6.875J/18.425J Cryptography and Cryptanalysis	October 10, 2001
Lecture 9	
Instructor: Anna Lysyanskaya	*Scribed by: Deniss Čebikins*

1 The big picture

What is the big picture of what we do in cryptography? In general, as in any applied subject, there are some fundamental problems that we try to solve using some interesting building blocks. Let us talk about these problems and these building blocks. Some of them are already familiar, and others will at least make intuitive sense now that we have explored public-key encryption in depth.

1.1 Fundamental problems

Encryption (secure transmission of messages)

- Public key

- Private key (Alice and Bob have a secret to begin with; this type of encryption can be much more efficient than public key encryption, and it may not rely on assumptions as strongly.)

Pseudorandomness (efficient random generator)

This problem is about generating a long binary sequence from a shorter random sequence, so that the resulting long sequence is indistinguishable from a random sequence of the same length.

Authentication (verification of identity of the sender)

- For public key encryption, digital signatures are used. For secret key encryption, Message Authentication Codes (MAC) are used. In both cases a definition of security is needed to establish the notion of effectiveness of the authentication method.

Secure protocols Types of secure protocols:

- Contract signing

- Cryptosystem where majority of people can decrypt, but no minority can.

1.2 Building blocks (= cryptographic primitives)

Trapdoor predicates Tailor-made for public-key encryption. For example, QRA.

Predicate P is trapdoor if there are algorithms (G, D) such that G is a key generation algorithm with the property that, P is an encryption algorithm, and D is a decryption algorithm: (1) D faithfully determines the predicate, i.e., for all $(PK, SK) \in G$, $D(SK, P(PK, b)) = b$ and (2) the predicate is secure, i.e. for all PPTF $\{A_k\}$, there exists a negligible ν such that

$$\Pr[(PK, SK) \leftarrow G(1^k); b \leftarrow \{0,1\}; c \leftarrow P(PK, b); b' \leftarrow A_k : b = b'] \leq 1/2 + \nu(k)$$

As we saw in previous lectures, secure public-key encryption can be constructed from trapdoor predicates. The SK is the *trapdoor*.

Trapdoor one-way permutations For example, RSA.

Here, a function family a is a function that takes two inputs, a parameter PK, and another input x.

A function family p is a trapdoor permutation if there are efficient algorithms (G, S, D) such that (1) G is a key generation algorithm; (2) S is an algorithm that samples the input domain; (3) $D(SK, p(PK, x)) = x$ for all $(PK, SK) \in G$, for all $x \in S(PK)$ and (4) the permutation is hard to invert, i.e.: for all PPTF $\{A_k\}$, there exists a negligible ν such that

$$\Pr[(PK, SK) \leftarrow G(1^k); x \leftarrow S(PK); x' \leftarrow A_k(p(x)) : x = x'] \leq \nu(k)$$

One-way permutation family For example, exponentiation and DES, AES

This is a relaxation of trapdoor one-way permutations, in that we do not require algorithm D to be efficient. In other words:

A function family p is a one-way permutation if there are efficient algorithms (G, S), and a not necessarily efficient algorithm D such that AS ABOVE: (1) G is a key generation algorithm; (2) S is an algorithm that samples the input domain; (3) $D(SK, p(PK, x)) = x$ for all $(PK, SK) \in G$, for all $x \in S(PK)$ and (4) the permutation is hard to invert, i.e.: for all PPTF $\{A_k\}$, there exists a negligible ν such that

$$\Pr[(PK, SK) \leftarrow G(1^k); x \leftarrow S(PK); x' \leftarrow A_k(p(x)) : x = x'] \leq \nu(k)$$

One-way function family For example, multiplication, exponentiation, DES, AES

This is a relaxation of one-way permutation family in that we do not require that the trasformation should be invertible (1-1).

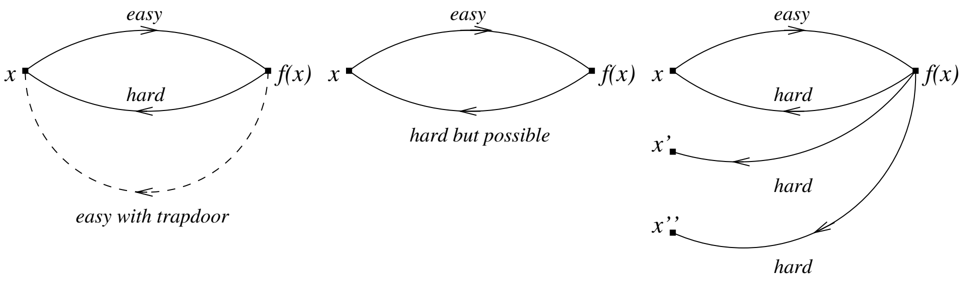
A function family f is one-way if there are efficient algorithms (G, S) such that (1) G is a key generation algorithm; (2) S is an algorithm that samples the input domain; (3) f is hard to invert, i.e.: for all PPTF $\{A_k\}$, there exists a negligible ν such that

$$\Pr[(PK, SK) \leftarrow G(1^k); x \leftarrow S(PK); x' \leftarrow A_k(f(x)) : f(x) = f(x')] \leq \nu(k)$$

A one-way function, as opposed to function family, is a function family with trivial key generation G whose output is null.

