A Mini-Course on the Theory of Cryptography

Charles Rackoff

Department of Computer Science, University of Toronto

Bonn, June 17-21, 2006

(Part 3)

Review

We showed that pseudo-random number generators exist
if and only if
pseudo-random function generators exist
if and only if
pseudo-random function generators with unbounded input exist.

How we construct an (instance of) a (hopefully) pseudo-random permutation generator using the engine of composition

1. Start with a very simple basic permutation generator that on key k of length m generates permutation $F_k : \{0,1\}^n \to \{0,1\}^n$. F should do "enough" mixing, but not give us any real security. F_k will usually have the feature that it is easy to invert given k.

2. Define $F'_{k_1 k_2 \ldots k_r}(x) = F_{k_1} \circ F_{k_2} \circ \ldots \circ F_{k_r}$.
   That is, we compose F for r rounds.

3. We don't want our key to be too long, so we use a (very simple)round-key generation function
   $keygen : \{0,1\}^l \to \{0,1\}^{r \cdot m}$;
   for key K of length l, we finally define:
   $F''_K(x) = F'_{keygen(K)} = F'_{k_1 k_2 \ldots k_r}$ where $keygen(K) = k_1 k_2 \ldots k_r$.

Note: We might be in trouble composing these functions if they weren't permutations!

Example 1: DES (1976)

$n = 64$, that is, we generate permutations that map $\{0,1\}^{64} \to \{0,1\}^{64}$. The key size for the basic generator is 48, and there are 16 rounds of composition. The final key size is 56, so the key generation function maps 56 bits to $16 \cdot 48 = 768$ bits.

The basic generator F works as follows:
$F_k(LR) = [R,\, L \oplus G_k(R)]$ where $|L| = |R| = 32$.
(We call this a Feistel permutation.) G is some simple, randomish function (not permutation) generator, $G_k : \{0,1\}^{32} \to \{0,1\}^{32}$.

Attacks: "Differential Cryptanalysis" and "Linear Cryptanalysis" (and variants) have been able to improve the brute force attack of 2^{56} evaluations to about 2^{40} evaluations.

Example 2: AES (2001)

$n = 128$, that is, we generate permutations that map $\{0,1\}^{128} \to \{0,1\}^{128}$. The key size for the basic generator is 128, and there are 10, 12, or 14 rounds of composition. The final key size is 128, 192, or 256, so the key generation function maps 128 bits to $10 \cdot 128$, $12 \cdot 128$, or $14 \cdot 128$ bits.

The basic generator F creates a permutation (that is easy to invert given the key) using algebraic methods rather than Feistel, and therefore more "mixing" is done each round than with DES. This may be a good thing, or it may be bad. (For example, if the basic F generates "linear" functions then their composition is also linear and therefore trivial to break.)

At the moment, no attack faster than brute force is known.

Remarks: Timing attacks (involving the cache) have been successful against standard implementations of AES. This is interesting, but there is no reason to believe that AES *should* have been secure in this way.

Our definition of security (pseudo-randomness) means only that, and no other assumptions are justified. For example, some people think that evaluating a function generator on something involving its own key is a good idea. For example, define:
$\mathrm{AES}'_K(x) = \mathrm{AES}_K(x \oplus K)$.

There is no reason to believe that AES' is pseudo random. In fact, this transformation is provably bad, in the following sense: if any pseudo-random function generator exists, then there is one whose transformation in this way is *not* pseudo-random.

Towards a Theory of Composition-based Function Generators

Under what circumstances can we conclude that if we start with a "basic" function generator F, and then compose it with itself for r rounds (using independently random round-keys), that the result is pseudo-random?

Suggestion: Start with a "simple" F. For example, we might insist that for each function generated by F, each output bit only depends on a finite (or $O(\log n)$) input bits. Then compose F for sufficiently many rounds to achieve some combinatorial property. For example, we may compose F for r rounds so that we have "approximate 4-way independence". (Note that approximate 4-way, and even 1-way, independence is neither necessary nor sufficient for pseudo-randomness.)

Conjecture: If F and r are as above, then the composition of $(k+1)r$ rounds satisfies approximately $(4+k)$-way independence.

Back to "Secure Sessions"

Our motivation for pseudo-random generators was to have a way for two parties who share a random session key to communicate securely over an insecure channel.

