Lecture 4 - Computational Indistinguishability, Pseudorandom Generators

Boaz Barak

September 26, 2005

Pace, proofs Why do we need mathematical proofs?

Computational difficulty Last lecture we introduced Boolean circuits to help us model *computationally bounded* adversaries. Today we'll start using this model to get definitions (and constructions) for better encryption schemes.

Computational Definitions As a first attempt, we can say that a scheme (E, D) is T-secure if there is no T sized circuit that given $y = \mathsf{E}_k(x)$, computes x. However, as before this won't be sufficient since we need a definition that implies that it is hard to even get partial information. Thus, we will need to use a more sophisticated definition. A crucial point in that definition will be the notion of *computational indistinguishability*.

Computational Indistinguishability Recall that we defined that statistical distance of two distributions X and Y, $\Delta(X, Y)$ to equal the maximum over all sets S of

$$\big|\Pr[X \in S] - \Pr[Y \in S]\big|$$

If $\Delta(X, Y) \leq \epsilon$ we said that $X \equiv_\epsilon Y$.

An equivalent way to phrase this is that $\Delta(X, Y)$ is equal to the maximum of

$$\big|\Pr[f(X) = 1] - \Pr[f(Y) = 1]\big|$$

for all Boolean functions $f : \mathsf{Supp}(X) \cup \mathsf{Supp}(Y) \to \{0, 1\}$.

We now define X and Y to be (T, ϵ)-*computationally indistinguishable* if

$$\big|\Pr[C(X) = 1] - \Pr[C(Y) = 1]\big| \leq \epsilon$$

for all circuits C of size $\leq T$. We denote this by $X \approx_{T,\epsilon} Y$.

A special case is when Y is equal to the *uniform distribution* U_n. In this case if $X \approx_{T,\epsilon} Y$ we say that X is (T, ϵ)-*pseudorandom*.

Semantic security and indistinguishability We now finally the language to define security for encryptions in a way that will bypass our impossibility results.

Definition 1 (Semantic Security)**.** Let (E, D) be a valid encryption scheme with

$$\mathsf{E} : \underbrace{\{0,1\}^n}_{\text{key}} \times \underbrace{\{0,1\}^m}_{\text{message}} \to \underbrace{\{0,1\}^*}_{\text{ciphertext}}$$

We say that (E, D) is (T, ϵ)-*semantically secure* if for every T-sized circuit Adv, and every distribution $X \subseteq \{0,1\}^m$ of plain-text messages and every function $f : \{0,1\}^n \to \{0,1\}^*$ (with $|f(x)| \leq T$),

$$\Big| \underbrace{\Pr_{x \leftarrow_R X, k \leftarrow_R \{0,1\}^n}[\mathsf{Adv}(\mathsf{E}_k(x)) = f(x)]}_{\text{posteriori success prob.}} - \underbrace{\text{max-pr}f(X)}_{\text{apriori success prob}} \Big| < \epsilon$$

Recall that max-pr(Z) is equal to the maximum probability of a single element in the distribution Z.

Likewise, we can we can give the analog for the *indistinguishability* definition for computationally bounded adversaries:

Definition 2 (Indistinguishability)**.** Let (E, D) be a valid encryption scheme with

$$\mathsf{E} : \underbrace{\{0,1\}^n}_{\text{key}} \times \underbrace{\{0,1\}^m}_{\text{message}} \to \underbrace{\{0,1\}^*}_{\text{ciphertext}}$$

We say that (E, D) is (T, ϵ)-*indistinguishable* if for every T-sized circuit Adv, and every pair $x_1, x_2\{0,1\}^m$ of plain-text messages

$$\Pr_{i \leftarrow_R \{1,2\}, k \leftarrow_R \{0,1\}^n}[\mathsf{Adv}(\mathsf{E}_k(x_i)) = i] < \frac{1}{2} + \epsilon$$

Once again both these definitions are equivalent to one another and to the following condition: if we define $Y_x \triangleq \mathsf{E}_{U_n}(x)$ then for every x, x', $Y_x \approx_{T,\epsilon} Y_{x'}$. (Actually these equivalences hold up to some constant factors in T and ϵ but such factors won't be of significance to us.)

