Notes for Lecture 4

Scribe: Anindya De, posted February 4, 2009

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.*

1 Description of RC4

We now present the description of RC4, a very simple candidate pseudorandom generator. This was proposed by Ron Rivest. Though there are some theoretical considerations in the choice of encryption scheme, we shall not be going into it. Below we give a slightly generalized description of RC4.

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:

The first phase is intended to construct a "pseudorandom" permutation. Before, we go into the actual construction used in RC4, we first give a description of how to construct a nearly random permutation given a sufficiently long seed. Note that the problem is non-trivial because even if we are given a very long random string and we interpret the string as a function in the obvious way, then it may not be a permutation. Hence, we need to try something more clever. A nearly random permutation may be created in the following way. Let K be a random element in $(\{0,1\}^8)^{256}$ which we may interpret as an array of 256 numbers each in the range $[0, 255]$. In particular, let $K(a)$ represent the a^{th} element. To create a permutation over 256 elements, do the following (id represents the identity permutation i.e. $id(x) = x$)

- $P := id$
- For $a \in \mathbb{Z}_{256}$
  - Swap $P(a)$ and $P(K(a))$

What distribution over permutations do we generate with the above process? The answer is not completely understood, but the final permutation is believed to be close to uniformly distributed.

However, the above process is inefficient in the amount of randomness required to create a random permutation. We now describe a process which is more randomness efficient i.e. uses a smaller key to construct a permutation (which shall now be "pseudorandom", although this is not meant as a technical term; the final permutation will be distinguishable from a truly random permutation). The seed $K \in \{0,1\}^k = (\{0,1\}^{\log_2 s})^t$ (interpreted as an array over elements in $\mathbb{Z}_s$ of length t) is converted into a permutation $P : \mathbb{Z}_s \to \mathbb{Z}_s$ as follows (the variables a, b are in $\mathbb{Z}_s$ and so addition is performed mod s):

- $P := id$
- $b := 0$

```
• for $a$ in $\{0, \dots s-1\}$ :
    – $b := b + P(a) + K[a \bmod t]$
    – swap $(P(a), P(b))$
```

(Note that if $k = s \log_2 s$ then the first phase has the following simpler description: for each $a \in \mathbb{Z}_s$, swap $P(a)$ with a random location, as in the simplified random process.)

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 which violates the definition of pseudorandom generator.

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 $M_1, \dots, M_c$, $M'_1, \dots, M'_c$ and every T of complexity $\leq t$ we have*

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

An important thing to be noted here is that encoding multiple messages requires the encryption scheme to be randomized even to meet the vanilla definition of multiple encryption security. This is because in case the scheme is deterministic, consider two distinct messages m_1 and m_2. Clearly for feasibility of decryption, $Enc(K, m_1) \neq Enc(K, m_2)$. Now consider the message pairs (m_1, m_1) and (m_1, m_2). For the first pair, $Enc(K, m_1), Enc(K, m_1)$ is of the form $x \circ x$ whereas $Enc(K, m_1), Enc(K, m_2)$ is of the form $x \circ y$ where $x \neq y$. Hence a deterministic encryption does not meet even the vanilla definition of security. In fact, with some kind of data frequency analysis, we might be able to break the pseudorandom generator.

We now describe how to meet this vanilla definition of security using pseudorandom generators. However, this requires the encryption and decryption to be stateful and further the messages to be synchronized i.e. they should be decrypted in the same order as they were sent. The idea is to use a PRG in order to expand the key and then keep moving a counter along the output of the PRG. Initially the counter is set to zero and if at the end of the $(i-1)^{th}$ encryption, the counter is at c_{i-1}, then in order to encrypt the i^{th} message (say m_i), it is XORed with the substring of the PRG output starting from the $(c_{i-1}+1)^{th}$ position to $(c_{i-1}+|m_i|)^{th}$ position. The counter is set to $c_{i-1} + |m_i|$ at the end of the encryption. The security of the encryption scheme follows because of the observation that this reduces to encryption of one message which is concatenation (in sequence) of the individual messages. Note that the PRG may not be "online" in the sense that it may be necessary to compute the 100^{th} bit before the 50^{th} bit can be computed. However, with a special kind of pseudorandom generators called Stream ciphers, this problem is avoided. In this, after encryption of the i^{th} message, a state is stored and encrypting the $(i+1)^{th}$ message, the PRG takes as input the key as well as the state. The state is updated after producing the output. This is much more "online" in some sense.

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). This model at least captures the situation when the messages which were initially secret have been made public in course of time and hence some plaintext ciphertext pairs are available to the adversary. Using new primitives called *pseudorandom functions* and *pseudorandom permutations*, it is possible to construct encryption schemes that satisfy this notion of security. In fact, an even stronger notion of security where adversary is allowed to see decryptions of certain chosen messages can also be constructed using 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. For the time being, we define security under CPA and show how it generalizes the notion of multiple encryption security given in the last section.

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*

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

We now prove that it is a generalization of security for multiple encryptions (as defined in the last section)

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.*

PROOF: We prove by contradiction i.e. we assume that there exist a pair of c messages $(M_1, M_2, \ldots, M_{c-1}, M_c)$ and $(M'_1, M'_2, \ldots, M'_{c-1}, M'_c)$ such that there is a procedure T' of complexity $(t - cm)$ which can distinguish between these two with probability greater than $c\epsilon$. We shall prove existence of two messages M and M' and an oracle procedure of complexity $\leq t$ such that

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

We shall assume a circuit model rather than the Turing machine model of computation here. We first start with a simple case i.e. let there exist pair of c messages

$M_1, M_2, \ldots, M_{c-1}, M_c$ and $M'_1, M'_2, \ldots, M'_{c-1}, M'_c$ such that the sequence differs exactly at one position say i i.e. $M_j = M'_j$ for $j \neq i$ and $M_i \neq M'_i$. By assumption, we can say that

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

The machine T on being given input C simulates machine T' as

$$T'(Enc(K, M_1), \ldots, Enc(K, M_{i-1}), C, Enc(K, M_{i+1}), \ldots, Enc(K, M_c))$$

using the oracle to know $Enc(K, M_j)$ for $j \neq i$. Clearly by assumption, T can distinguish between $Enc(K, M_i)$ and $Enc(K, M'_i)$ with probability more than $c\epsilon > \epsilon$. Also, the hardwiring of the messages M_j for $j \neq i$ increases the circuit complexity by at most cm (there are c of them and each is of length at most m). Hence T is a circuit of complexity at most t. To move to the general case where the sequences differ at more than one place, we use the hybrid argument which is a staple of proofs in cryptography. We shall use it here as follows. Consider c-tuple D_a defined as -

$$\begin{gathered} D_a = Enc(K, M_1), \ldots, Enc(K, M_{c-a}), \\ Enc(K, M'_{c-a+1}), \ldots, Enc(K, M'_c) \end{gathered}$$

Then, we have that $|\mathbb{P}[T'(D_0) = 1] - \mathbb{P}[T'(D_c) = 1]| > c\epsilon$ which can be rewritten as $|\sum_{a=0}^{c-1} \mathbb{P}[T'(D_a) = 1] - \mathbb{P}[T'(D_{a+1}) = 1]| > c\epsilon$. By triangle inequality, we get that there exists some j such that $|\mathbb{P}[T'(D_j)] - \mathbb{P}[T'(D_{j+1})]| > \epsilon$. However, note that D_j and D_{j+1} differ at exactly one position, a case which we have already solved. The only difference is that the distinguishing probability is ϵ rather than $c\epsilon$ because of introduction of the hybrid argument. $\square$