# Foundations of Cryptography

Notes of lecture No. 8A (given on Apr. 30th by Oded GOLDREICH)

Notes taken by Yaron Kretchmer and Itzhak Parnafes

**Summary**

In this lecture we present a public-key encryption system based on the existence of one way permutations with trapdoor. We prove that it is semantically secure.

## 1. One - Way Permutations with Trapdoor

We recall the definition of a one-way functions with trapdoor as introduced in lecture 3. We consider a set of one-way permutations

$$\{f_i : D_i \rightarrow D_i\}\ ,\ \ i \in I \subseteq \{0,1\}^*$$

where the $D_i$ are finite domains, and $I$ is some index set.
**example:** in **RSA** scheme the functions are $\{f_{(N,e)} : Z_N^* \rightarrow Z_N^*\}$, where $N = p \cdot q$ for primes $p, q$ and $e$ is relatively prime with $\phi(N)$.
The functions $f_i$ satisfy the following requirements: (all algorithms mentioned hereafter are polynomial-time)

**(1)** There exists a sampling algorithm $S_1$ such that,

$$S_1(1^n) \text{ ranges over } \left[ I \cap \{0,1\}^n \right] \times \{0,1\}^*$$

Namely, in addition to the index $i$ in I, the algorithm $S_1$ gives a value $t(i)$, called the *trapdoor*.
The length $n$ is called the *"security parameter"*.
**example:** in the **RSA**, $S_1(1^n) = \left[ (N,e), d \right]$, where $N$ is a product of two primes $p$ and $q$ of length $\frac{n}{4}$, and $e$ is relatively prime to $\phi(N)$, the trapdoor $t(N,e)$ is $(N,d)$ such that $d = (e^{-1} \bmod \phi(N))$. Alternatively, one can consider as the *trapdoor* the pair of primes $(p,q)$, from which the value of $d$ can be computed in polynomial time.
There are several probabilistic polynomial algorithms that can output $p$, $q$ and $e$ as requested.

**(2)** There exists a sampling algorithm $S_2$, such that

$$S_2(i) \text{ ranges over } D_i$$

**example:** in the **RSA** the output of $S_2(N,e)$ is a random element from $Z_N^*$.

**(3)** The functions $f_i$ are polynomial time computable, that is there exists an algorithm $A_1$ such that

$$A_1(i,x) = f_i(x)$$

**example:** in the **RSA**, $A_1((N, e), x) = (x^e \bmod N)$

**(4)** The functions $f_i$ are easy to invert given the trapdoor $t(i)$, i.e. there exists an algorithm $A_2$, such that

$$A_2(t(i),y) = f_i^{-1}(y)$$

**example:** In the *RSA*, $A_2((N,d),y) = (y^d \bmod N) = (x \bmod N)$, where $y = (x^e \bmod N)$.

**(5)** The functions $f_i$ are hard to invert (without the trapdoor information) i.e. for all probabilistic polynomial algorithm $A'$, $\forall c>0\ \exists N>0\ \forall n>N$

$$\text{Prob}(\ A'(i,\ f_i(x)) = x) < \frac{1}{n^c}$$

Where $i$ is from the distribution induced by $S_1(1^n)$, and $x$ is taken from the distribution of $S_2(i)$.

From requirements (4) and (5) it follows that the value of $t(i)$ is hard to obtain from $i$, otherwise an algorithm $A'$ computing $t(i)$ and applying $A_2$ would invert the functions $f_i$.

**Remark:** If in the above requirement (5), we replace "for all probabilistic polynomial time algorithms $A'$" by "for all non-uniform polynomial algorithms $A'$", we get a family of permutations which are one-way in a "non-uniform sense" (see lecture 2).

**Reminder:** We will later use the definition of *hard-core*: (see lecture 4)

A predicate $b: \{0,1\}^* \rightarrow \{0,1\}$ is called a *hard-core* of a one-way function, $f$, if it is polynomial time computable but for all polynomial-time probabilistic algorithm $A'$ $\forall c>0\ \exists N>0\ \forall n>N$

$$\text{Prob}(A'(f(x)) = b(x)) < \frac{1}{2} + \frac{1}{n^c}$$

## 2. Construction of a secure encryption system

Following we describe a secure public-key encryption system. It is based on a family of one-way permutations as defined above and on a hard-core predicate $b(\cdot)$ for this family. (As was shown in lecture 4, any such family has a hard-core predicate.)