Before we only assumed the channel was insecure in the sense that the adversary could listen in. Now we want to assume that that the adversary has *complete control* over the channel: he can read, remove from, add to, or change in any way the bits going over the channel.

For simplicity, we assume that A receives message bits (or blocks) that he wishes to communicate to B.

(If B wishes to speak to A at the same time, we can use the same protocol, with a second session key. **If**, that is, we have a good definition of security!)

Setting: We have a security parameter n, and the length of the shared session key k will depend (polynomially) on n. (Of course, we can always use a key of length n to pseudo-randomly generate a longer key.)

A sees the message come in, in *pieces*: $m_0, m_1, m_2, \ldots$. The piece size can be constant – for example, 1 – or it can depend on n; for concreteness, let's assume the piece size is n.

A uses k to (perhaps probabilistically) encrypt one piece at a time, creating strings $e_0, e_1, e_2, \ldots$ of length, say, $\ell(n)$ that are sent to B over the insecure channel.

B reads one encrypted piece at a time and uses k to decrypt each piece, outputting $m'_0, m'_1, m'_2, \ldots$.

Both A and B are allowed to have a polynomial (in n) amount of memory (in addition to the shared key k).

We will only consider *correct* systems, that is, systems where, in the absence of an adversary, B correctly decrypts $m_0, m_1, m_2, \ldots$.

Informally, *security* will mean that the adversary can't learn anything about the $\{m_i\}$ (i.e., *Privacy*), and can't trick B into outputting something wrong (i.e., *Integrity*).

Here are some examples of correct systems. We are interested in whether or not they are secure.

For each system, we will assume the existence of a suitable pseudo-random function generator F.

Cryptosystem I. The shared key is k, $|k| = n$.
F is a pseudo-random *permutation* generator,
$F_k : \{0,1\}^n \to \{0,1\}^n$.

For n-bit message pieces $m_0, m_1, m_2 \ldots$,
A encrypts each m_i by $e_i = F_k(m_i)$.

B decrypts in the obvious way: Given encryptions $e'_0, e'_1, e'_2, \ldots$,
B computes $m'_i = F_k^{-1}(e'_i)$.

Is this secure?

Cryptosystem I. The shared key is k, $|k| = n$.
F is a pseudo-random *permutation* generator,
$F_k : \{0,1\}^n \to \{0,1\}^n$.

For n-bit message pieces $m_0, m_1, m_2 \ldots$,
A encrypts each m_i by $e_i = F_k(m_i)$.

B decrypts in the obvious way: Given encryptions $e'_0, e'_1, e'_2, \ldots$,
B computes $m'_i = F_k^{-1}(e'_i)$.

Is this secure? **NO!**

Cryptosystem I. The shared key is k, $|k| = n$.
F is a pseudo-random *permutation* generator,
$F_k : \{0,1\}^n \to \{0,1\}^n$.

For n-bit message pieces $m_0, m_1, m_2 \ldots$,
A encrypts each m_i by $e_i = F_k(m_i)$.

B decrypts in the obvious way: Given encryptions $e'_0, e'_1, e'_2, \ldots$,
B computes $m'_i = F_k^{-1}(e'_i)$.

Is this secure? **NO!**

- It does not satisfy *Privacy*: Adversary can tell which pieces are equal to which other pieces. (Anecdote: Chosen Plaintext Attack and the Battle of Midway Island.)

Cryptosystem I. The shared key is k, $|k| = n$.
F is a pseudo-random *permutation* generator,
$F_k : \{0,1\}^n \to \{0,1\}^n$.

For n-bit message pieces $m_0, m_1, m_2 \ldots$,
A encrypts each m_i by $e_i = F_k(m_i)$.

B decrypts in the obvious way: Given encryptions $e'_0, e'_1, e'_2, \ldots$,
B computes $m'_i = F_k^{-1}(e'_i)$.

Is this secure? **NO!**

- It does not satisfy *Privacy*: Adversary can tell which pieces are equal to which other pieces. (Anecdote: Chosen Plaintext Attack and the Battle of Midway Island.)
- It does not satisfy *Integrity*: Adversary can send anything he wants to B, and B will happily decrypt it. (Adversary can even trick B into reordering the pieces of the message.)

Cryptosystem II. The shared key is k, $|k| = n$.
F is a pseudo-random function generator, $F_k : \{0,1\}^n \to \{0,1\}^n$.

For n-bit message pieces $m_0, m_1, m_2 \ldots$,
A encrypts each m_i by $e_i = m_i \oplus F_k(\bar{i})$.