Constructions: Definitions don't mean much without a construction. Unfortunately, at the moment we don't know of any efficiently computable encryption scheme that satisfies this definition.

Thus we'll need to make some conjecture or axiom, which we'll then use to construct such a scheme. This is what we do now:

Definition 3. A function $g : \{0,1\}^n \to \{0,1\}^m$ is called a (T, ϵ)-*pseudorandom generator* if $g(U_n)$ is (T, ϵ)-pseudorandom.

Note that it's trivial to construct a pseudorandom generator with $m \leq n$: just use $g(x) = x$, therefore we'll require that $m > n$. $m - n$ is called the *stretch* of the pseudorandom generator.

Also note that to get a pseudorandom generator with a non-trivial stretch, we must use the *computational indistinguishability*. That is, we have the following lemma:

Lemma 1. *For every $g : \{0,1\}^n \to \{0,1\}^m$ with $m > n$, $dist(g(U_n), U_m) \geq 1/2$.*

Proof. Define S to be the image of $g(\cdot)$ (i.e. $s = \mathsf{Supp}(g(U_n))$. Note that $\Pr[g(U_n) \in S] = 1$. However $|S| \leq 2^n$ and hence $\Pr[U_m \in S] \leq 1/2$ (since $2^m \geq 2 \cdot 2^n$). □

We note that there does in fact exist a pseudorandom generator with a very large stretch and very strong computational indistinguishability:

Lemma 2. *For every (sufficiently large) n, there exist a $(2^{n/10}, 2^{-n/10})$-pseudorandom generator $g : \{0,1\}^n \to \{0,1\}^{2^{n/10}}$*

Proof. Left as exercise. □

Efficiently computable PRG's The problem is that we can't use the existence result of Lemma 2 since to actually construct usable encryption schemes we need a pseudorandom generator $g(\cdot)$ that is itself efficiently computable.

This is what we define next:

Definition 4. Let $\mathsf{G} = \{\mathsf{G}_n\}$ be a function with infinite domain where for every n, $\mathsf{G}_n : \{0,1\}^n \to \{0,1\}^{m(n)}$ with $m(n) > n$. We say that G is a *pseudorandom generator* if

- $\mathsf{G} \in \mathbf{P}$ (G is computable in polynomial-time)
- There exist functions $T, \epsilon : \mathbb{N} \to \mathbb{N}$ such that $T(n) = n^{\omega(1)}$ and $\epsilon(n) = n^{-\omega(1)}$ and such that for every n, G_n is a $(T(n), \epsilon(n))$-pseudorandom generator.

Note: We say that functions $T(\cdot), \epsilon(\cdot)$ are *super-polynomial* if they satisfy these conditions (i.e., for every large enough n and polynomial $p(\cdot)$, $T(n) > p(n)$ and $\epsilon(n) < 1/p(n)$).[1] Observe that if $T(\cdot), \epsilon(\cdot)$ are super-polynomial than so is $\frac{T^{1/10}}{n^{10}}$ and $n^{10}\epsilon^{1/10}$ (where 10 can be replaced with any other constant). So we'll treat a (T, ϵ)-PRG and a $(T^{1/c}/n^c, n^c\epsilon^{1/c})$-PRG as having essentially equivalent security.

Asymptotic vs. concrete. At the end of the day, our goal is to construct asymptotic (infinite length) PRGs, encryptions etc.. This is because such construction guarantee us that whatever security we want, we can always use a larger input length and get it (where because of the super-polynomial relationship, we'll only need to increase the input a bit to get strong security). However, throughout this course we're going to switch between the concrete and asymptotic viewpoint, always trying to use the notations that make thing more intuitive and simple.[2]

We're going to make the following assumption/axiom:

Axiom 1 (The PRG axiom)**.** *There exists a pseudorandom generator* G.

Note that this axiom implies $\mathbf{P} \neq \mathbf{NP}$ (can you see why?).

We'll see that we can do wonderful things with this axiom.

Candidates for pseudorandom generators: To justify this axiom we need at least candidate functions that we conjecture to be pseudorandom generators (even if we can't prove it). Below are two such candidate functions. We'll see more such candidates later on in the course.