The encryption system is a triplet (G,E,D) of probabilistic polynomial-time algorithms:

**Key Generation :** $G(1^n) \rightarrow (i,t(i))$

Where $e{=}i$ is the (public) encryption key, and $d{=}t(i)$ is the decryption key.

**Encryption:** Let $\alpha$ be an $m$-bit message, $\alpha = \sigma_1 \cdots \sigma_m$

Define $E_e(\alpha) = ((c_1,y_1),\cdots,(c_m,y_m))$, where $c_j$ and $y_j$ are computed as follows:

for $j = 1$ to $m$

$x_j \leftarrow S_2(e)$ (* i.e. select $x_j$ at random in $D_e$ *)

$y_j \leftarrow A_1(e,x_j)$ (* i.e. $y_j{=}f_e(x_j)$ *)

$c_j \leftarrow \sigma_j \oplus b(x_j)$

end.

Note that the coin tosses of $S_2$ during its repetitive activations are probabilistically independent.

**Decryption:** Let $C = ((c_1, y_1), \cdots, (c_m, y_m))$ be an encrypted message. The decryption algorithm $D_d(C) = \sigma_1 \cdots \sigma_m$, is:

for $j = 1$ to $m$

$x_j \leftarrow A_2(d, y_j)$ (* $x_j = f_e^{-1}(y_j)$ *)

$b_j \leftarrow b(x_j)$

$\sigma_j \leftarrow b_j \oplus c_j$

end.

**Theorem 1:** if $\{f_i\}_{i \in I}$ is a collection of one-way permutations with a trapdoor, then the system described above is a secure public-key encryption system. (The encryption system in uniformly / non-uniformly secure according to the uniform / non-uniform one-wayness of the family $\{f_i\}$).

**Proof:** First, note that $D_d E_e(m) = m$, namely that the above system constitutes an encryption system. Next we will show that the encryption system is secure in the sense of indistinguishability. By the equivalence theorem proved in lecture No. 7, it follows that the encryption system is semantically secure.
The theorem follows from the following two lemmas:

**Lemma 1:** The above encryption system is secure in the sense of indistinguishability, when encrypting one-bit messages.

**Lemma 2:** If the above encryption system is secure for one-bit messages, then it is secure for encrypting messages with length polynomial in the key length.

**Proof of Lemma 1:** Let us assume, on the contrary, that there is a polynomial probabilistic algorithm, $A$, which distinguishes between the encryptions of the bits 0 and 1. That is, there exists a distinguishing algorithm $A$, a constant $c_0$ and infinitely many $n$'s for which

$$\left| \mathrm{Prob}\left[A(E_e(0)) = 1\right] - \mathrm{Prob}\left[A(E_e(1)) = 1\right] \right| > \frac{1}{n^{c_0}}$$

where the probabilities are taken over the key space $G(1^n)$ and over the coin tosses of $A$. Without loss of generality we may assume that the above inequality holds for infinitely many $n$'s even without the absolute value. Otherwise, we may use $\bar{A}$ instead of $A$, and it will satisfy the inequality. So, we have

$$\mathrm{Prob}\left[A(E_e(0)) = 1\right] - \mathrm{Prob}\left[A(E_e(1)) = 1\right] > \frac{1}{n^{c_0}} \quad (1)$$

Since $E_e(\sigma) = (\sigma \oplus b(x), f_e(x))$, for a randomly sampled $x$, we get from (1)

$$\mathrm{Prob}\left[A(b(x), f_e(x)) = 1\right] - \mathrm{Prob}\left[A(\overline{b(x)}, f_e(x)) = 1\right] > \frac{1}{n^{c_0}} \quad (2)$$

(we use $\overline{b(x)}$ for $1 \oplus b(x)$).

Because of this bias in $A$'s output, we can consider $A$ as a "judge" which on input $(\sigma, f_e(x))$ for $\sigma \in \{0,1\}$, outputs a vote ('1') indicating $b(x)=\sigma$, or a vote ('0') for $b(x)=\overline{\sigma}$. For infinitely many $n$'s this judge is correct in a significant majority of cases. Based on this observation, we construct the following algorithm $A'$ which "guesses" $b(x)$ from $f_e(x)$ with probability significantly better than ½.

