CS 600.442 – Modern Cryptography 09/19/2016

Lecture 3: Hard Core Predicates and Pseudorandomness

Instructor: Abhishek Jain *Scribe: David Li*

1 Hard Core Predicates

Last time, we covered the fact that $f_\times$ is a weak one way function, and in the review session, we showed how we can amplify a weak one way function into a strong one way function. Today, we will first examine what information one-way functions hide, and that will introduce us to the notion of hard core predicate.

The idea of a one-way function is very intuitive, but by themselves, they're not very useful. Why? Because it only guarantees that $f(x)$ will hide the preimage x, but no more than that! For instance, if we have a one-way function f, then

$$f'(x) = (f(x), x_{[1,\cdots,n/2]}),$$

where $x_{[1,\cdots,n/2]}$ are the first half bits of x, is also a one-way function! In fact, a function f may not hide any subset of the input bits, and still be a one-way function. More generally, for any non-trivial function $a(\cdot)$, $f(x)$ may not hide $a(x)$.

Hard Core Predicate: Intuition. A hard core predicate for a one-way function f is a function over its inputs $\{x\}$ and its output is a single bit. We want this function to be easily computed given x, but "hard" to compute given $f(x)$. Intuitively, this says that f can leak some (or many) bits of x, but does not leak the hard core bit. In other words, finding out the hard core bit, even *given* $f(x)$, is as difficult as inverting f. Of course, we need to be a little careful with "hard to compute" for a single bit, since it can be guessed correctly with probability $\frac{1}{2}$. So, intuitively, "hardness" should mean that it is impossible for any efficient adversary to gain any non-trivial advantage over guessing. We formalize this below:

Definition 1 (Hard Core Predicate) *A predicate $h : \{0,1\}^* \mapsto \{0,1\}$ is a **hard core predicate** for $f(\cdot)$ if h is efficiently computable given x and there exists a negligible function ν such that for every non-uniform PPT adversary $\mathcal{A}$ and $\forall n \in \mathbb{N}$,*

$$\mathbf{Pr}[x \leftarrow \{0,1\}^n : \mathcal{A}(1^n, f(x)) = h(x)] \leq \frac{1}{2} + \nu(n).$$

1.1 Hard Core Predicate via Inner Product

We now show construct hard core predicate for *any* one-way function using inner product.

Theorem 1 (Goldreich-Levin) *Let f be a one-way function (permutation). Then define $g(x) = (f(x), r)$ where $|x| = |r|$. Then, g is a one-way function (permutation) and $h(x, r) = \langle x, r\rangle$ is a hard-core predicate for f, where $\langle x, r\rangle = (\sum_i x_i r_i) \mod 2$.*

How should we prove this? If we use reduction, our main challenge is that our adversary $\mathcal{A}$ for h only outputs one bit, but our inverter $\mathcal{B}$ for f needs to output n bits. Amazingly, Goldreich and Levin proved that this can be done!

We start by considering two warmup cases, where we make assumptions on the adversary.

Assumption 1 *First, let's suppose that given $g(x,r) = (f(x), r)$, adversary $\mathcal{A}$ will* always *(with probability 1) output $h(x,r)$ correctly.*

Proof. We can use the property of the inner product to recover each bit of x one-by-one. Specifically, we construct an adversary $\mathcal{B}$ for f that works as follows: on input $f(x)$, for every $i \in [n]$, it sets e_i to be the ith standard basis vector for $\mathbb{R}^n$ (i.e., e_i is such that its ith bit is 1 but every other bit is 0). It then runs the adversary $\mathcal{A}$ on input $(f(x), e_i)$ to recover x_i^*. Now, $\mathcal{B}$ simply outputs $x^* = x_1^* \cdots x_n^*$. ■

Okay, that was fairly straightforward. Let's see what happens when the adversary doesn't output $h(x,r)$ correctly with probability 1.

