A Mini-Course on the Theory of Cryptography

Charles Rackoff

Department of Computer Science, University of Toronto

Bonn, June 17-21, 2006

(Part 1)

TOPICS: PRIMITIVES and PROTOCOLS

Cryptographic Primitives

- Pseudo-random Generators (of various sorts)
- One-way Functions
- Shared-Private-key signature schemes (MACs)
- Public-key signature schemes
- Distribution-smoothing families of hash functions (cryptography not required!)
- Collision-resistant families of hash functions
- Public-key encryption schemes

Cryptographic Protocols

- Secure session, assuming a shared session key
- Sharing a session key, assuming various kinds of credentials

CAVEATS (or rather apologies)

Our notions of security will always be respect to a well-defined notion of adversary. Our adversaries will usually be constrained to be efficient in some asymptotic sense, and for this I do not apologize. However, our adversaries will always have as their inputs bits computed by good guys and *nothing else*. In particular, they will not have access to:

CAVEATS (or rather apologies)

Our notions of security will always be respect to a well-defined notion of adversary. Our adversaries will usually be constrained to be efficient in some asymptotic sense, and for this I do not apologize. However, our adversaries will always have as their inputs bits computed by good guys and *nothing else*. In particular, they will not have access to:

- Electromagnetic radiation coming from computers, or amount of power used. (No apology.)

CAVEATS (or rather apologies)

Our notions of security will always be respect to a well-defined notion of adversary. Our adversaries will usually be constrained to be efficient in some asymptotic sense, and for this I do not apologize. However, our adversaries will always have as their inputs bits computed by good guys and *nothing else*. In particular, they will not have access to:

- Electromagnetic radiation coming from computers, or amount of power used. (No apology.)
- Sounds of people typing, and of their hard drives. (No apology)

CAVEATS (or rather apologies)

Our notions of security will always be respect to a well-defined notion of adversary. Our adversaries will usually be constrained to be efficient in some asymptotic sense, and for this I do not apologize. However, our adversaries will always have as their inputs bits computed by good guys and *nothing else*. In particular, they will not have access to:

- Electromagnetic radiation coming from computers, or amount of power used. (No apology.)
- Sounds of people typing, and of their hard drives. (No apology)
- Amount of computing time spent by the good guys.
  Big apology!

CAVEATS (or rather apologies)

Our notions of security will always be respect to a well-defined notion of adversary. Our adversaries will usually be constrained to be efficient in some asymptotic sense, and for this I do not apologize. However, our adversaries will always have as their inputs bits computed by good guys and *nothing else*. In particular, they will not have access to:

- Electromagnetic radiation coming from computers, or amount of power used. (No apology.)
- Sounds of people typing, and of their hard drives. (No apology)
- Amount of computing time spent by the good guys.
  Big apology!
- Analogue content of the signal the adversary is reading.
  Apology???

Nicest Theorem of Cryptography

The following are all equivalent:

- There exist one-way functions.
- There exist one kind of pseudo-random generator.
- There exist another kind of pseudo-random generator.
- There exist secure shared-private-key signature schemes (MACs).
- There exist secure public-key signature schemes.
- There exist weakly collision-resistant families of hash functions

Cryptography can be viewed as the study of average case complexity theory, from a very natural point of view.

To motivate pseudo-random generators,
consider "the" cryptography problem:

A and B share a secret random key $k = k_1 k_2 \ldots k_n$.
A wishes to "secretly" send to B the message $m = m_1 m_2 \ldots m_\ell$;
there is an eavesdropping adversary E on the channel, and we don't want E to learn anything about the message.

Famous Solution: Use k as a *one-time pad*:

A sends over the channel $c_1 = m_1 \oplus k_1, c_2 = m_2 \oplus k_2, \ldots$
B computes $m_1 = c_1 \oplus k_1, m_2 = c_2 \oplus k_2, \ldots$.

Intuitively this yields *perfect* security, but it is only defined if the message is no longer than the shared key.

Suggestion: Both A and B use a pseudo-random number generator to "stretch" the key k to a much longer key k', and then use k' as in a one-time pad.

What definition of security do we want here?

What definition of security do we want here?

Bad Idea: E, seeing the encryption of a random ℓ-bit string m under a random key, should not be able to guess m except with negligible probability. (**not strong enough**)

What definition of security do we want here?

Bad Idea: E, seeing the encryption of a random ℓ-bit string m under a random key, should not be able to guess m except with negligible probability. (**not strong enough**)

Good Idea: E chooses two $\ell - bit$ strings m^0 and m^1, and sees the encryption of a random one of them under a random key; he should not be able to guess which one was encrypted, with probability significantly above 1/2.

How should we parse:
"pseudo random number generator"?

(Keep in mind that "number" means "string".)

How should we parse:
"pseudo random number generator"?

(Keep in mind that "number" means "string".)

- [pseudo-random number] generator?

How should we parse: "pseudo random number generator"?

(Keep in mind that "number" means "string".)

- [pseudo-random number] generator?
  NO! There is no notion of "pseudo-random number" that is useful in cryptography.