Algorithm $A'$ on input $f_e(x)$:

choose $\sigma \in_R \{0,1\}$

if $A(\sigma, f_e(x)) = 1$ then output $\sigma$

else output $\overline{\sigma}$

The success probability for $A'$ is

$$\mathrm{Prob}\left[A'(f_e(x)) = b(x)\right] =$$

$$\mathrm{Prob}\left[A(\sigma, f_e(x))=1 \wedge \sigma=b(x)\right] + \mathrm{Prob}\left[A(\sigma, f_e(x))=0 \wedge \sigma=\overline{b(x)}\right]$$

$$= \mathrm{Prob}\left[A(\sigma, f_e(x)) = 1 \mid \sigma=b(x)\right] \cdot \mathrm{Prob}(\sigma=b(x)) + \mathrm{Prob}\left[A(\sigma, f_e(x)) = 0 \mid \sigma=\overline{b(x)}\right] \cdot \mathrm{Prob}(\sigma=\overline{b(x)})$$

Since both $\mathrm{Prob}[\sigma=b(x)]$ and $\mathrm{Prob}[\sigma=\overline{b(x)}]$ differ from ½ by a negligible amount, (otherwise $b(\cdot)$ cannot be a hard-core at all), we get that (up to a negligible difference)

$$\mathrm{Prob}\left[A'(f_e(x)) = b(x)\right] =$$

$$= \mathrm{Prob}\left[A(b(x), f_e(x)) = 1\right] \cdot \frac{1}{2} + \mathrm{Prob}\left[A(\overline{b(x)}, f_e(x)) = 0\right] \cdot \frac{1}{2}$$

$$= \frac{1}{2} \cdot \left[\mathrm{Prob}\left[A(b(x), f_e(x)) = 1\right] + 1 - \mathrm{Prob}\left[A(\overline{b(x)}, f_e(x)) = 1\right]\right]$$

From (2) we get

$$\mathrm{Prob}\left[A'(f_e(x)) = b(x)\right] > \frac{1}{2} + \frac{1}{2 \cdot n^{c_0}}$$

in contradiction with the security of the hard-core. □

We prove Lemma 2 in both non-uniform and uniform cases. We use the following definition:

**Definition (hybrid):** Let $\alpha = a_1 \cdots a_m$ and $\beta = b_1 \cdots b_m$ be two $m$-bit strings. The $i$*–hybrid* of $\alpha$ and $\beta$ is the $m$-bit string $\gamma^{(i)} = a_1 \cdots a_i b_{i+1} \cdots b_m$.

We will use the notion of hybrids several times during the course, so we will devote a few words for it.

The procedure is as follows: suppose we are given two elements (or distributions), and their "distance" is greater than some $d$. Now, we build a sequence of $n$ intermediate elements, hybrids, that serve as a "bridge" between the two given elements. Using the following simple claim, we see that there must be a successive pair of hybrids with a distance of at least $\frac{d}{n}$.

**Claim:** Given a series of numbers $P_0 \cdots P_n$, such that $P_n - P_0 > d$, there exists $i$, $0 \le i < n$, such that $P_{i+1} - P_i > \frac{d}{n}$.

The way we shall typically use the hybrids technique is by first showing that the difference between successive pairs is negligible, and then concluding that the difference between the given elements, which are the extreme hybrids, must be also negligible.

**Proof of Lemma 2 (Non - Uniform case)**

Assume the Lemma does not hold. That is, there exists a family $\{C_n\}$ of polynomial-size circuits and a sequence of pairs $\alpha_n$, $\beta_n$, ($|\alpha_n| = |\beta_n| = m$), such that for a positive constant $c_0$ and infinitely many $n$'s,

$$\left| \left[ \mathrm{Prob}(C_n(E_e(\alpha_n),e)=1) - \mathrm{Prob}(C_n(E_e(\beta_n),e)=1) \right] \right| > \frac{1}{n^{c_0}}$$