Assumption 2 *Now assume $\mathcal{A}$ outputs $h(x,r)$ with probability $\frac{3}{4} + \varepsilon(n)$.*

Proof. [Informal Strategy] The main issue here is that our adversary $\mathcal{A}$ might not work on some specific inputs - such as e_i as in the previous case. We need the inputs to "look" random in order to mimic the expected distribution over the inputs of $\mathcal{A}$. So, we split our original single query into two queries such that each query looks random individually. Specifically, our inverter $\mathcal{B}$ computes $a := \mathcal{A}(f(x), e_i + r)$ and $b := \mathcal{A}(f(x), r)$ for $r \xleftarrow{\$} \{0,1\}^n$. Then, compute $c := a \oplus b$, where $\oplus$ is `xor`. By using a union bound, one can show that $c = x_i$ with probability $\frac{1}{2} + \varepsilon$, and we can then repeatedly compute x_i and take the majority to get x_i^* with $x_i^* = x_i$ with probability $1 - \text{negl}(n)$.

By repeating this process for every i, $\mathcal{B}$ can learn every x_i^* and output $x^* = x_1^* \cdots x_n^*$.

■

The full proof of **Theorem 1** will be on the first homework assignment. Note that this theorem is actually very important, even outside cryptography! It has applications to learning, list-decoding codes, etc.

1.2 Final Remarks on OWFs

One-way functions are necessary for most of cryptography. Nevertheless, they are often not sufficient for most of cryptography. In particular, it is known that many advanced cryptographic primitives cannot be constructed by making *black-box* use of one-way functions; however, full separations are not known.

Also, recall that we don't know if one-way functions actually exist. (We only have candidates based on assumptions such as hardness of factoring.) Now, suppose someone told you one-way functions exist (perhaps by an existence proof, and not a constructive one). Then, simply using that knowledge, could you create an explicit one-way function? Surprisingly, it *can* be done! Levin gives a proof here: `https://arxiv.org/pdf/cs/0012023.pdf`.

2 Pseudorandomness

Now we move into our second main topic in the study of cryptography: pseudorandomness.

2.1 Introduction

Our computers use randomness every day, but what exactly is randomness? How does your computer get this randomness? Some common sources of randomness are key-strokes, mouse movement, and power consumption, but the amount of randomness generated by these isn't a lot, and often a lot of randomness is required (such as for encryption).

This brings us to the fundamental question: Can we "expand" a few random bits into many random bits? There are many heuristic approaches to this, but this isn't good enough for cryptography. We need bits that are *"as good as truly random bits"* (to a PPT adversary). This isn't very precise, so let's define it formally.

2.2 Computational Indistinguishability and Prediction Advantage

Suppose we have n uniformly random bits, $x = x_1 \| \cdots \| x_n$, and we want to find a deterministic polynomial time algorithm G that outputs $n+1$ bits: $y = y_1 \| \cdots \| y_{n+1}$ and looks *"as good as"* a truly random string $r = r_1 \| \cdots \| r_{n+1}$. We call such a $G : \{0,1\}^n \to \{0,1\}^{n+1}$ a **pseudorandom generator**, or PRG for short.

But what does *"as good as"* really mean? Intuitively it means that there should be no obvious patterns and that it should pass all statistical tests a truly random string would pass (all possible k-length substrings should occur equally). But the key point is that we only need to address our adversary, so as long as there is no efficient test that can tell $G(x)$ and r apart, then that is enough!

This gives us the notion of **computational indistinguishability** of $\{x \leftarrow \{0,1\}^n : G(x)\}$ and $\{r \xleftarrow{\$} \{0,1\}^{n+1} : r\}$. We will use this notion and an equivalent one called prediction advantage in order to define pseudorandomness. Then, we will devise a complete test for pseudorandom distributions (next-bit prediction), and examine pseudorandom generators.

First, we will have to define some terms.

Definition 2 (Distribution) *X is a **distribution** over a sample space $\mathcal{S}$ if it assigns a probability p_s to the element $s \in \mathcal{S}$ such that $\sum_s p_s = 1$.*

Definition 3 (Ensemble) *A sequence $\{X_n\}_{n \in \mathbb{N}}$ is called an **ensemble** if for each $n \in \mathbb{N}$, X_n is a probability distribution over $\{0,1\}^*$.*

Generally, we will take X_n to be a distribution over $\{0,1\}^{\ell(n)}$, where $\ell(\cdot)$ is a polynomial.

