CMSC 858K — Advanced Topics in Cryptography January 29, 2004

Lecture 2

Lecturer: Jonathan Katz *Scribe(s): Alvaro A. Cardenas, Avinash J. Dalal, Julie Staub*

1 Summary

In the last set of notes the concept of a trapdoor permutation was discussed. In this set of lecture notes we begin by defining a *public-key encryption scheme*, and what it means for that scheme to be *semantically secure*. We show that if a public-key encryption scheme is secure under this definition then the encryption algorithm cannot be deterministic. We then define a *hardcore bit* and use it to build a provably-secure public-key encryption scheme.

2 Public-Key Cryptography

Definition 1 A **public-key encryption scheme** is a triple of PPT algorithms $(\mathsf{Gen}, \mathcal{E}, \mathcal{D})$, where

1. Gen is the key generation algorithm. $\mathsf{Gen}(1^k)$ outputs a pair (pk, sk). We assume for simplicity that $|pk| = k$.
2. $\mathcal{E}$ is the encryption algorithm. Given a plaintext m from a message space $\mathcal{M}$, algorithm $\mathcal{E}_{pk}(m)$ returns a ciphertext C of polynomial length $p(k)$.
3. $\mathcal{D}$ is the decryption algorithm. $\mathcal{D}_{sk}(C)$ returns a message m or the symbol $\perp$ representing incorrect decryption. Incorrect decryption can happen for example if C is not a valid ciphertext. (We will assume for simplicity — unless stated otherwise — that the decryption algorithm is deterministic.)
4. The public-key encryption scheme must satisfy **correctness:** i.e., for all $m \in \mathcal{M}$ and all possible (pk, sk) output by Gen, we have $\mathcal{D}_{sk}(\mathcal{E}_{pk}(m)) = m$.

$\diamondsuit$

In the following we assume that authentication of the public keys is possible and thus our main security concern is an attacker with access to the public key who attempts to obtain information about the plaintext from the ciphertext. Since the adversary has access to the public key pk, she can encrypt any message she wants and thus this scenario is sometimes known as a **chosen plaintext attack (CPA)**.

Our following definition of a secure public-key encryption scheme is strong in the sense that we do not only require that an adversary cannot obtain the plaintext m from the knowledge of the public-key pk and the ciphertext C, but also that an adversary cannot obtain any partial information about m (except probably some information about the length). This security notion is known as *semantic security* or *indistinguishability* [3, 1].

The security of the scheme is stated as a game in which an adversary has the ability to select two messages. One of the messages is randomly selected and encrypted. The encryption is then called secure if the adversary cannot do better than a random guess in finding out which message was encrypted. Before we formalize the game we recall what a negligible function is.

Definition 2 A function $\varepsilon(\cdot) : \mathbb{N} \to [0, 1]$ is **negligible** iff $\forall\ c > 0$, there exists an $N_c > 0$ such that $\forall\ N > N_c$ we have $\varepsilon(N) < 1/N^c$. $\diamondsuit$

An easier way of saying this is that $\varepsilon(\cdot)$ is negligible iff it grows smaller than any inverse polynomial. A very common example of a negligible function is the inverse exponential, $\varepsilon(k) = 2^{-k}$. Note that $2^{-k} = \mathcal{O}(1/k^c)$ for any c. We will use this definition of a negligible function to explicitly define what it means for an encryption scheme to be secure.

Definition 3 A public-key encryption scheme $(\mathsf{Gen}, \mathcal{E}, \mathcal{D})$ is **semantically secure** if for all PPT algorithms A, the following is negligible:

$$\left| \Pr\left[\begin{array}{c} (pk, sk) \leftarrow \mathsf{Gen}(1^k); (m_0, m_1) \leftarrow A(pk); \\ b \leftarrow \{0,1\}; C \leftarrow \mathcal{E}_{pk}(m_b); b' \leftarrow A(pk, C) \end{array} \ :\ b = b' \right] - \frac{1}{2} \right|.$$

$\diamondsuit$

Theorem 1 *If a public-key encryption scheme is semantically secure, then the encryption algorithm is not deterministic.*