As before, we may omit the absolute value. For such a pair $\alpha_n$, $\beta_n$, consider the corresponding hybrids $\gamma_n^{(i)}$, $0 \le i \le n$. Obviously $\alpha_n = \gamma_n^{(m)}$, $\beta_n = \gamma_n^{(0)}$. Thus,

$$\left[ \mathrm{Prob}(C_n(E_e(\gamma_n^{(m)}),e) = 1) - \mathrm{Prob}(C_n(E_e(\gamma_n^{(0)}),e) = 1) \right] > \frac{1}{n^{c_0}}$$

By the above claim, there exists $i_0 \in \{1 \cdots m\}$ and infinitely many $n$'s such that

$$\left[ \mathrm{Prob}(C_n(E_e(\gamma_n^{(i_0)}),e) = 1) - \mathrm{Prob}(C_n(E_e(\gamma_n^{(i_0-1)}),e) = 1) \right] > \frac{1}{m \cdot n^{c_0}} = \frac{1}{n^{c_1}} \qquad (3)$$

That is, there are two successive hybrids whose encryptions can be distinguished by the above circuit. We will now construct a second family of circuits, $C'_n$ which will decrypt one-bit messages. The circuit $C'_n$ takes advantage of the bias in (3) in the following way.

The circuit $C'_n$ has $i_0$, $\alpha_n = a_1 a_2 \cdots a_m$ and $\beta_n = b_1 b_2 \cdots b_m$ wired in. On input $(E_e(\sigma),e)$ the circuit $C'_n$ first computes the encryptions of the first $i_0 - 1$ bits of $\alpha_n$ and of the last $m - i_0 - 1$ bits of $\beta_n$ by the key $e$, and then concatenates the encryptions of the first $i_0 - 1$ bits of $\alpha_n$, the encryption of $\sigma$ (the input), and encryptions of the last $m - i_0 - 1$ bits of $\beta_n$, and feeds it to the circuit $C_n$. The output of $C'_n$ is the output of $C_n$.

Assume, without loss of generality, that $a_{i_0} = 1$ and $b_{i_0} = 0$. It is clear that $a_{i_0} \ne b_{i_0}$, since otherwise $\gamma_n^{(i_0)} = \gamma_n^{(i_0-1)}$, and **no** algorithm can distinguish between identical objects! We show now that $C'_n$ distinguishes between the encryptions of 0 and 1, in contradiction to the hypothesis of the Lemma.

When $\sigma = b_{i_0} = 0$, the concatenated string is $\gamma_n^{(i_0-1)}$. Therefore

$$\mathrm{Prob}(C'_n(E_e(0),e) = 1) = \mathrm{Prob}(C_n(E_e(\gamma_n^{(i_0-1)},e) = 1)$$

Similarly,

$$\mathrm{Prob}(C'_n(E_e(1),e) = 1) = \mathrm{Prob}(C_n(E_e(\gamma_n^{(i_0)},e) = 1)$$

From (3) the contradiction to the security of one-bit encryption arises. □

**Proof of Lemma 2 - (Uniform case)**

The proof follows the same outline as in the non-uniform case. The difference is that we cannot "wire" the sequence $\{\alpha_n, \beta_n\}$ in the algorithm, but have to sample it by polynomial uniform means.

Let $\{X_n\}$ and $\{Y_n\}$ be two sequences of random variables, and let us assume that there is a distinguishing probabilistic polynomial algorithm $A$, such that there exists $c_0$ and infinitely many $n$'s,

$$\sum_{\alpha,\beta} \mathrm{Prob}(X_n{=}\alpha, Y_n{=}\beta) \cdot \left| \mathrm{Prob}(A(\alpha,\beta,E_e(\alpha),e) = 1) - \mathrm{Prob}(A(\alpha,\beta,E_e(\beta),e){=}\ 1\ )\right| > \frac{1}{n^{c_0}} \tag{4}$$

We will call "good" the pairs $(\alpha,\beta)$ for which

$$\left| \mathrm{Prob}(A(\alpha,\beta,E_e(\alpha),e) = 1) - \mathrm{Prob}(A(\alpha,\beta,E_e(\beta),e) = 1) \right| \geq \frac{1}{2 \cdot n^{c_0}}$$

and we get from (4)

$$\sum_{\text{good }(\alpha,\beta)} \mathrm{Prob}(X_n{=}\alpha, Y_n{=}\beta) > \frac{1}{2 \cdot n^{c_0}}$$

namely, that the probability to sample from the "good" set is non-negligible.

