n $\forall n > 0$

[i] actually we used the stronger notion of a one way permutation

[ii] expands from n bit seed to an output of $n+1$ bits

- **Pseudorandomness:** for every probabilistic polynomial time algorithm $\mathcal{D}$, there is a negligible function $\mathsf{negl}(n)$ such that $|\Pr[\mathcal{D}(r) = 1] - \Pr[\mathcal{D}(\mathbf{G}(s) = 1]| \leq \mathsf{negl}(n)$. Here $r \in_R \{0,1\}^{l(n)}$.

$l(n)$ is known as the expansion factor of the PRG.

4 One Way Functions (OWFs)

Informally, a one way function is one that is easy to evaluate for all possible inputs but difficult to invert for any given output. Given a value of the function calculated by using an input drawn uniformly at random from the domain, it should be only negligibly likely for an algorithm that runs in polynomial time to be able to find *any* element in the functions domain which maps to this value. Note that not only is it difficult to find the input used but rather it is hard to find any input that gives this output. In particular, a one way function need not be injective.

Definition. A function $f : \{0,1\}^n \to \{0,1\}^n$ is said to be one-way if it is computable in polynomial time and for every probabilistic polynomial time adversary $\mathcal{A}$, there is a negligible function $\mathsf{negl}(n)$ such that

$$\Pr[\mathcal{A}(f(x), 1^n) \in f^{-1}(f(x))] \leq \mathsf{negl}(n) \text{ for } x \in_R \{0,1\}^n$$

The probability is taken over the randomness associated with the choice of x and also any randomness used by the adversary $\mathcal{A}$.

We can equivalently define OWFs via the means of an indistinguishability based game. This form would be convenient while proving statements using reduction. Let us call this game $\mathsf{Invert}_{\mathcal{A},f}(n)$. It shall be in the same spirit as other indistinguishability games seen previously and is outlined below:

- A verifier picks a value y at random from $\{0,1\}^n$ and gives this value to the adversary $\mathcal{A}$.
- $\mathcal{A}$ returns a value x to the verifier.
- If $f(x) = y$, then the game outputs 1, else the game outputs 0.

f is one-way if:

- it is computable in polynomial time $\forall$ input in it's domain
- $\Pr[\mathsf{Invert}_{\mathcal{A},f}(n) = 1] \leq \mathsf{negl}(n)$.

It is to be noted that if we remove the polynomial bound on the adversary's computing power then no function would remain one way. This is because the adversary would simply keep trying inputs in the domain till one of them matches the output.

Digression: Why do we believe OWFs exist?

One-way functions are widely believed to exist because polynomial time inverting algorithms have not been found for several well known functions[iii].

However, the existence of OWFs remains a conjecture till date. In fact the belief in existence of OWFs is even stronger than assuming $P \neq NP$. If f is a one-way function, then the inversion of f would be a problem whose output is hard to compute (by definition) but easy to check (just by computing f on it). Thus, the existence of a one-way function implies that $FP \neq FNP$, which in turn implies that $P \neq NP$. However, it is not known whether $P \neq NP$ implies the existence of one-way functions. If the inversion of a function f is NP-complete for example, it does not immediately follow that f is an OWF. For f to be an OWF, every probabilistic polynomial time algorithm must fail to solve *almost every* instance of the inversion problem *with high probability*. But for the inversion problem to be NP-complete, we only require that for every probabilistic polynomial time algorithm there is just *one* instance of the problem which it fails to solve.
This is basically because by our definition of OWF, the function must be "hard to invert" in the average-case, rather than worst-case sense. This is different from much of complexity theory (e.g., NP-hardness), where the term "hard" is meant in the worst-case. That's why even if some candidates for one-way functions are known to be NP complete, it does not imply that they are one way. The latter property is only *based on the lack of known algorithm to solve the problem.*