[1] This is a slight abuse of notation since we should really call $\epsilon(\cdot)$ sub-polynomial, but it makes things somewhat less cumbersome.

[2] There's also a practical advantage to using the concrete notations and that is that we can derive very specific guarantees on security and so derive specific recommendations for key sizes. However, we only aim for simplicity of exposition in this course, and we won't try to derive the tightest reductions possible.

RC4 RC4 was invented by Ron Rivest for RSA. It's design is a trade secret and so the actual algorithm is not supposed to be known (security by obscurity). Nevertheless the code was obtained by reverse engineering and leaked to the cyberpunks mailing list. Even though RC4 is widely used including in the WEP and WPA protocols for wireless networks (IEEE 802.11) and the SSL protocol, several weaknesses were found in it and so it can *not* be considered a secure pseudorandom generator. (And so in fact it's not a good candidate, however I present it here since it is widely used and illustrates principles that are utilized in more secure candidates for pseudorandom generators.)

A bite is a number from 1 to 256 (or equivalently, a string in $\{0,1\}^8$). The input to the pseudorandom generator is a permutation $S : [256] \to [256]$. The output is `m` bytes (where we can control the value of `m` to be as large as we want). The following is the pseudocode for RC4:

```
i := 0
j := 0
num_outputted = 0;
while num_outputted <= m:
    i := (i + 1) mod 256
    j := (j + S[i]) mod 256
    swap(S[i],S[j])
    num_outputted := num_outputted + 1
    output S[(S[i] + S[j]) mod 256]
```

We see that RC4 expands $\log(256!)$ bits which is roughly $8 \cdot 256 = 2048$ bits into an arbitrary large `m` number of bits. However, in most current applications people desire an input much smaller than 2048 and so there's a separate pseudorandom generator (called the *key scheduling algorithm* or KSA) that takes an input of size ℓ bits, for $40 \leq \ell \leq 128$, and outputs an initial permutation S. The page `http://www.wisdom.weizmann.ac.il/~itsik/RC4/rc4.html` (written by my friend Itsik Mantin) is a good source of information on RC4 and the attacks on it.

Blum-Blum-Shub The Blum-Blum-Shub generator is even simpler than RC4 but it is much less efficient. However it has the advantage that we can relate its security to a well known problem. Assuming factoring a random n bit integer[3] cannot be done be a T-sized circuit with probability more than ϵ, this pseudorandom generator will be $(\text{poly}(T), \text{poly}(\epsilon))$-secure. The input is a number N (of length n bits) and X where $1 \leq X < N$.[4] The output will be `m` bits where again we can choose `m` to be as large as we want. The pseudocode is as follows:

```
num_outputted = 0;
while num_outputted <= m:
    X := X*X mod N
    num_outputted := num_outputted + 1
    output least-significant-bit(X)
```

[3]Random here means that we choose random primes p and q of length $n/2$ bits, where for technical reasons we require that their remainder modulu 4 is 3, and let $n = p \cdot q$.

[4]Actually x should satisfy $gcd(X, N) = 1$ but this will happen with overwhelmingly high probability for a random X.

We'll prove that a variant of this generator is as secure as factoring later in the course.

Using PRGs to construct an encryption. There is a pretty natural construction of a private key encryption with key length < message length using a pseudorandom generator.

Let G be the pseudorandom generator, with $\mathsf{G}_n : \{0,1\}^n \to \{0,1\}^m$ for $m > n$.

The encryption scheme will be the following: $\mathsf{E}_k(x) = x \oplus \mathsf{G}(k)$, $\mathsf{D}_k(y) = y \oplus \mathsf{G}(k)$. That is, we use a *pseudorandom* pad instead of a random pad in the One-Time-Pad scheme. Intuitively this should be secure since using a pseudorandom string instead of random should be good enough for all practical purposes. However, relying on intuition is very dangerous in cryptography, and so we need to verify this with a proof. Fortunately, this time the intuition holds:

Theorem 1. *Suppose that G_n is (T,ϵ)-pseudorandom. Then (E,D) is $(T/10, 10\epsilon)$-indistinguishable.*

Proof. Let $Y_x \triangleq \mathsf{E}_{U_n}(x)$ as usual. We'll prove that for every $x, x' \in \{0,1\}^m$, $Y_x \approx_{T/5,5\epsilon} Y_{x'}$. This will imply the result.

