COM S 6830 – Cryptography October 20, 2011

Lecture 14: Public Key Encryption

Instructor: Rafael Pass *Scribe: Nick Alessi*

1 Single-Message CPA Gives Multi-Message Security

Let the encryption scheme (gen, enc, dec) be single message CCA2 secure. Assume that (gen, enc, dec) is not multi-message secure. Then $\exists\ \mathcal{A} \in nuPPT$ that picks $\vec{m} = (m_1, \ldots, m_{q(n)})$ and $\vec{m}' = (m'_1, \ldots, m'_{q(n)})$, and can distinguish the following:

$$\{k \leftarrow gen(1^n) : enc_k(m_1), \ldots enc_k(m_{q(n)})\}$$
$$\{k \leftarrow gen(1^n) : enc_k(m'_1), \ldots enc_k(m'_{q(n)})\}$$

Then define a series of hybrids H_i by the messages $c_i = m_1, \ldots m_i, m'_{i+1}, \ldots m_{q(n)}$. So by the hybrid lemma some i exists such that H_i and H_{i+1} are distinguishable. However, we could define the CPA attacker $\mathcal{A}'$ by the system that knows the correct i, and stores the $\vec{m}$ and $\vec{m'}$ output by $\mathcal{A}$. It then chooses the messages that it will distinguish to be m_i and m'_i. Then it requests $enc(m_1), \ldots enc(m_i$ and $enc(m'_{i+2}), \ldots enc(m'_{q(n)})$. Now it returns its choice of messages to distinguish as m_i and m'_i. Whatever coded message c it gets it passes $\mathcal{A}$ the multiple messages $enc(m_1), \ldots enc(m_i, c, enc(m'_{i+2}), \ldots enc(m'_{q(n)})$. Since $\mathcal{A}$ distinguishes H_i and H_{i+1}, $\mathcal{A}'$ distinguishes the two messages using a CPA attack. Thus we have a contradiction, so single-message CPA secure implies multi-message secure.

2 Public Key Encryption

The motivation of public key encryption is that Alice and Bob have never met, but they want to securely comunicate.

2.1 An Incorrect Proof of Public Key Impossibility

To decrypt a message you need the key or else anyone could just decrypt any message. To encrypt a message the key or else the coded message will not be correlated to the key,

or else decryption would be impossible. Thus both the sender and receiver need the key and public key encryption is impossible.

This proof is faulty because it only implies that both parties need correlated information. It does not mean that both must have the same information. By making a public and a secret key it is possible to share correlated information without needing both to have the same information.

2.2 Public Key Encryption Scheme Definition

(gen, enc, dec) is a public key encryption scheme if:

- gen is a PPT key generator outputting (pk, sk)
- enc is a PPT such that $c \leftarrow enc_{pk}(m)$
- dec is a PPT such that $m \leftarrow dec_{sk}(c)$

Then require the condition $\forall m \in \{0,1\}^*$ and $\forall (pk, sk) \in gen(1^{|m|})$ we want

$$\Pr\left[dec_{sk}(enc_{pk}(m)) = m\right] = 1$$

Now we want to define what it means for a public key encryption scheme (gen, enc, dec). It is secure if $\forall \mathcal{A} \in nuPPT\ \exists \epsilon \in neg$ and $\forall n \in \mathbb{N}\ \forall m_0, m_1 \in \{0,1\}^n$ $\mathcal{A}$ distinguishes the following distributions with probability $\leq \epsilon(n)$:

$$\{(pk, sk) \leftarrow gen(1^n); (pk, enc_{pk}(m_0)\}_{n\in\mathbb{N}}$$
$$\{(pk, sk) \leftarrow gen(1^n); (pk, enc_{pk}(m_1)\}_{n\in\mathbb{N}}$$

3 Trapdoor Permutations

3.1 RSA Collection

Recall that the RSA collection is a collection of OWF's such that $gen(1^n) \to (N = pq, e \in \mathbb{Z}_N^*$. Then $f_{(N,e)}(x) = x^e$. However if we know $d = e^{-1} \mod \varphi(n)$ then $(x^e)^d = x^{ed} = x$. Then d serves as a trapdoor which makes reversing the OWF's easy.

3.2 Definition of a Trapdoor Permutation

$F = \{f_i : \mathcal{D}_i \to \mathcal{R}_i\}_{i \in I}$ is a collection of trapdoor permutations if: $\forall i \in I$ f_i is a permutation and

- $\exists gen \in PPT$ such that $gen(1^n) \to (i, t)$, $i \in I$.
- $\exists PPT$ that efficiently samples uniformly from $\mathcal{D}_i$ given i
- $f_i(x)$ is computable in PPT
- $\forall \mathcal{A} \in nuPPT$ $\exists \epsilon \in neg$ such that $\forall n \in \mathbb{N}$

$$\Pr\left[(i, t) \leftarrow gen(1^n), x \in \mathcal{D}_i, \mathcal{A}(i, x) \in f_i^{-1}(x)\right] \leq \epsilon(n)$$

- $\exists \mathcal{B} \in PPT$ such that $\forall (i, t) \in gen(1^n)$, $\forall x \in \mathcal{D}_i$, $f_i(x) = y$ then $\mathcal{B}(i, y, t) = f_i^{-1}(y) = x$

3.3 RSA Collection as a Collection of Trapdoor Permutations

We have $(i, t) \leftarrow gen(1^n)$ with $i = (N, e)$ and $t = d$. Then $f_i(x) = x^e \mod N$, and $f_i^{-1}(y) = y^d \mod N$. We get the conditions of OWF because the RSA collection is a collection of OWF's. The trapdoor obviously inverts f_i uniquely, so this is a collection of trapdoor permutations.

4 Creating a Public Key Encryption System

4.1 Naive RSA

We can then create a simple encryption scheme based on the RSA collection being a collection of trapdoor permutations. Just take the definition from above and then $i = (N, e) = pk$ and $t = d = sk$. However this may not work because a OWF exists that leaves half the bits known (see homework 2 #4a). So constructing things like this does not necessarily work.

4.2 Padded RSA

A first attempt to fix this problem would be to define *gen* as in naive RSA. Then define $enc_{pk}(m) = r \leftarrow \{0,1\}^n, f_{pk}(m||r)$. Then $dec_{sk}(c) = m||r = f_{sk}^{-1}(c), m$. This does better because there is extra randomness that would make decryption harder without the secret key.

4.3 Securely Sending a Single Bit

Let f be a trapdoor permutation and h be a hard-core predicate for f (since f is also a OWF we can get such a predicate by adjusting f). Then to securely send 1 bit b we make $pk = i$ public and $sk = t$ is kept secret. So the sender first selects $r \leftarrow \{0,1\}^n$ and then $enc_{pk}(b) = f_{pk}(r), m \oplus h(r)$. Then decrypting uses the trapdoor to get r then the last bit xor $h(r)$ is b the message bit.