Again, we will construct an algorithm $A'$ that will decrypt one-bit messages with significant probability:

$A'$ is given as input $(E_e(\sigma),e)$ and performs the following steps :

**(1)** sample $\alpha,\beta$ from $(X_n, Y_n)$

**(2)** Check whether $(\alpha,\beta)$ is a good pair. This is done by testing $A$ on $E_e(\alpha)$ and $E_e(\beta)$ sufficiently many times in order to get a good approximation of the difference $\mathrm{Prob}[A(\alpha,\beta,E_e(\alpha),e) = 1] - \mathrm{Prob}[A(\alpha,\beta,E_e(\beta),e) = 1]$. If $(\alpha,\beta)$ are seen to be "bad" - return to step (1).

As in the uniform case, we are allowed to remove the absolute value because we may choose either $A$ or the inverse algorithm $\overline{A}$, according to our needs. To ensure recognizing with high probability pairs with probability difference of $\frac{1}{2 \cdot n^{c_0}}$, we may accept as "good" those that were measured to have a difference of more then $\frac{1}{4 \cdot n^{c_0}}$. The algorithm, after finding the desired "almost good" samples, continues the same as in the non-uniform case, now $\alpha$ and $\beta$ serve as $\alpha_n$ and $\beta_n$ in the non-uniform algorithm. Instead of having $i_0$ wired in, the algorithm can choose it at random. The probability calculation of the non-uniform case applies here too, with differences emerging due to the "guessing" of $i$, and the fact that we are not sure that the pair found is actually good.

The modification is as follows: if we choose $i$ at random with uniform distribution, the average of the differences of the probabilities, assuming the algorithm actually found a "good" pair, is

$$\sum_{i=0}^{m-1} \left[ \mathrm{Prob}\left[A(\alpha,\beta,E_e(\gamma_n^{i+1}),e){=}1\right] - \mathrm{Prob}\left[A(\alpha,\beta,E_e(\gamma_n^{i}),e){=}1\right]\right] \cdot \mathrm{Prob}(i_0{=}i)$$

$$= \frac{1}{m} \cdot \sum_{i=0}^{m-1} \left[ \mathrm{Prob}\left[A(\alpha,\beta,E_e(\gamma_n^{i+1}),e){=}1\right] - \mathrm{Prob}\left[A(\alpha,\beta,E_e(\gamma_n^{i}),e){=}1\right]\right]$$

$$=\frac{1}{m}\cdot\left[\text{Prob}\left[A\,(\alpha,\beta,E_e(\alpha),e){=}1\right]-\text{Prob}\left[A\,(\alpha,\beta,E_e(\beta),e){=}1\right]\right] > \frac{1}{m\cdot 4\cdot n^{c_0}}$$

Namely, the lower bound of probability of distinguishment between $\alpha$ and $\beta$ decreased with a factor 4 from the non-uniform case.

**Remarks:**

(1) There is a negligible probability that the sampling algorithm, even after running for a long time, will not find a "good" pair. We may set a pre-defined limit for the run time, and if the algorithm fails to find a "good" pair in that limit, it will halt. Of course, in this case the distinguishing algorithm will fail, but it will happen only with exponentially small probability.

(2) There is a exponentially small probability that the sampling algorithm will declare a pair as "good" even if it is not. Then again, the distinguishing algorithm may fail.

Those remarks cause a certain decrease of the distinguishing probability but not one that is significant.

Again, we have built a system which decrypts a one-bit messages with a non-negligible probability of success in contradiction with Lemma 1 □

From Lemma 1 and Lemma 2 it follows that the proposed encryption system is secure, proving Theorem 1 ■

**Note:** A drawback of the proposed encryption system is that the encryption expands *m*-bit messages to cryptograms of length $m\cdot n$, where $n$ is the *security parameter*. In the next lecture we will present a encryption system which, while maintaining the security conditions, expand *m*-bit messages into $m+n$ bit cryptograms, which is reasonable for practical use.