B decrypts in the obvious way: Given encryptions $e'_0, e'_1, e'_2, \ldots$,
B computes $m'_i = e'_i \oplus F_k(\bar{i})$.

Is this secure?

Cryptosystem II. The shared key is k, $|k| = n$.
F is a pseudo-random function generator, $F_k : \{0,1\}^n \to \{0,1\}^n$.

For n-bit message pieces $m_0, m_1, m_2 \ldots$,
A encrypts each m_i by $e_i = m_i \oplus F_k(\bar{i})$ and sends $e_0, e_1, e_2 \ldots$, to B.

B decrypts in the obvious way: Given encryptions $e'_0, e'_1, e'_2, \ldots$,
B computes $m'_i = e'_i \oplus F_k(\bar{i})$.

Is this secure? **NO!**

Cryptosystem II. The shared key is k, $|k| = n$.
F is a pseudo-random function generator, $F_k : \{0,1\}^n \to \{0,1\}^n$.

For n-bit message pieces $m_0, m_1, m_2 \ldots$,
A encrypts each m_i by $e_i = m_i \oplus F_k(\bar{i})$.

B decrypts in the obvious way: Given encryptions $e'_0, e'_1, e'_2, \ldots$,
B computes $m'_i = e'_i \oplus F_k(\bar{i})$.

Is this secure? **NO!**

- It *does* satisfy *Privacy*: No matter how the next piece is chosen, its encryption will look completely random

Cryptosystem II. The shared key is k, $|k| = n$.
F is a pseudo-random function generator, $F_k : \{0,1\}^n \to \{0,1\}^n$.

For n-bit message pieces $m_0, m_1, m_2 \ldots$,
A encrypts each m_i by $e_i = m_i \oplus F_k(\bar{i})$.

B decrypts in the obvious way: Given encryptions $e'_0, e'_1, e'_2, \ldots$,
B computes $m'_i = e'_i \oplus F_k(\bar{i})$.

Is this secure? **NO!**

- It *does* satisfy *Privacy*: No matter how the next piece is chosen, its encryption will look completely random
- It does *not* satisfy *Integrity*: Adversary can send anything he wants to B, and B will happily decrypt it.

Cryptosystem III. The shared key is (k_1, k_2), $|k_1| = |k_2| = n$.
F is a pseudo-random function generator, $F_k : \{0,1\}^n \to \{0,1\}^n$.
G is a pseudo-random function generator, $G_k : \{0,1\}^{2n} \to \{0,1\}^n$.

For n-bit message pieces $m_0, m_1, m_2 \ldots$,
A encrypts each m_i by computing $\alpha_i = m_i \oplus F_{k_1}(\bar{i})$ and
$e_i = [\alpha_i, G_{k_2}(\bar{i}\,\alpha_i)]$.

B decrypts: Given $e'_0, e'_1, e'_2, \ldots$, each $2n$ bits long, decrypt piece i as follows:
Say $e'_i = [\alpha'_i, \beta'_i]$. If $\beta'_i = G_{k_2}(\bar{i}\,\alpha'_i)$, then let $m'_i = \alpha'_i \oplus F_{k_1}(\bar{i})$;
else let $m'_i =$ FAIL, and in fact let $m'_j =$ fail for all $j \geq i$. FAIL is a special symbol indicating failure. We will insist that whenever B outputs FAIL for one piece, he does so for every subsequent piece he decrypts.

Is this secure? **YES!**

Definition of *Integrity* for such a cryptosystem:
For every polynomial bounded D:

D chooses m_0, sees e_0, chooses m_1, sees e_1, etc., for a polynomial number s of steps;

D then chooses a string γ of length ns and gives it to B, who then outputs $m'_0, m'_1, \ldots, m'_{s-1}$.

Let $P_D(n)$ = the probability that for some $i < s$,
$m'_i \neq m_i$ and $m'_i \neq$ FAIL.

Then $P_D(n)$ is negligible.

Definition of *Privacy* for such a cryptosystem:

For every polynomial bounded D:

D chooses an i.

D then chooses m_0, sees e_0, chooses m_1, sees e_1, ..., chooses m_{i-1}, sees e_{i-1}.

D now chooses two n-bit strings M^0 and M^1; a random $b \in \{0, 1\}$ is chosen (unknown to D) and $m_i = M^b$ is given to A, and D sees e_i.