Now, we will try to define computational indistinguishability. This captures what it means for distributions X, Y to "look alike" to any efficient test. In other words, no non-uniform PPT "distinguisher" algorithm D can tell X and Y apart, or the behavior of D on X and Y is the same.

We can try to think of this as a game of sorts. Let's say we give D a sample of X. Then, D gains a point if it says the sample is from X, and loses a point if it says the sample is from Y. Then we can encode D's output with one bit. In order for X and Y to be indistinguishable, D's average score on a sample of X should be basically the same as its average score on a sample of Y.

$$\mathbf{Pr}[x \leftarrow X; D(1^n, x) = 1] \approx \mathbf{Pr}[y \leftarrow Y; D(1^n, y) = 1]$$

or

$$\mathbf{Pr}[x \leftarrow X; D(1^n, x) = 1] - \mathbf{Pr}[y \leftarrow Y; D(1^n, y) = 1] \leq \mu(n)$$

for negligible $\mu(\cdot)$.

This brings us to the formal definition of **computational indistinguishability**.

Definition 4 (Computational Indistinguishability) *Two ensembles of probability distributions $X = \{X_n\}_{n\in\mathbb{N}}$ and $Y = \{Y_n\}_{n\in\mathbb{N}}$ are said to be* computationally indistinguishable *if for every non-uniform PPT distinguisher D there exists a negligible function $\nu(\cdot)$ such that*

$$|\mathbf{Pr}[x \leftarrow X; D(1^n, x) = 1] - \mathbf{Pr}[y \leftarrow Y; D(1^n, y) = 1]| \leq \nu(n).$$

We can see that this formalizes the notion of a PRG if we let X be the distribution over the PRG outputs and Y be the uniform distribution over strings of the same length as PRG outputs.

But, there is actually another model for the same idea! If we give D a sample from either X or Y, and ask it to identify which distribution it is from, if D is not right with probability better than $\frac{1}{2}$, then X and Y look the same to it! We will change notation a bit and set $X^{(1)} = X, X^{(0)} = Y$.

Definition 5 (Prediction Advantage) *Prediction Advantage is defined as*

$$\max_{\mathcal{A}} \left| \mathbf{Pr}\left[b \xleftarrow{\$} \{0,1\}, t \leftarrow X_n^b : \mathcal{A}(t) = b\right] - \frac{1}{2} \right|.$$

Proposition 1 *Prediction advantage is equivalent to computational indistinguishability.*

Proof.

$$\begin{aligned}
&\left| \mathbf{Pr}\left[b \leftarrow \{0,1\}; z \leftarrow X^{(b)}; D(1^n, z) = b\right] - \frac{1}{2} \right| \\
&= \left| \mathbf{Pr}[D(x) = 1]_{x\leftarrow X^1} \cdot \mathbf{Pr}[b = 1] + \mathbf{Pr}[D(x) = 0]_{x\leftarrow X^0} \cdot \mathbf{Pr}[b = 0] - \frac{1}{2} \right| \\
&= \frac{1}{2} \left| \mathbf{Pr}[D(x) = 1]_{x\leftarrow X^1} + \mathbf{Pr}[D(x) = 0]_{x\leftarrow X^0} - 1 \right| \\
&= \frac{1}{2} \left| \mathbf{Pr}[D(x) = 1]_{x\leftarrow X^1} - (1 - \mathbf{Pr}[D(x) = 0]_{x\leftarrow X^0}) \right| \\
&= \frac{1}{2} \left| \mathbf{Pr}[D(x) = 1]_{x\leftarrow X^1} - \mathbf{Pr}[D(x) = 1]_{x\leftarrow X^0} \right|
\end{aligned}$$

So they are equivalent within a factor of 2.