In fact we'll prove this by showing that for *every* x, Y_x is $(T/2, 2\epsilon)$-pseudorandom and hence we have that Y_x is indistinguishable from U_m which is indistinguishable from $Y_{x'}$.

That is, we prove the following claim:

Claim 1.1. *Let $\mathsf{G}_n : \{0,1\}^n \to \{0,1\}^m$ be a (T,ϵ)-pseudorandom generator. Let $x \in \{0,1\}^m$. Then the distribution $Y = \mathsf{G}_n(U_n) \oplus x$ is $(T/2, 2\epsilon)$-pseudorandom.*

Proof. The proof is by reduction. Assume, for the sake of contradiction, that there is a $T/2$ sized circuit $C : \{0,1\}^m \to \{0,1\}$ such that

$$\Big|\Pr[C(\mathsf{G}_n(U_n) \oplus x) = 1] - \Pr[C(U_m) = 1]\Big| > 2\epsilon$$

We'll construct a circuit $C' : \{0,1\}^m \to \{0,1\}$ which will contradict the security of G_n. We define C' in the following way: $C'(y) \triangleq C(y \oplus x)$.

Now, the size of C' is roughly the size of C plus m (the number of bits it takes to describe y). Since we always assume that T is much larger than the input length (e.g., $T > 100m$), we get that the size of C' is at most T.

Now $\Pr[C'(\mathsf{G}_n(U_n)) = 1]$ is by definition equal to $\Pr[C(\mathsf{G}_n(U_n) \oplus x) = 1]$.

On the other hand $\Pr[C'(U_m) = 1] = \Pr[C(U_m \oplus x) = 1]$. However, for every fixed x, the distribution $U_m \oplus x$ is also equal to the uniform distribution! Thus,

$$\Pr[C(U_m \oplus x) = 1] = \Pr[C(U_m) = 1]$$

(Note that the two occurrences of U_m in the above equation refer to different (i.e., independent) copies of the uniform distribution.)

Thus

$$\Big|\Pr[C'(\mathsf{G}_n(U_n)) = 1] - \Pr[C'(U_m) = 1]\Big| > 2\epsilon$$

contradicting the security of G_n. □

Note: To complete the proof, we need to show that like statisticial indistinguishable, computational indistinguishability is *transitive*. That is, we need to show the following claim:

Claim 1.2 (Transitivity of Computational Indistinguishability)**.** *Let X, Y, Z be distributions such that $X \approx_{T,\epsilon} Y$ and $Y \approx_{T,\epsilon'} Z$. Then $X \approx_{T,\epsilon+\epsilon'} Z$*

Before proving it let me explain why we need to use it: We want to prove that for every x, x' it holds that Y_x is indistinguishable from $Y_{x'}$ but what we proved is that for every x, Y_x is indistinguishable from U_n. However, the transitivity will imply that if Y_x and $Y_{x'}$ are both indistinguishable from U_m then they are also indistinguishable from each other.

Proof. Suppose for the sake of contradiction that there's a T-sized circuit C with $|\Pr[C(X) = 1] - \Pr[C(Z) = 1]| > \epsilon + \epsilon'$. Let's use the shorthand $p_C(X)$ for $\Pr[C(X) = 1]$ (and similarly define $p_C(Y)$, $p_C(Z)$. Note that

$$p_C(X) - p_C(Z) = p_C(X) - p_C(Y) + p_C(Y) - p_C(Z) = \Big(p_C(X) - p_C(Y)\Big) - \Big(p_C(Y) - p_C(Z)\Big)$$

The *triangle inequality* says that for every three real numbers $s, t, u \in \mathbb{R}$, $|s - u| \leq |s - t| + |t - u|$. This means that

$$\epsilon + \epsilon' < |p_C(X) - p_C(Z)| \leq |p_C(X) - p_C(Y)| + |p_C(Y) - p_C(Z)|$$

but this is impossible since $X \approx_{T,\epsilon} Y$ and hence $|p_C(X) - p_C(Y)| \leq \epsilon$ and $Y \approx_{T,\epsilon'} Z$ and hence $|p_C(Y) - p_C(Z)| \leq \epsilon'$.

