CS 600.442 – Modern Cryptography September 28, 2015

Lecture 8: Public-Key Encryption

Instructor: Abhishek Jain *Scribe: Rono Dasgupta*

1 Introduction

In the previous lecture, we discussed the concept of Secret-Key Encryption. However, this scheme has a major limitation in the fact that the key has to be shared between the sender and the receiver before any message can be transmitted. There might be several cases where doing this is infeasible if there has been no prior communication between the sender and the receiver. In today's lecture, we will look at an encryption scheme which provides a solution for this problem. This encryption scheme is called Public-Key Encryption. In this scheme, instead of a single shared secret key, the sender and the receiver now each have two keys - a public key and a secret key. The public key can be made public without giving away any information that can reduce the overall security of the scheme. The secret key remains secret, just like in Secret-Key Encryption. When a sender wants to send a message to the receiver, she uses the public key of the receiver to encrypt it. The receiver then uses his secret key to decrypt the message. There is no secret shared between the sender and the receiver in this scheme.

We will first define the informal goals of this scheme and gradually come up with a formal definition.

1.1 Goals:

Public Key: The encryption key is different from the decryption key in this scheme and the encryption key can be made public, that is, visible to everyone, including a potential adversary.

Correctness: The sender of the message can compute an encrypted cipher c of a message m using the public key of the receiver. The receiver can decrypt c to get back m using their secret key.

Security: An adversary $\mathcal{A}$ cannot distinguish between the encryptions of two different messages m and m', even with the public key.

We will now try to create a formal definition for Public-Key Cryptography using these goals.

Definition 1 *A public-key cryptographic scheme consists of three different algorithms, namely* ***Gen****,* ***Enc*** *and* ***Dec****. These can be defined as:*

- $\textbf{\textit{Gen}}(1^n) \to pk, sk$
- $\textbf{\textit{Enc}}(pk, m) \to c$
- $\textbf{\textit{Dec}}(sk, c) \to m'$

where pk stands for ***public key*** *and sk stands for* ***secret key****. All 3 algorithms run in polynomial time.*

1.2 Correctness:

The correctness of this public-key cryptography scheme can be defined as:

Correctness: $\forall m, Pr[\mathbf{Dec}(sk, \mathbf{Enc}(pk, m)) = m | (pk, sk) \leftarrow \mathbf{Gen}(1^n)] = 1$

In normal terms, this implies that decrypting an encrypted message with the correct pair of public and secret keys should **always** return the original message.

1.3 Security:

In order to define the security constraints for this scheme, we will first come up with a weaker indistinguishability definition and then move on to a stronger version of the same.

Weak Indistinguishability Security: We will define a security game to capture Weak-Indistinguishability w.r.t. Chosen Plaintext Attacks (Weak IND-CPA).

- First, the adversary chooses plaintexts m_0, m_1 s.t. $|m_0| = |m_1|$ and sends them to the challenger.
- Next, the challenger generates $(pk, sk) \leftarrow \mathbf{Gen}(1^n)$. It also chooses a random bit b and computes $c \leftarrow \mathbf{Enc}(pk, m_b)$. It sends pk, c to the adversary.
- The adversary outputs his guess b'. We say that the adversary wins the game if $b' = b$.

A public-key encryption scheme is said to be Weak IND-CPA secure if for every PPT adversary $\mathcal{A}$, there exists a negligible function $\mu(\cdot)$ s.t. the advantage of the adversary in winning the above game is at most $\mu(n)$, where n is the security parameter.

Strong Indistinguishability Security: We will define a security game to capture Strong Indistinguishability w.r.t. Chosen Plaintext Attacks (Strong IND-CPA)

- First, the challenger generates $(pk, sk) \leftarrow \mathbf{Gen}(1^n)$ and sends pk to the adversary.
- Next, the adversary chooses plaintexts m_0, m_1 s.t. $|m_0| = |m_1|$ and sends them to the challenger.
- The challenger chooses a random bit b and computes $c \leftarrow \mathbf{Enc}(pk, m_b)$. It sends c to the adversary.
- The adversary outputs his guess b'. We say that the adversary wins the game if $b' = b$.

A public-key encryption scheme is said to be Strong IND-CPA secure if for every PPT adversary $\mathcal{A}$, there exists a negligible function $\mu(\cdot)$ s.t. the advantage of the adversary in winning the above game is at most $\mu(n)$, where n is the security parameter.

In the future, we will refer to strong IND-CPA security as simply IND-CPA security for public-key

encryption. We note that the above definition of IND-CPA secure public-key encryption only considers security for a single message. We now show that for the case of public-key encryption, one message security implies many-message security.