5 Hard Core Predicates (HCPs)

Before actually defining hard core predicates we motivate the need of such a concept. We eventually will try to prove that OWF with HCP implies PRG. Why does not simply defining our PRG as a particular pseudo random function work? This is actually because an OWF may reveal quite a lot about it's input bits. Suppose f is an OWF. Given $f(x)$, it might be possible to determine a large portion of x even though f is an OWF. On the other hand, since f is an OWF, it is difficult to determine x *completely* given $f(x)$. It is intuitive that there should be *something* about x that $f(x)$ keeps hidden or else $f(x)$ would not be an OWF. Hard-core predicates are a formal way of stating this fact.
Informally, a hard-core predicate is a *single* bit of information about x than no adversary can guess with non-negligible probability of success, when given access to only the value $f(x)$.
We now define a hard core predicate. A function $hc : \{0,1\} \mapsto \{0,1\}$ is a hard core predicate of a function f if hc can be computed in polynomial time, and for every probabilistic polynomial-time algorithm $\mathcal{A}$ there is a negligible function negl such that:

$$\Pr[\mathcal{A}(f(x) = hc(x)] \leq \frac{1}{2} + \mathsf{negl}$$

We also, as always, have an equivalent definition based on a game that we would use in the proof below.

[iii] for example the subset sum based function

- The verifier selects an input, x, uniformly at random from the domain of the OWF and computes $f(x)$. This $f(x)$ is given to the challenger.
- The challenger attempts to find $hc(x)$ given only $f(x)$. The guess of the challenger is sent back to the verifier.
- The verifier has x and so computes $hc(x)$ easily. Then it is checked if the guess of the challenger was correct.

The game outputs 1 if the challenger guessed correctly, else it outputs 0. hc is considered a hard core predicate for the OWF if :

- $hc(x)$ can be computed in polynomial time $\forall x$ in the domain of the OWF.
- Probability, given any probabilistic polynomial time adversary, of the game outputting $1 \leq \frac{1}{2} + \mathsf{negl}$

6 PRG with minimal expansion factor from OWF and HCP

Theorem 6.1 *Given a one-way permutation f with a hard-core predicate hc, the function $G(x) = f(x)||hc(x)$ is a PRG. Given $x \in_R \{0,1\}^n$, this is a PRG with expansion factor $n+1$.*

Proof (by reduction on the contrapositive statement)

We prove that given there is a probabilistic polynomial time distinguisher $\mathcal{D}$ for the claimed PRG G, we can construct a probabilistic polynomial time adversary $\mathcal{A}$ which can break the hard-core predicate hc. By the definition of $\mathcal{D}$ the following is true for $r \in_R \{0,1\}^{n+1}$ and $s \in_R \{0,1\}^n$ for infinitely many $n \in \mathbb{N}$ and for some polynomial $p(n)$.

$$\begin{aligned}\frac{1}{2} + \frac{1}{p(n)} \leq & \mathsf{Pr}\left[\mathcal{D}(r) = 1\right] - \mathsf{Pr}\left[\mathcal{D}(G(s)) = 1\right] \\ = & \frac{1}{2}\mathsf{Pr}\left[\mathcal{D}(f(s)||hc(s)) = 1\right] + \frac{1}{2}\mathsf{Pr}\left[\mathcal{D}(f(s)||hc(s)') = 1\right] - \mathsf{Pr}\left[\mathcal{D}(f(s)||hc(s)) = 1\right] \\ & \text{(Since } f \text{ is a permutation, } r = f(s)||r_n \text{ for some } s \in_R \{0,1\}^n. \text{ Also, } r_n = hc(s) \text{ w.p. } \frac{1}{2}) \\ = & \frac{1}{2}\left(\mathsf{Pr}\left[\mathcal{D}(f(s)||hc(s)') = 1\right] - \mathsf{Pr}\left[\mathcal{D}(f(s)||hc(s)) = 1\right]\right) \qquad (1)\end{aligned}$$

This implies that $\mathcal{D}$ can distinguish between the correct and incorrect $hc(s)$ appended at the end of $f(s)$ with probability $\frac{2}{p(n)}$.

We now construct an adversary $\mathcal{A}$ for the distinguishability game of the hard-core predicate hc as follows. Given $\mathcal{A}'$ receives a string $f(s)$, $s \in_R \{0,1\}^n$, it appends a bit $r \in_R \{0,1\}$ and sends $f(s)||r$ to $\mathcal{D}$. If $\mathcal{D}$ sends back the bit 1 (denoting a PRG string), $\mathcal{A}$ returns the bit r as the hard-core predicate. Otherwise, it returns r'. Now, probability of $\mathcal{A}$ guessing