How should we parse: "pseudo random number generator"?

(Keep in mind that "number" means "string".)

- [pseudo-random number] generator?
  NO! There is no notion of "pseudo-random number" that is useful in cryptography.
- [pseudo-random] [number generator]
  is the parsing we use.

How should we parse: "pseudo random number generator"?

(Keep in mind that "number" means "string".)

- [pseudo-random number] generator?
  NO! There is no notion of "pseudo-random number" that is useful in cryptography.
- [pseudo-random] [number generator]
  is the parsing we use.

An adversary should not be effectively able to tell the difference between a string generated from a random key, and a truly randomly generated string.

"Effectively" means: the adversary has to run in feasible time, and his ability to tell the difference should be very small.

Definitions:

A *number generator* is a polynomial time computable function $G : \{0,1\}^* \to \{0,1\}^*$, such that $|G(s)| = \ell(|s|) > |s|$ for some function ℓ and every string s. (Assume ℓ is one-one.)

The number generator G is *pseudo-random* if the following holds for every *distinguisher* D:
(This is the *uniform adversary* version of the definition.)

Let D be a probabilistic, polynomial time algorithm.
For each $n \in \mathbb{N}$, define

$p_D(n) =$ the probability that if s is randomly chosen from $\{0,1\}^n$ and D is run on $G(s) \in \{0,1\}^{\ell(n)}$, then D accepts;

$r_D(n) =$ the probability that if α is randomly chosen from $\{0,1\}^{\ell(n)}$, and D is run on α, then D accepts.

THEN for every c and sufficiently large n, $|p_D(n) - r_D(n)| \leq \frac{1}{n^c}$.

(We may omit the subscript D if it is understood.)

There is also a *nonuniform adversary* version of this definition: In this version, D is a polynomial-size family $\{D_1, D_2, \ldots\}$ of (deterministic) *circuits*, D_n having $\ell(n)$ input bits and one output bit.

It is easy to see that this latter definition would not change if we permitted the circuits to be probabilistic. This is because for each such circuit there would be a way to fix its random input to an optimum value. Such a value might be hard to find, but it would exist.

It is therefore easy to see that
"pseudo-random against nonuniform adversaries" implies
"pseudo-random against uniform adversaries".
but the converse is believed not to be true.

Every definition of security in this course will have a "uniform adversary" version and a "nonuniform adversary" version; even if we state just one version, the other version will be self-evident. All the theorems we will state in this course of the form "if this thingy is secure then that thingy is secure" will be true, assuming we *consistently* use one of the two versions. In practice, it is usually slightly easier to define security and to prove theorems in the *nonuniform adversary* setting; in at least one case, it is *much* easier.

Note that it is never necessary to allow our nonuniform circuits to use randomness. However, it is occasionally useful.

Notation: The (nonnegative) function $f(n)$ is *negligible* if for all c and all sufficiently large n, $f(n) \leq 1/n^c$. f is *significant* if it is not negligible.

The intuition of pseudo-randomness is that G should pass *all* efficient statistical tests. This definition is very robust.

Equivalent definition: G is pseudo-random if for every adversary D, if we give D both:
$G(s)$ for a random $s \in \{0,1\}^n$, and
a randomly chosen $\alpha \in \{0,1\}^{\ell(n)}$,
in a random order,
then the probability he can guess which was which is negligibly greater than $1/2$.

What about other variants?

The intuition of pseudo-randomness is that G should pass *all* efficient statistical tests. This definition is very robust.

Equivalent definition: G is pseudo-random if for every adversary D, if we give D both:
$G(s)$ for a random $s \in \{0,1\}^n$, and
a randomly chosen $\alpha \in \{0,1\}^{\ell(n)}$,
in a random order,
then the probability he can guess which was which is negligibly greater than $1/2$.

What about other variants?
What if the distinguisher gets to see *many* samples?

Definition: We say G is *pseudo-random against multiple sampling* if for every D (probabilistic poly time, or poly size):

For each n, define

$p_D(n)$ = the probability that if $s_1, s_2, \ldots, s_n$ are randomly (and independently) chosen from $\{0,1\}^n$ and D is run on $[G(s_1)G(s_2)\ldots G(s_n)]$, then D accepts;

$r_D(n)$ = the probability that if α is randomly chosen from $\{0,1\}^{n\cdot\ell(n)}$ and D is run on α, then D accepts.

THEN for every c and sufficiently large n, $|p_D(n) - r_D(n)| \leq \frac{1}{n^c}$.

Theorem: Let G be a number generator.
G is pseudo-random $\Longleftrightarrow$ G is pseudo-random against multiple sampling.

Proof of $\Longleftarrow$: Obvious.

Proof of $\Longrightarrow$: Assume that G is *not* pseudo-random against multiple sampling. We will show that G is not pseudo-random.

Let D be an adversary for distinguishing G using multiple sampling. For each n define $p_D(n)$ and $r_D(n)$ as in the definition of "pseudo-random against multiple sampling". Fix n and say (without loss of generality) that $p_D(n) - r_D(n) > \frac{1}{n^c}$. We will describe a circuit D'_n for distinguishing G.