□

The geometrical viewpoint: Note that it is not accidental that we used the triangle inequality in the proof. We can view $|s - u|$ as the distance between s and u. What the triangle inequality says that the distance between s and u is at most the distance between s and t plus the distance between t and u. Transitivity of computational indistinguishability is also similar: let $\Delta_T(X, Y)$ be the maximum over T-sized circuits C of $|C(X) - C(Y)|$. We can think of $\Delta_T(X, Y)$ as some kind of distance function measuring the distance between distributions. The transitivity lemma says that this function satisfies the triangle inequality, and so it is no coincidence that we use triangle inequality over the reals to prove it.

□

Increasing the output length Our the PRG axiom only guaranteed us a pseudorandom generator with output m larger than n. As far as we know, it may be that $m = n + 1$. It seems to be a lot of trouble to get into for reducing the key size by only one bit!

Fortunately, it turns out we can use a PRG with $m = n + 1$ to construct a PRG with an arbitrary polynomial stretch.

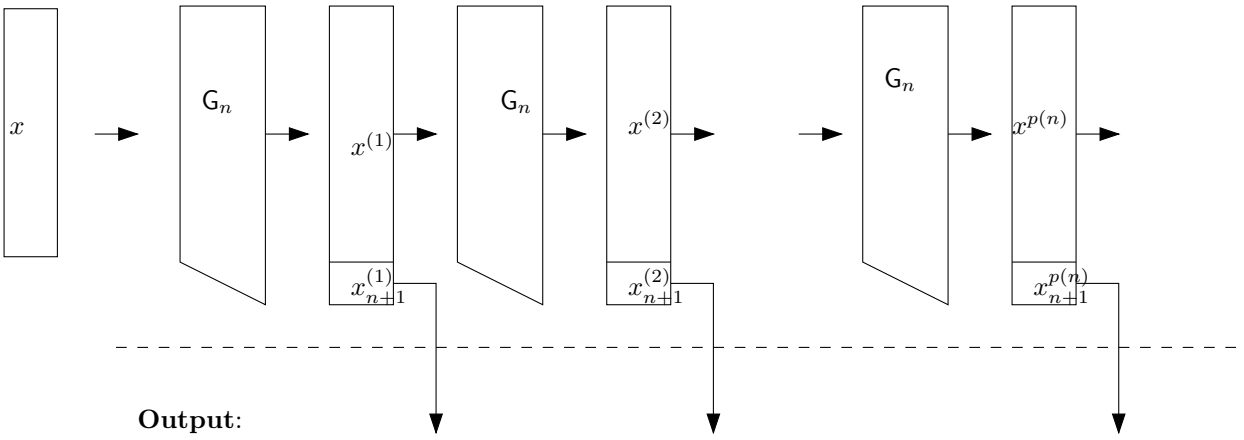

Figure 1: Extending output of pseudorandom genertator

Theorem 2. *Assume that there exists a PRG. Then for every polynomial $p(\cdot)$, there exists a PRG $\mathsf{G} = \{\mathsf{G}_n\}$ with $\mathsf{G}_n : \{0,1\}^n \to \{0,1\}^{p(n)}$.*

Proof. Assume that we have a PRG $PRG' = \{\mathsf{G}'_n\}$ with $\mathsf{G}'_n : \{0,1\}^n \to \{0,1\}^{n+1}$. Let $p(\cdot)$ be a polynomial. We'll construct a PRG $\mathsf{G} = \{\mathsf{G}_n\}$ with $\mathsf{G}_n : \{0,1\}^n \to \{0,1\}^{p(n)}$ with running-time(G) equal to roughly $p(n)$ times the running time of G'.

The algorithm for G_n will be as follows: (notation: for a string $x \in \{0,1\}^m$, and $i < j \leq m$, $x_{[i...j]}$ is $x_i x_{i+1} \cdots x_j$)

Input: $x \in \{0,1\}^n$.