in $hc(s)$ correctly is

$$\begin{aligned}
\mathsf{Pr}\left[A(f(s)) = hc(s)\right] &= \frac{1}{2}(\mathsf{Pr}\left[A(f(s)) = hc(s)|r = hc(s)\right] + \mathsf{Pr}\left[A(f(s)) = hc(s)|r' = hc(s)\right]) \\
&= \frac{1}{2}(\mathsf{Pr}\left[\mathcal{D}(f(s)||hc(s)) = 1\right] + \mathsf{Pr}\left[\mathcal{D}(f(s)||hc'(s) = 0\right]) \\
&= \frac{1}{2}(1 + \mathsf{Pr}\left[\mathcal{D}(f(s)||hc(s)) = 1\right] - \mathsf{Pr}\left[\mathcal{D}(f(s)||hc'(s) = 1\right]) \\
&\geq \frac{1}{2} + \frac{1}{p(n)} \text{ Using inequality 1 from above.}
\end{aligned}$$

■

7 PRG with poly expansion factor from PRG with minimal expansion factor

Theorem 7.1 *Given a PRG, G with expansion factor $n+1$, there exists a PRG, G^* with expansion factor $p(n)$ for any polynomial p.*

Proof By the previously proved theorem, we have a PRG, G, with expansion factor $n+1$. Consider that we want to construct G^* such that it is a PRG and has expansion factor $n + p(n)$ where p is some polynomial. On input $s \in_R \{0,1\}^n$ G^* does:

- Set $s_0 = t_0 = s$.
- Then $\forall i$ in $1, 2, ..., p(n)$ do:
  - (a) Let s_{i-1} be the first n bits of t_{i-1}, and let σ_{i-1} denote the remaining $i-1$ bits. (When $i = 1, s_0 = t_0$ and σ_0 is the empty string.)
  - (b) Set $t_i = G(s_{i-1})||\sigma_{i-1}$.
- Output $t_{p(n)}$.

We show that G^* is a PRG. For any n and $0 \leq j \leq p(n)$, let H_n^j be the distribution on strings of length n+p(n) defined as follows: choose $t_j \in_R \{0,1\}^{n+j}$, then run G^* starting from iteration $j+1$ and output $t_{p(n)}$. (When $j = p(n)$ this means we simply choose $t_{p(n)} \in_R \{0,1\}^{n+p(n)}$ and output it.) The crucial observation is that H_0^n corresponds to outputting G^*(s) for $s \in_R \{0,1\}^n$, while $H_{p(n)}^n$ corresponds to outputting a uniform $(n+p(n))$ bit string. Let r be a random $n + p(n)$ bit string, while r is a random n bit string. Fixing any polynomial-time distinguisher D, this means that:

$$|P[D(G^*(s) = 1] - P[D(r) = 1]| = |P[D(G^*(H_n^0)^{\text{iv}} =] - P[D(H_n^{p(n)}) = 1]|$$

We shall show that the above is negligible, hence proving that G^* is a PRG. Fix D as above and consider D_1 which behaves as follows when given a string $w \in \{0,1\}^{n+1}$:

[iv] note that D(a distribution) is used throughout to mean D(t) where t is drawn uniformly at random from this distribution.

- choose $j \in_R \{0,1,2,...,p(n)\}$.
- choose $\mu(j) \in_R \{0,1\}^{j-1}$.
- set $t_j = w||\mu(j)$ and then run G^* from iteration $j+1$ to find $t_{p(n)}$. Output $D(p(n))$.

Clearly D_1 runs in polynomial time. Fix n and consider D_1 chooses $j = J$. If w is uniform, then t_J is uniform and so the distribution on $t = t_p(n)$ is exactly that of distribution H_n^J. That is:

$$P[D_1(w) = 1|j = J] = P[D(H_n^J) = 1]$$

Since each value of J is chosen with equal probability:

$$P[d_1(w) = 1] = \frac{1}{p(n)} \sum_{J=1}^{p(n)} P[D_1(w) = 1|j = J] = \frac{1}{p(n)} \sum_{J=1}^{p(n)} P[D(H_n^J) = 1]$$