Lemma 2 (Prediction Lemma) *Let $\{X_n^{(0)}\}, \{X_n^{(1)}\}$ be ensembles of probability distributions. Let D be a non-uniform PPT adversary that $\varepsilon(\cdot)$-distinguishes $\{X_n^{(0)}\}, \{X_n^{(1)}\}$ for infinitely many $n \in \mathbb{N}$. Then $\exists$ a non-uniform PPT $\mathcal{A}$ such that*

$$\mathbf{Pr}\left[b \xleftarrow{\$} \{0,1\}, t \leftarrow X_n^b : \mathcal{A}(t) = b\right] - \frac{1}{2} \geq \frac{\varepsilon(n)}{2}$$

for infinitely many $n \in \mathbb{N}$.

Properties of Computational Indistinguishability.

1. First, we define the notation $\{X_n\} \approx_C \{Y_n\}$ to mean computational indistinguishability.

2. If we apply an efficient algorithm on X and Y, then their images under this operation are still indistinguishable. Formally, $\forall$ non-uniform PPT M, $\{X_n\} \approx_C \{Y_n\} \implies \{M(X_n)\} \approx_C \{M(Y_n)\}$. If this were not the case, then a distinguisher could simply use M to tell $\{X_n\}$ and $\{Y_n\}$ apart!

3. If X, Y are indistinguishable with advantage at most μ_1 and Y, Z are indistinguishable with advantage at most μ_2, then X, Z are indistinguishable with advantage at most $\mu_1 + \mu_2$. This follows from the triangle inequality.

This last property is actually quite nice, and we would like to generalize it a bit.

Lemma 3 (Hybrid Lemma) *Let $X^1, \ldots, X^m$ be distribution ensembles for $m = \text{poly}(n)$. Suppose D distinguishes X^1 and X^m with advantage ε. Then $\exists i \in \{1, \cdots, m-1\}$ such that D distinguishes X^i, X^{i+1} with advantage $\geq \frac{\varepsilon}{m}$.*

This follows from the pigeonhole principle.

Returning to pseudorandomness, we define a bit more notation. We call the uniform distribution over $\{0,1\}^{\ell(n)}$ by $U_{\ell(n)}$. Intuitively, a distribution is pseudorandom if it looks like a uniform distribution to any efficient test. We have the tools now to formulate this:

Definition 6 (Pseudorandom Ensembles) *An ensemble $\{X_n\}$, where X_n is a distribution over $\{0,1\}^{\ell(n)}$ is said to be pseudorandom if*

$$\{X_n\} \approx_C \{U_{\ell(n)}\}.$$

This is relevant for PRGs, as their outputs should be pseudorandom.

2.3 Next-Bit Test

Our last topic for this lecture is about an interesting way to characterize pseudorandomness. We know a pseudorandom string should pass all efficient tests a true random string would pass. For a truly random string, given a subsequence, it is not possible to predict the "next bit" with probability better than $\frac{1}{2}$ if you're given the previous bits in the subsequence. So this gives us the notion of a test using the "next bit" idea. So, we say a sequence of bits *passes the next-bit test* if no efficient adversary can predict "the next bit" in the sequence with probability better than $\frac{1}{2}$ even given all previous bits of the sequence.

Definition 7 (Next-bit Unpredictability) *An ensemble of distributions $\{X_n\}$ over $\{0,1\}^{\ell(n)}$ is next-bit unpredictable if, for all $0 \leq i < \ell(n)$ and non-uniform PPT adversaries $\mathcal{A}$, $\exists$ negligible function $\nu(\cdot)$ such that*

$$\mathbf{Pr}\big[t = t_1 \cdots t_{\ell(n)} \leftarrow X_n : \mathcal{A}(t_1 \cdots t_i) = t_{i+1}\big] \leq \frac{1}{2} + \nu(n).$$

Theorem 4 (Completeness of the Next-bit Test) *If $\{X_n\}$ is next-bit unpredictable then $\{X_n\}$ is pseudorandom.*

We will leave the proof of this last theorem until next lecture.