D then chooses m_{i+1}, sees e_{i+1}, chooses m_{i+2}, sees e_{i+2}, ..., chooses m_{s-1}, sees e_{s-1}.

D then chooses a string γ of length ns and gives it to B, who then outputs $m'_0, m'_1, \ldots, m'_{s-1}$. D sees if and for which piece B outputs FAIL.

D then outputs a guess b' at b.

Let $P_D(n)$ = the probability that $b' = b$.

Then $P_D(n)$ is negligibly above $1/2$.

Remark: This definition may not be quite strong enough if we are only interested in privacy, but I believe the following definition for the (total) security of such a cryptosystem is a good one.

Definition: Such a cryptosystem is *secure* if it satisfies Integrity and Privacy.

(Let's go back and see why Cryptosystem III is secure.)

Note: We are *not* defending against "traffic analysis". That is, we assume adversaries always know who is talking with whom when.

Let's consider a different version of Cryptosystem III, where we try to use a version of cipher-block-chaining to accomplish Privacy.

Cryptosystem IV. The shared key is (k_1, k_2), $|k_1| = |k_2| = n$.
F is a pseudo-random *permutation* generator,
$F_k : \{0,1\}^n \to \{0,1\}^n$.
G is a pseudo-random function generator, $G_k : \{0,1\}^{2n} \to \{0,1\}^n$.
For n-bit message pieces $m_0, m_1, m_2 \ldots$,
A encrypts each m_i by computing
$\alpha_i = F_{k_1}(\alpha_{i-1} \oplus m_i)$ (where $\alpha_{-1} = F_{k_1}(0)$), and
$e_i = [\alpha_i, G_{k_2}(\bar{i}\, \alpha_i)]$.
Theorem: Cryptosystem IV does *not* satisfy Privacy. The adversary chooses $i = 2$, chooses m_0 to be anything, sees α_0, chooses $m_1 = \alpha_0$, sees $\alpha_1 = F_k(\bar{0})$, chooses $M^0 = \alpha_1$ and $M^1 \neq M^0$ and sees α_2; if $\alpha_2 = \alpha_1$ then he outputs 0, otherwise he outputs 1. It is easy to see that $q(n) = 1$.

NOTE: This insecurity is worse than it at first appears.

This problem can be fixed by revising the system by letting:
$\alpha_i = F_{k_1}(\alpha_{i-1}) \oplus m_i$ for $i \geq 0$.

Cryptosystem V. The shared key is k, $|k| = n$.
F is a pseudo-random permutation generator,
$F_k : \{0,1\}^{2n} \to \{0,1\}^n$.

For n-bit message pieces $m_0, m_1, m_2 \ldots$,
A encrypts each m_i by computing $e_i = F_k(\bar{i}\, m_i)$.

B decrypts: Given $e'_0, e'_1, e'_2, \ldots$, decrypt piece i as follows:
Compute $[u, v] = F_k^{-1}(e'_i)$ where $|u| = |v| = n$; if $u = \bar{i}$ then output $m'_i = v$, otherwise output $m'_j = \text{FAIL}$ for all $j \geq i$.

Is this secure? **YES**

Here is an example of a Cryptosystem using randomness.

Cryptosystem VI. The shared key is (k_1, k_2), $|k_1| = |k_2| = n$.
F is a pseudo-random function generator, $F_k : \{0,1\}^n \to \{0,1\}^n$.
G is a pseudo-random function generator, $G_k : \{0,1\}^{2n} \to \{0,1\}^n$.

For n-bit message pieces $m_0, m_1, m_2 \ldots$,
A encrypts each m_i by choosing a random n-bit r_i and computing
$\alpha_i = [r_i, m_i \oplus F_{k_1}(r_i)]$ and $e_i = [\alpha_i, G_{k_2}(\bar{i}\,\alpha_i)]$.

B decrypts: Given $e'_0, e'_1, e'_2, \ldots$, each $2n$ bits long, decrypt piece i
as follows: Say $e'_i = [\alpha'_i, \beta'_i]$. If $\beta'_i = G_{k_2}(\bar{i}\,\alpha'_i)$, then denoting
$\alpha'_i = [r'_i, \beta'_i]$ let $m'_i = \beta'_i \oplus F_{k_1}(r'_i)$;
else let $m'_j = \text{FAIL}$, for all $j \geq i$.

Is this secure? **YES**