Proof Consider an adversary who outputs (m_0, m_1) with $m_0 \neq m_1$. When presented with a ciphertext C, which is either an encryption of m_0 or m_1, compute $C_0 = \mathcal{E}_{pk}(m_0)$. If $C = C_0$ output 0 else output 1. This adversary succeeds in guessing b (cf. the above game) with probability 1; we use the fact that decryption succeeds with probability 1 and hence the space of encryptions of m_0 must be disjoint from the space of encryptions of m_1. ■

In the first lecture we defined what one-way trapdoor permutations are. Intuitively a one-way trapdoor permutation seems to be a good suggestion for a public-key encryption scheme as it easy to evaluate the function (encrypt) and hard to invert without the trapdoor (decrypt). More formally, given a one-way trapdoor permutation Gen_{td}, it is tempting to use the following encryption scheme: to generate keys, run $\mathsf{Gen}_{td}(1^k)$ to obtain (f, f^{-1}). Set $pk = f$ and $sk = f^{-1}$. Set the encryption algorithm $\mathcal{E}_f(\cdot) = f(\cdot)$, and set the decryption algorithm $\mathcal{D}_{f^{-1}}(\cdot) = f^{-1}(\cdot)$. However, from Theorem 1 we can conclude that a one-way trapdoor permutation cannot be used as a semantically secure public-key encryption scheme because the evaluation algorithm (i.e., computing $f(\cdot)$) is deterministic. Note in particular that "textbook RSA" (where encryption is $\mathcal{E}_{(N,e)}(m) = m^e \bmod N$) is susceptible to the adversary in the proof of Theorem 1. (It should also be clear, however, that the problems of the above approach — and in particular the case of "textbook RSA" — go beyond the fact that encryption is deterministic. For example, randomly padding the message before encrypting is not sufficient to guarantee semantic security either.)

However not all hope for using one-way trapdoor permutations as a basis for a secure encryption scheme is lost. First we will need to define hard-core bits.

3 Hard-Core Bits

Another problem with using one-way trapdoor permutations to encrypt (as suggested above) is that they can potentially reveal some information about the input when we have access to the output. For example, if $f(x)$ is a one-way trapdoor permutation, then it is easy to show that the function $f'(x_1|x_2) = x_1|f(x_2)$ (for $|x_1| = |x_2|$) is also a one-way trapdoor permutation. Here, however, we see that f' reveals half of the bits of its input directly. A *hardcore bit* of a one-way permutation is a bit of information that cannot be correctly identified better than with random guessing. Hardcore bits help us to formalize the notion of a single bit of information about the input x that is effectively obscured by the action of a one-way trapdoor permutation. More formally we have:

Definition 4 Let $H = \{h_k : \{0,1\}^k \to \{0,1\}\}_{k \geq 1}$ be a collection of efficiently-computable functions and let $\mathcal{F} = (\mathsf{Gen}_{td})$ be a trapdoor permutation. H is a **hard-core bit** for $\mathcal{F}$ if for all PPT algorithms A, the following is negligible (in k):

$$\left|\Pr[(f, f^{-1}) \leftarrow \mathsf{Gen}_{td}(1^k); x \leftarrow \{0,1\}^k; y = f(x) : A(f, y) = h_k(x)] - \frac{1}{2}\right|.$$

$\diamondsuit$

Theorem 2 ([2]) ***Existence of hard-core bits.*** *Let $\mathcal{F} = (\mathsf{Gen}_{td})$ be a trapdoor permutation with $f : \{0,1\}^k \to \{0,1\}^k$ (for security parameter k). Consider the permutation family $\mathcal{F}' = (\mathsf{Gen}'_{td})$ with $f' : \{0,1\}^{2k} \to \{0,1\}^{2k}$ defined as $f'(x|r) \overset{\text{def}}{=} f(x)|r$, and the function family $\mathcal{H} = \{h_k : \{0,1\}^{2k} \to \{0,1\}\}$ defined by $h_k(x|r) \overset{\text{def}}{=} x \cdot r$ (where "$\cdot$" represents the binary dot product). Then $\mathcal{F}'$ is a trapdoor permutation with hard-core bit $\mathcal{H}$.*

Recall that if $x = x_1 x_2 \ldots x_k \in \{0,1\}^k$ and $r = r_1 r_2, \ldots r_k \in \{0,1\}^k$ then $x \cdot r \overset{\text{def}}{=} x_1 r_1 \oplus x_2 r_2 \oplus \cdots \oplus x_k r_k = \bigoplus_{i=1}^k x_i r_i$ (where $\oplus$ represents binary exclusive-or). For example, $1101011 \cdot 1001011 = 1 \oplus 0 \oplus 0 \oplus 1 \oplus 0 \oplus 1 \oplus 1 = 0$.