We first describe a sequence of experiments that are "hybrids" between the experiment that gives rise to $p_D(n)$ and the experiment that gives rise to $r_D(n)$.

For $0 \leq i \leq n$ let q_i be the probability, IF $s_1, s_2, \ldots, s_i$ are i randomly chosen strings from $\{0,1\}^n$, and $t_{i+1}, t_{i+2}, \ldots, t_n$ are $n-i$ randomly chosen strings from $\{0,1\}^{\ell(n)}$, and D_n is given as input $[G(s_1)G(s_2)\ldots G(s_i)t_{i+1}t_{i+2}\ldots t_n]$, THEN D accepts. Clearly $q_n = p_D(n)$ and $q_0 = r_D(n)$.

So there exists an i, $0 \leq i < n$, such that $q_{i+1} - q_i > \frac{1}{n^{c+1}}$; fix such an i.

We now describe a *probabilistic* D' for distinguishing G. The input will be an $\ell(n)$ bit string α. D' will choose i strings $s_1, s_2, \cdots, s_i$ randomly from $\{0,1\}^n$ and $n-(i+1)$ strings $t_{i+2}, t_{i+3}, \cdots, t_n$ randomly from $\{0,1\}^{\ell(n)}$, and run D on $[G(s_1)G(s_2)\cdots G(s_i)\,\alpha\, t_{i+2}t_{i+3}\cdots t_n]$.

Let $p_{D'}(n)$ be the probability D' accepts when $\alpha = G(s)$ for s randomly chosen from $\{0,1\}^n$; clearly $p_{D'}(n) = q_{i+1}$.

Let $r_{D'}(n)$ be the probability D' accepts when α is randomly chosen from $\{0,1\}^{\ell(n)}$; clearly $r_{D'}(n) = q_i$.

So $p_{D'}(n) - r_{D'}(n) > \frac{1}{n^{c+1}}$.

(Minor Problems: In the uniform adversary setting, we have to say how D chooses i. In the nonuniform setting, we have to make D deterministic.)

Predictability

Informally, we say that number generator G is *unpredictable* (from the left) if given a proper prefix of $G(s)$, one cannot guess the next bit with probability significantly above $1/2$.

Definition: G is *unpredictable* (from the left) if the following holds for every (prob. polytime, or poly size) A:

A has, firstly, input 1^n, and A outputs i, $1 \leq i \leq \ell(n)$.

Then A sees an $i-1$-bit string α and outputs a bit.

Define, $pred_A(n)$ = the probability that if s is randomly chosen from $\{0,1\}^n$ and $G(s) = [b_1, b_2, \cdots, b_{\ell(n)}]$ and A_n is given $[b_1, b_2, \cdots, b_{i-1}]$, then A_n outputs b_i.

THEN for every c and sufficiently large n, $pred_A(n) \leq \frac{1}{2} + \frac{1}{n^c}$.

Theorem: G is pseudo-random $\iff$ G is unpredictable.

Proof of $\Longrightarrow$: Obvious, since predictability yields a statistical test.

Proof of $\Longleftarrow$: Let D be an adversary for distinguishing G, and let n be such that $p_D(n) - r_D(n) > 1/n^c$.

We will describe an adversary A for predicting G.

We first describe a sequence of experiments that are "hybrids" between the experiment that gives rise to $p_D(n)$ and the experiment that gives rise to $r_D(n)$.
For $0 \leq i \leq \ell(n)$ let p_i be the probability:
IF s is randomly chosen from $\{0,1\}^n$ and $G(s) = [b_1, b_2, \cdots, b_{\ell(n)}]$, and $a_{i+1}, a_{i+2}, \cdots, a_{\ell(n)}$ are $\ell(n) - i$ randomly chosen bits, and D is given as input $[b_1, b_2, \cdots, b_i, a_{i+1}, a_{i+2}, \cdots, a_{\ell(n)}]$, THEN D_n accepts.

Clearly $p_{\ell(n)} = p(n)$ and $p_0 = r(n)$, so $p_{\ell(n)} - p_0 > 1/n^c$.

So choose i, $0 < i \leq \ell(n)$, such that $p_i - p_{i-1} > 1/(\ell(n)n^c)$.

We now describe a *probabilistic* A for predicting bit i of the output of G.

The input to A will be an $i-1$ bit string α. A will choose a random bit a and $\ell(n) - i$ random bits $a_{i+1}, a_{i+2}, \cdots, a_{\ell(n)}$ and run D on $[\alpha a, a_{i+1}, a_{i+2}, \cdots, a_{\ell(n)}]$.

If D accepts then A outputs bit a, otherwise A outputs bit $1 - a$.

We compute: $pred_A(n) > \frac{1}{2} + \frac{1}{\ell(n)n^c}$

Remark: We can also define a notion of G being "unpredictable from the right". Because our notion of pseudo-random is symmetric with respect to left and right, we easily get the theorem that G is pseudo-random if and only if G is unpredictable from the right. As a corollary, we therefore get that ***G* is unpredictable from the left if and only if *G* is unpredictable from the right.**