Other primitives This list is by no means exhaustive. There are other primitives which are of importance which we will see later.

The picture below illustrates three of the primitives above.



Trapdoor permutation *One-way permutation* *One-way function*

1.3 Preview of future material

How do the primitives relate to each other? Which primitives are sufficient to solve which fundamental problems? These are natural problems which we have been addressing so far, and will continue addressing throughout the rest of the semester.

2 Hardcore bits

In the spirit of relating primitives to each other, let us relate one-way predicates (resp. trapdoor predicates) and one-way permutations (resp. trapdoor permutations).

A hardcore bit for a one-way permutation family p is a boolean function b such that, given $p(PK, x)$, it is hard to determine $b(PK, x)$. In addition, b should be a non-trivial function, i.e. $b(PK, x)$ should be true for approximately $1/2$ of possible values of x specified by PK. Here is the rigorous definition.

Definition. Let $p(\cdot, \cdot)$ be a one-way permutation with key generator G and sampler algorithm S. A boolean function $b(PK, x)$ is a *hardcore bit for p* if the following conditions hold:

(i) There exists a negligible function $\mu(k)$ such that for all $PK \in G(1^k)$, $|Pr[x \leftarrow S(PK) : b(PK, x)] - 1/2| \leq \mu(k)$;

(ii) For all $PPTF$ $\{A_k\}$ there exists a negligible function $\nu(k)$ such that

$$\Pr[PK \leftarrow G(1^k); x \leftarrow S(PK); b' \leftarrow A_k(PK, p(PK, x)) : b' = b(PK, x)] \leq 1/2 + \nu(k)$$

Example. Suppose p is a OWP (one-way permutation) and $x = x_0 \circ x_1 \circ \ldots \circ x_l$. Is $b(PK, x) = x_0$ a hardcore bit? Not necessarily! Suppose $p(PK, x) = x_0 \circ p'(x')$, where p' is a OWP of $(l-1)$-bit strings, and x' is obtained from x by removing bit x_0. Obviously, there is an adversary which can always determine $b(PK, x)$ from $p(PK, x)$ without knowing x.

Theorem. Any one-way permutation has a hardcore bit.

This theorem will be proved in the next lecture.

3 Pseudorandom generators

3.1 Motivation

Random sequence generation is a crucial step of secure message encryption. Unfortunately, producing random bits is a fairly complex operation. In order to reduce time complexity of secure communication, we would like to obtain an algorithm which would produce essentially random sequences by utilizing ing only a small number of random bits. Such algorithms are called Pseudorandom Generators (PRG).

3.2 Definition

Let $l < m$, and let G is a deterministic algorithm which, given a sequence of l bits, outputs a sequence of m bits. G is a pseudorandom generator (PRG) if for all $PPTF$ A_l there is a negligible function $\nu(l)$ such that

$$\Pr[x \leftarrow \{0,1\}^l; y_0 \leftarrow G(x); y_1 \leftarrow \{0,1\}^m; b \leftarrow \{0,1\}; b' = A_l(y_b) \ : \ b' = b] \leq 1/2 + \nu(l)$$

The intuition of this definition is that the resulting string is just as good as a random string, because no one can tell whether it was selected by flipping a coin m times, or by running G on a random l-bit string.

3.3 PRG using HB

Here is an example of a PRG that uses the fact that every one-way permutation has a hardcore bit (HB). Let p be a one-way permutation, and let b be its hardcore bit. Given PK and a seed s (s is a short sequence of random bits) we can construct a long "pseudorandom" sequence $y_0 \circ y_1 \circ y_2 \circ \ldots$ as follows:

$$\begin{array}{ccccccc} PK, s & \rightarrow & p(PK,s) = x_1 & \rightarrow & p(PK,x_1) = x_2 & \rightarrow & \ldots \\ & & \downarrow & & \downarrow & & \\ b(PK,s) = y_0 & \circ & b(PK,x_1) = y_1 & \circ & b(PK,x_2) = y_2 & \circ & \ldots \end{array}$$

The sequence $y_0 \circ y_1 \circ y_2 \circ \ldots$ can be made as long as required. On the next problem set, you will prove that the resulting sequence is indeed indistinguishable from random, i.e. the above algorithm is a PRG.