4 Public-Key Encryption From Trapdoor Permutations

In the following we assume for simplicity that $\mathcal{M} = \{0,1\}$, i.e. we only are interested in encrypting single-bit messages (we will later show how any single-bit encryption scheme can be used to derive an encryption scheme for poly-many bits). Given a trapdoor permutation $\mathcal{F} = (\mathsf{Gen}_{td})$, construct the following encryption scheme

1. $\mathsf{Gen}(1^k)$:
   $(f, f^{-1}) \leftarrow \mathsf{Gen}_{td}(1^k)$
   Select a random r: $r \leftarrow \{0,1\}^k$
   Output $pk = (f, r)$ and $sk = f^{-1}$
2. $\mathcal{E}_{pk}(m)$ (where $m \in \{0,1\}$):
   pick $x \leftarrow \{0,1\}^k$
   Compute $y = f(x)$
   Compute $h' = x \cdot r$
   Output $C = \langle y | h' \oplus m \rangle$

3. $D_{sk}(y|b)$ (where $|y| = k$ and $|b| = 1$):
   Output $(f^{-1}(y) \cdot r) \oplus b$

Correctness Note that if $y|b$ is a valid encryption of m then $f^{-1}(y) = x$ and $b = (x \cdot r) \oplus m$. So the decryption algorithm will output $(x \cdot r) \oplus (x \cdot r) \oplus m = m$.

Theorem 3 *Assuming $\mathcal{F}$ is a trapdoor permutation, the encryption scheme presented above is semantically secure.*

Proof Assume toward a contradiction that the encryption scheme is not semantically secure. Then there exists a PPT algorithm A such that

$$\left|\Pr[(pk, sk) \leftarrow \mathsf{Gen}(1^k); b \leftarrow \{0,1\}; C \leftarrow \mathcal{E}_{pk}(b); b' \leftarrow A(pk, C) : b = b'] - \frac{1}{2}\right| \tag{1}$$

is not negligible. For simplicity, we simply assume $m_0 = 0$ and $m_1 = 1$ (recall we are working over a single-bit message space anyway, and the adversary cannot possibly succeed with better than half probability if $m_0 = m_1$).

Let the one-way trapdoor permutation that we are using for the encryption scheme be $\mathcal{F} = (\mathsf{Gen}_{td})$. With this trapdoor permutation we construct $\mathcal{F}' = (\mathsf{Gen}'_{td})$ with hard-core bit $\mathcal{H} = \{h_k\}$ as in Theorem 2; i.e., the hardcore bit for $f'(x|r) = f(x)|r$ is $h_k(x|r) = x \cdot r$. We know that for any PPT adversary A', the probability of guessing the hardcore bit $x \cdot r$ given $f_k(x)|r$ is negligible; that is, the following is negligible for any PPT A':

$$\left|\Pr\left[\begin{array}{c} (f', f'^{-1}) \leftarrow \mathsf{Gen}'_{td}(1^k); x \leftarrow \{0,1\}^k; \\ r \leftarrow \{0,1\}^k; y = f(x) \end{array} : A'(f', y|r) = x \cdot r\right] - \frac{1}{2}\right|. \tag{2}$$

Given A as above, our goal is to construct a PPT algorithm A' contradicting the above equation. We proceed as follows:

```
$A'(f', y|r)$
    $\alpha \leftarrow \{0,1\}$
    Define $pk = (f, r)$ and $C = (y|\alpha)$
    run $A(pk, C)$
    if the output of $A$ equals 0 then output $\alpha$
    else output the complement $\bar{\alpha}$
```

We may also rephrase the execution of A' as follows: it runs $A(pk, C)$ as above, and then outputs $\alpha \oplus A(pk, C)$ (this gives exactly the same output as above). Note also that A' runs in probabilistic polynomial time, assuming A does.