Lemma 1 (Multi-Message Security) *One-message security implies multi-message security for public-key encryption.*

Intuition. The proof for the lemma follows easily by a standard hybrid argument. The key point is that in the public-key setting, we can easily reduce security of multiple messages to security of a single message – during the reduction, we can "embed" the challenge ciphertext at any position while still being able to create ciphertexts for other positions on our own using the public key (note that a similar strategy does not work in the secret-key case).

A corollary of the above lemma is that it is sufficient to construct a public-key encryption scheme for one-bit messages. Indeed, by using the above lemma, we can use an IND-CPA public-key encryption scheme for one-bit messages to obtain an IND-CPA public-key encryption scheme for multi-bit messages, by encrypting the long message bit-by-bit.

Our goal now is to construct an IND-CPA secure public-key encryption scheme for one bit messages. We will use the concept of trapdoor permutations for this, which we will see later in this lecture.

2 Collection of One-Way Functions

Definition 2 *A collection of one-way functions is a family $\mathcal{F} = \{f_i : \mathcal{D}_i \to \mathcal{R}_i\}_{i \in \mathcal{I}}$ which satisfies the following conditions:*

- ***Sampling function:*** *$\exists$ a Probabilistic Polynomial Time Generator such that $\boldsymbol{Gen}(1^n)$ outputs $i \in \mathcal{I}$.*
- ***Sampling from domain:*** *$\exists$ a Probabilistic Polynomial Time algorithm that on input i outputs a uniformly random element of $\mathcal{D}_i$.*
- ***Evaluation:*** *$\exists$ a Probabilistic Polynomial Time algorithm that on inputs i and $x \in \mathcal{D}_i$ outputs $f_i(x)$.*
- ***Hard to invert:*** *$\forall$ non-uniform Probabilistic Polynomial Time adversary $\mathcal{A}$, $\exists$ a negligible function $\mu(.)$ such that*

$$Pr[i \leftarrow \boldsymbol{Gen}(1^n), x \leftarrow \mathcal{D}_i, y \leftarrow f_i(x) : f_i(\mathcal{A}(1^n, i, y)) = y] \leq \mu(n)$$

Theorem: There exists a collection of One-Way Functions if and only if there exists a strong One-Way Function.

3 Trapdoor Permutations

For public-key encryption, we need permutations, specifically one-way permutations. A collection of one-way permutations can be defined as a collection $\mathcal{F} = \{f_i : \mathcal{D}_i \to \mathcal{R}_i\}_{i \in \mathcal{I}}$ where $\mathcal{F}$ is a collection of one-way functions and for every $i \in \mathcal{I}$, f_i is a permutation.

We now define a special type of one-way permutation called a trapdoor permutation. Very roughly, one-way permutation is a trapdoor permutation, when it is easy to compute the inverse of any image given a trapdoor.

Definition 3 *A collection of trapdoor permutations is a family of permutations* $\mathcal{F} = \{f_i : \mathcal{D}_i \to \mathcal{R}_i\}_{i \in \mathcal{I}}$ *which satisfies the following conditions:*

- ***Sampling function:*** $\exists$ *a Probabilistic Polynomial Time Generator* $\boldsymbol{Gen}_t(1^n)$ *that outputs* (f_i, t) *where* $i \in \mathcal{I}$.
- ***Sampling from domain:*** $\exists$ *a Probabilistic Polynomial Time algorithm that on input* i *outputs a uniformly random element of* $\mathcal{D}_i$.
- ***Evaluation:*** $\exists$ *a Probabilistic Polynomial Time algorithm that on inputs* i *and* $x \in \mathcal{D}_i$ *outputs* $f_i(x)$.
- ***Hard to invert:*** $\forall$ *non-uniform Probabilistic Polynomial Time adversary* $\mathcal{A}$, $\exists$ *a negligible function* $\mu(.)$ *such that*

$$Pr[i \leftarrow \boldsymbol{Gen}(1^n), x \leftarrow \mathcal{D}_i, y \leftarrow f_i(x) : f_i(\mathcal{A}(1^n, i, y)) = y] \leq \mu(n)$$

- ***Easy to invert given trapdoor:*** *There exists a Probabilistic Polynomial Time algorithm that when given* (i, t, y) *outputs* $f_i^{-1}(y)$.

4 Public-Key Encryption from Trapdoor Permutations

Let $\mathcal{F} = \{f_i : \mathcal{D}_i \to \mathcal{R}_i\}_{i \in \mathcal{I}}$ be a family of one-way trapdoor permutations. We construct a public-key encryption scheme (**Gen**,**Enc**,**Dec**) for one-bit messages.