On the other hand, say D_1 chooses $j = J$ and $w = G(s)$ for $s \in_R \{0,1\}^n$. Defining $t_{J-1} = s||\mu_J$, we see that t_{J-1} is uniform and so the experiment involving D_1 is equivalent to running G^* from iteration J to compute $t_p(n)$. That is, the distribution on $t = t_p(n)$ is now exactly that of the distribution H_n^{J-1}, and so:

$$P[D_1(w) = 1|j = J] = P[D(H_n^{J-1}) = 1]$$

$$\begin{aligned} P[D_1(G(s)) = 1] =& \frac{1}{p(n)} \sum_{J=1}^{p(n)} P[D_1(G(s)) = 1|j = J] \\ =& \frac{1}{p(n)} \sum_{J=1}^{p(n)} P[D(H_n^{J-1}) = 1] \\ =& \frac{1}{p(n)} \sum_{J=0}^{p(n)-1} P[D(H_n^J) = 1] \end{aligned}$$

Consider now $w \in_R \{0,1\}^{n+1}$,

$$\begin{aligned} & |P[D_1(G(s)) = 1] - P[D_1(w) = 1]| \\ =& \frac{1}{p(n)} | \sum_{J=0}^{p(n)-1} P[D(H_n^J) = 1] - \sum_{J=1}^{p(n)} P[D(H_n^J) = 1] \\ =& \frac{1}{p(n)} |P[D(H_n^0) = 1] - P[D(H_n^{p(n)}) = 1]| \end{aligned}$$

But as G is a PRG we know that:

$$|P[D_1(G(s)) = 1] - P[D_1(w) = 1]| \leq \mathsf{negl}(n)$$

Hence:

$$|P[D(H_n^0) = 1] - P[D(H_n^{p(n)}) = 1]| \leq p(n) \cdot \mathsf{negl}(n) = \mathsf{negl}(n)$$

But we showed that:

$$|P[D(H_n^0) = 1] - P[D(H_n^{p(n)}) = 1]| = |P[D(G^*(s) = 1] - P[D(r) = 1]$$

Hence $G^* = f^{(l)}(s)||hc(f^{(l-1)}(s))||...||hc(s)$ is a PRG completing the proof. ■

Digression: Hybrid arguments

In the above proof we used hybrid arguments. A hybrid argument is a basic tool for proving indistinguishability when a basic primitive is (or several different primitives are) applied multiple times. Somewhat informally, the technique works by defining a series of intermediate *hybrid distributions* that bridge between two *extremal distributions* that we wish to prove indistinguishable. This is often useful because it might be difficult to construct a way to emulate such widely different distributions to another adversary; but this is what we would want to do in a reduction based proof. In the proof above, these extreme distributions correspond to the output of $G*$ and a random string. To apply the hybrid based proof technique, three conditions should hold:

- The extremal distributions should match the original cases of interest. In the proof above, H_n^0 was equal to the distribution induced by G , while $H_n^{p(n)}$ was the uniform distribution.
- It must be possible to translate the capability of distinguishing consecutive hybrid distributions into breaking some underlying assumption. In this case we essentially showed that distinguishing H_n^i from H_n^{i+1} was equivalent to distinguishing the output of G from a random string.
- The number of hybrid distributions should be polynomial because we are assuming only polynomially bounded computation power for all our algorithms and we would require, in our proofs, for an adversary to emulate two hybrids for another adversary in another indistinguishability game. This requirement is also satisfied in the above proof.

8 Conclusion: Putting it all together

Thus we have seen how, assuming one way functions and hard core predicates exist, the existence of a pseudo random generator with minimal expansion factor is implied. From such a PRG we have shown how to construct a PRG with expansion factor being any given polynomial. Furthermore in lecture 11 we saw how a PRG, which outputs $2n$ bits given a seed n bits long, implies the existence of a PRF. Thus we have finally completed what was charted in the road map already discussed. We outline once again the entire chain of reasoning:

- Assume OWFs exist
- OWF exists implies OWF with HCP exists (by Goldreich Levin Theorem)[v]

[v] proved partially in lecture 12

- OWF with HCP implies PRG with minimal expansion factor (as proved in section 6 above)
- PRG with minimal expansion implies PRG with poly expansion (as proved in section 7 above)
- PRG with $2n$ expansion factor implies PRF[vi]

This result provides justification for the construction of many of the schemes we have previously discussed that assume the existence of PRFs/PRGs.
This brings us to the end of the first half of the course.

[vi] proved using hybrid arguments in lecture 11

References

[1] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography, Second Edition*. Chapman and Hall/CRC Cryptography and Network Security.

[2] Arpita Patra: *Lecture 13 (E0 235: Cryptography)*. Lecture Slides.