$i \leftarrow 0$
$x^{(0)} \leftarrow x$
`while` $i \leq p(n)$`:`
    $i \leftarrow i + 1$
    $x^{(i)} \leftarrow \mathsf{G}_n(x^{(i-1)}_{[1...n]})$
    `output` $x^{(i)}_{n+1}$

Useful property. We're going to make use of the following property. The *statistical distance* satisfies the following property: If $\Delta(X,Y) \leq \epsilon$ then for every function $f(\cdot)$, $\Delta(f(X), f(Y)) \leq \epsilon$. It turns out that computational indistinguishability satisfies a similar property, as long as $f(\cdot)$ is efficiently computable.

Claim 2.1 (Functions of indistinguishable distributions.)**.** *Let X, Y over $\{0,1\}^m$ such that $X \approx_{T,\epsilon} Y$ and let $f : \{0,1\}^m \to \{0,1\}^{m'}$ be a function computable by a t circuit (for $t < T$). Then, $f(X) \approx_{T-t-100,\epsilon} f(Y)$.*

(The proof of the claim is left as an exercise.)

Let $m = p(n)$. We define now the following random variables $Y^{(0)}, \ldots, Y^{(m)}$. The variable $Y^{(i)}$ will range over $\{0,1\}^{n+i}$ and will reflect the state of our pseudorandom generator at the i^{th} step. That is, $Y^{(0)} \triangleq U_n$, $Y^{(1)} \triangleq \mathsf{G}_n(U_n)$, $Y^{(i+1)} = \mathsf{G}_n(Y^{(i)}_{[1\ldots n]})Y^{(i)}_{[n+1\ldots n+1]}$.

We'll prove the following claim:

Claim 2.2. *Let t denote the running time of G_n on length-n inputs (note that t is polynomial). For every $1 \leq i \leq m$,*

$$Y^{(i)} \approx_{T-2mti,2i\epsilon} U_{n+i}$$

Before proving Claim 2.2, note that it does indeed imply that our generator's output is pseudorandom. Firstly, note that if T and ϵ are super-polynomial then so is $T - tm^2$ and $2m\epsilon$. Now the output of our pseudorandom generator is simply the last m bits of $Y^{(m)}$ and so if $Y^{(m)}$ is pseudorandom then so is this output.

Proof of Claim 2.2. We prove this by induction. $Y^{(0)}$ is simply equal to U_n so there's nothing to prove in that case. For $Y^{(1)} = \mathsf{G}_n(Y^{(0)})$ the claim follows from the security of G_n. Thus, let $i \geq 1$ and assume that $Y^{(i)} \approx_{T-2ti,2i\epsilon} U_{n+i}$ and we'll prove this for $Y^{(i+1)}$.

Consider the function $f : \{0,1\}^{n+i} \to \{0,1\}^{n+i+1}$ defined as follows: $f(y) = \mathsf{G}_n(y_{[1\ldots n]})y_{[n+1\ldots n+i]}$. That is, $f(Y^{(i)}) = Y^{(i+1)}$. Note that $f(\cdot)$ is computable in $2t$ time (assume $t \geq m$ for convenience). We claim that $f(U_{n+i}) \approx_{T-m,\epsilon} U_{n+i+1}$. Indeed, any $T-m$ sized distinguisher between these two distribution can be turned (by hardwiring the last m bits) into a T sized distinguisher for G_n.

Now by Claim 2.1, this implies that if $Y^{(i)} \approx_{T-2mti,2i\epsilon} U_{n+i}$ then $f(Y^{(i)} \approx_{T-2mti-2t,2i\epsilon} f(U_{n+1})$. By transitivity (Claim 1.2), we get that

$$f(Y^{(i)}) \approx_{T-2mti-2t,2i\epsilon+\epsilon} U_{n+i+1}$$

which implies

$$f(Y^{(i)}) \approx_{T-2mt(i+1),(2i+1)\epsilon} U_{n+i+1}$$

□

□

Note: This proof technique — proving that two distributions X and Y are indistinguishable by presenting *intermediate* distributions $X^{(0)}, \ldots, X^{(m)}$ with $X^{(0)} = X$ and $X^{(m)} = Y$ and the showing that $X^{(i)}$ is indistinguishable from $X^{(i+1)}$ — is called the *hybrid* technique, and is a very important technique in cryptographic proofs. I recommend that you also review the description of the same theorem and proof in Goldreich's book (see course web site for link).