- **Gen**(1^n): Compute $(f_i, {f_i}^{-1}) \leftarrow \mathbf{Gen}_t(1^n)$. Set $pk = f_i$, $sk = f_i^{-1}$.
- **Enc**(pk, m): Choose $r \xleftarrow{\$} \{0,1\}^n$ and compute ciphertext $c = f_i(r), m \oplus h_i(r)$, where h_i is the hardcore predicate for f_i.
- **Dec**(sk, c): Parse $c = c_1, c_2$. Output $h(f_i^{-1}(c_1)) \oplus c_2$

The correctness of the scheme is easy to verify. Here, we crucially rely on the fact that f is a permutation. To prove security, we need to show that encryptions of 0 are indistinguishable from encryptions of 1. This follows easily from the security of the hardcore predicate.

We consider the following hybrids:

H_1: This corresponds to encryption of 0, i.e., $f_i(r)$, $0 \oplus h_i(r)$ where r is randomly chosen.

H_2: This is similar to H_1, except that we now replace $h_i(r)$ with a random bit. That is, encryption now corresponds to $f_i(r)$, $0 \oplus R$, where R is a randomly chosen bit.

H_3: This is similar to H_2, except that we now replace 0 with 1, that is is encryption corresponds to $f_i(r)$, $1 \oplus R$, where R is a randomly chosen bit.

H_4: This corresponds to encryption of 1, i.e., $f_i(r)$, $1 \oplus h_i(r)$ where r is randomly chosen.

The indistinguishability of H_1 and H_2 follows from the security of hardcore predicate (proof left as an exercise). H_2 and H_3 are identical distributions, and the proof of indistinguishability of H_3 and H_4 follows in the same manner as that of H_1 and H_2.

5 Candidate Trapdoor Permutations

We will now look at an example of a one-way trapdoor permutation called RSA (named after its creators Rivest, Shamir, Adleman). The security of this scheme relies on the hardness of factoring. $\mathbb{Z}$

5.1 Definition (RSA Collection)

Definition 4 *RSA Collection can be defined as a family of permutations* $\mathcal{F} = \{f_i : \mathcal{D}_i \to \mathcal{R}_i\}_{i \in \mathcal{I}}$ *where:*

- $\mathcal{I} = \{(N, e) | N = p.q \text{ such that } p, q \in \pi_n, e \in \mathbb{Z}^*_{\Phi(N)}\}$
- $\mathcal{D}_i = \{x | x \in \mathbb{Z}^*_N\}$
- $\mathcal{R}_i = \mathbb{Z}^*_N$
- $\boldsymbol{Gen}(1^n) \to ((N, e), d)$ *where* $(N, e) \in \mathcal{I}$ *and* $e.d = 1 \ mod \ \Phi(N)$
- $f_{N,e}(x) = x^e \ mod \ N$
- ${f^{-1}}_{N,d}(y) = y^d \ mod \ N$

where $\mathbb{Z}^*_N$ *is an abelian group,* $\Phi(N)$ *is the order of* $\mathbb{Z}^*_N$.

When $N = p.q$, $\Phi(N) = (p-1)(q-1)$.

When $N = p$, $\Phi(N) = (p-1)$.

$\mathbb{Z}^*_N$ *is a family of permutations.*

d *is the multiplicative inverse of* e.

Then,

$$ {f^{-1}}_{N,d}(y) = y^d \bmod N $$

$$ (x^e \bmod N) = (x^e \bmod N)^d \bmod N $$

$$ = x^{e,d} \bmod N $$

$$ = x^{c\Phi(N)+1} \text{ ,where c is a constant.} $$

$$= x^{c\Phi(N)}x \mod N$$

$$= x \mod N, using Euler's Theorem which says that x^{\Phi(N)} = 1 \mod N.$$

Note: $ed = 1 \mod \Phi(N) \rightarrow ed = c\Phi(N) + 1$.

RSA Assumption: $\forall$ non-uniform Probabilistic Polynomial Time adversaries $\mathcal{A}, \exists$ a negligible function $\mu(.)$ such that

$$Pr[p, q \xleftarrow{\$} \pi_N, N = p.q, e \xleftarrow{\$} \mathbb{Z}^*_{\Phi(N)}, y \xleftarrow{\$} \mathbb{Z}^*_{(N)}; x \leftarrow \mathcal{A}(N, e, y) : x^e = y \mod N] \leq \mu(n)$$

Theorem: Assuming the RSA assumption, the RSA collection is a family of trapdoor permutations.