Let us first examine the intuition behind this construction of A'. We have $A(pk, C) = A((f, r), (y|\alpha))$; thus, if A always correctly guessed which message was encrypted, then A would always output $(f^{-1}(y) \cdot r) \oplus \alpha$, and hence A' would always output $f^{-1}(y) \cdot r$. The key thing to notice here is that $f^{-1}(y) \cdot r$ is the hardcore bit $h_k(x|r)$ of $f'(x|r)$ (where we let $x \overset{\text{def}}{=} f^{-1}(y)$). So, if A always succeeds (in "breaking" the encryption scheme) then A' always succeeds in guessing the hardcore bit. Of course, there is no reason to assume that A *always* succeeds and a formal proof is needed.

To complete our proof we need to massage Eq. (2) into Eq. (1). We are interested in the probability that A' correctly predicts the hard-core bit (this is just Equation (2), replacing f' by its definition in terms of f), i.e., we are interested in the following:

$$\left|\Pr[(f, f^{-1}) \leftarrow \mathsf{Gen}_{td}(1^k); x, r \leftarrow \{0,1\}^k; y = f(x) : A'(f, y|r) = x \cdot r] - \frac{1}{2}\right|.$$

Re-writing the above in terms of how A' was constructed, we obtain:

$$\begin{aligned}
&\left|\Pr[(f, f^{-1}) \leftarrow \mathsf{Gen}_{td}(1^k); x, r \leftarrow \{0,1\}^k; y = f(x) : A'(f, y|r) = x \cdot r] - \frac{1}{2}\right| \\
&= \left|\Pr\left[\begin{array}{c}(f, f^{-1}) \leftarrow \mathsf{Gen}_{td}(1^k); x, r \leftarrow \{0,1\}^k; \\ y = f(x); \alpha \leftarrow \{0,1\}\end{array} : A((f,r),(y|\alpha)) \oplus \alpha = x \cdot r\right] - \frac{1}{2}\right|.
\end{aligned}$$

Next, we modify the experiment syntactically by choosing a bit b at random and setting $\alpha = (x \cdot r) \oplus b$ (I will omit the parentheses from now on). Note, however, that from the point of view of A this is *exactly* equivalent to the above (because α is still uniformly distributed over $\{0,1\}$). Thus, we obtain (after some algebraic simplification):

$$\begin{aligned}
&\left|\Pr\left[\begin{array}{c}(f, f^{-1}) \leftarrow \mathsf{Gen}_{td}(1^k); x, r \leftarrow \{0,1\}^k; \\ y = f(x); \alpha \leftarrow \{0,1\}\end{array} : A((f,r),(y|\alpha)) \oplus \alpha = x \cdot r\right] - \frac{1}{2}\right| \\
&= \left|\Pr\left[\begin{array}{c}(f, f^{-1}) \leftarrow \mathsf{Gen}_{td}(1^k); x, r \leftarrow \{0,1\}^k; \\ y = f(x); b \leftarrow \{0,1\}; \alpha = x \cdot r \oplus b\end{array} : A((f,r),(y|\alpha)) \oplus \alpha = x \cdot r\right] - \frac{1}{2}\right| \\
&= \left|\Pr\left[\begin{array}{c}(f, f^{-1}) \leftarrow \mathsf{Gen}_{td}(1^k); x, r \leftarrow \{0,1\}^k; \\ y = f(x); b \leftarrow \{0,1\}\end{array} : A((f,r),(y|x \cdot r \oplus b)) = b\right] - \frac{1}{2}\right|. \quad (3)
\end{aligned}$$

Finally, let us look at the inputs given to A in the last expression above. The first input (f, r) is exactly a public-key for the encryption scheme under consideration. Furthermore, the second input $y|x \cdot r \oplus b$ given to A (with x, y and r chosen at random) is *exactly* a (random) encryption of the bit b with respect to the given public key. Thus, Equation (3) is exactly equal to Equation (1) (and hence Equation (2) is equal to Equation (1)). But we began by assuming that Equation (1) was non-negligible; this means that we have a particular PPT adversary A' for which Equation (2) is non-negligible. But this contradicts the assumed security of the trapdoor permutation (i.e., Theorem 2). ■

References

[1] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among notions of security for public-key encryption schemes. In *Adv. in Cryptology — CRYPTO 1998.*

[2] O. Goldreich and L. Levin, A hard-core predicate for all one-way functions, *Proc. 21st Ann. ACM Symp. on Theory of Computing*, 1989, pp. 25–32.

[3] S. Goldwasser and S. Micali, Probabilistic encryption, *Journal of Computer and System Sciences,* 28 (1984), pp. 270–299.