Notes for Lecture 4 (Draft)

Summary

Last time we defined pseudorandom generators and proved that, if they exist, they provide message-indistinguishable (and hence semantically secure) one-time encryption.

How do we construct a pseudorandom generator? We can't if $P = NP$, so the security of any construction will have to rely on an unproved assumption which is at least as strong as $P \neq NP$. We shall see, later on, how to construct a pseudorandom generator based on well-established assumptions, such as the hardness of integer factorization, and we shall see that the weakest assumption under which we can construct pseudorandom generators is the *existence of one-way functions*.

Today, we shall instead look at RC4, a simple candidate pseudorandom generator designed by Ron Rivest. RC4 is very efficient, and widely used in practice – for example in the WEP standard for wireless communication. It is known to be insecure in its simplest instantiation (which makes WEP insecure too), but there are variants that may be secure.

This gives a complete overview of one-time symmetric-key encryption: from a rigorous definition of security to a practical construction that may plausibly satisfy the definition.

A usable, system, however, should be able to handle *multiple* encryptions. To define security for multiple encryptions, we have to define what an adversary is able to do with past messages.

In the most basic (and unsatisfactory) setting, the adversary simply sees the encryptions of past messages. (Some systems used in practice fail to satisfy even this very basic notion of security.) We can achieve this kind of security using a pseudorandom generator if the communicating parties keep state information between communication sessions and if messages are received in the order in which they were sent.

A more satisfactory notion of security allows the adversary to see encryptions of *known plaintexts*.

What's Coming Next

Using new primitives called *pseudorandom functions* and *pseudorandom permutations*, it is possible to construct encryption schemes that satisfy this notion of security and

that do not require synchronization between sender and receiver, and that do not require them to keep state information.

The best notion of security resists even an attack in which the adversary has the ability to see decryptions of chosen messages. This notion too is achievable via pseudorandom functions, but it will take us some time to develop the right tools to analyze a construction meeting this level of security.

How do we construct pseudorandom functions and permutations? It is possible to construct them from pseudorandom generators (and hence from one-way functions), and there are ad-hoc constructions which are believed to be secure.

1 RC4

RC4 is a very simple candidate pseudorandom generator. We will give a slightly generalized presentation of how it works.

Fix a modulus s, which is 256 in RC4, and let $\mathbb{Z}_s$ be the finite group of s elements $\{0, \ldots, s-1\}$ together with the operation of addition mod s. (The notation $\mathbb{Z}/s\mathbb{Z}$ is more common in math.)

The generator has two phases:

In the first phase, a short seed $K \in \{0,1\}^k$ is converted into a permutation $P : \mathbb{Z}_s \to Z_s$ as follows (id is the identity permutation $id(x) = x$, the variables a, b are in $\mathbb{Z}_s$ and so addition is performed mod s):

- $P := id$
- $b := 0$
- for a in $\{0, \ldots s-1\}$:
  - $b := b + P(a) + K(a \bmod k)$
  - swap $(P(a), P(b))$

(Note that if $k = s$ then the first phase has the following simpler description: for each $a \in \mathbb{Z}_s$, swap $P(a)$ with a random location.)

In the second phase, the permutation is used to produce the output of the generator as follows:

- $a := 0$; $b := 0$
- for $i := 1$ to m :

- $a := a + 1$
- $b = b + P(a)$
- output $P(P(a) + P(b))$
- swap $(P(a), P(b))$

In RC4, s is 256, as said before, which allows extremely fast implementations, and k is around 100.

The construction as above is known to be insecure: the second byte has probability 2^{-7} instead of 2^{-8} of being the all-zero byte.

There are other problems besides this bias, and it is possible to reconstruct the key and completely break the generator given a not-too-long sequence of output bits. WEP uses RC4 as described above, and is considered completely broken.

If one discards an initial prefix of the output, however, no strong attack is known. A conservative recommendation is to drop the first 4096 bits.

2 Security for Multiple Encryptions: Vanilla Version

Last time we introduced the following notion of security for multiple encryptions.

Definition 1 (Message indistinguishability for multiple encryptions) (Enc, Dec) *is* (t, ϵ)*-message indistinguishable for* c *encryptions if for every* $2c$ *messages* $M_1, \ldots, M_c$, $M'_1, \ldots, M'_c$ *and every* T *of complexity* $\leq t$ *we have*

$$|\mathbb{P}[T(Enc(K, M_1), \ldots, Enc(K, M_c)) = 1] - \mathbb{P}[T(Enc(K, M'_1), \ldots, Enc(K, M'_c)) = 1]| \leq \epsilon$$

If we allow the encryption algorithm to keep *state* information, then a pseudorandom generator is sufficient to meet this definition. Indeed, usually pseudorandom generators designed for such applications, including RC4, are optimized for this kind of "stateful multiple encryption."

3 Security for Multiple Encryptions: Chosen Plaintext Attack

In realistic scenarios, an adversary has knowledge of plaintext-ciphertext pairs. A broadly (but not fully) general way to capture this knowledge is to look at a model

in which the adversary is able to see *encryptions of arbitrary messages of her choice*. An attack in this model is called a Chosen Plaintext Attack (CPA).

If O is a, possibly randomized, procedure, and A is an algorithm, we denote by $A^O(x)$ the computation of algorithm A given x as an input and given the ability to execute O. We charge just one unit of time for every execution of O, and we refer to A as having *oracle access* to O.

Definition 2 (Message indistinguishability against CPA) *(Enc, Dec) is (t, ϵ)-message indistinguishable against CPA if for every 2 messages M, M' and every T of complexity $\leq t$ we have*

$$|\mathbb{P}[T^{Enc(K,\cdot)}(Enc(K, M)) = 1]-$$
$$\mathbb{P}[T^{Enc(K,\cdot)}(Enc(K, M')) = 1]| \leq \epsilon$$

This is a generalization of security for multiple encryptions

Lemma 3 *Suppose (Enc, Dec) is (t, ϵ)-message indistinguishable against CPA. Then for every c it is $(t - cm, c\epsilon)$-message indistinguishable for c encryptions.*