Lecture #8

Public-key Encryption

2024-11-13

1 Public-Key Encryption

Private key encryption needs an identical pair of keys to be secure. Thus we need a secure method to generate this pair of keys. A direct attempt is to generate it in one side and try to send it in the presence of an eavesdropper, which is called key-exchange.

Consider the following scheme $\Pi = (\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})$

- $\mathrm{Gen}(1^\lambda) \to (pk, sk)$
- $\mathrm{Enc}(pk, m) \to c$
- $\mathrm{Dec}(sk, c) \to m$

that is correct (i.e. $\mathrm{Dec}(sk, \mathrm{Enc}(pk, m)) = m$). If this scheme satisfies that, for any p.p.t. adversary $\mathcal{A}$, the following game satisfies $\Pr[\mathrm{PubK}^{\mathrm{eav}}_{\mathcal{A},\Pi} \to 1] \leq \frac{1}{2} + \mathrm{negl}(\lambda)$ then we say Π is secure, or has indistinguishable encryptions:

1. The challenger samples $(sk, pk) \leftarrow \mathrm{Gen}(1^\lambda)$ and send pk to $\mathcal{A}$
2. $\mathcal{A}$ send a pair of message (m_0, m_1) with same length to the challenger
3. The challenger samples $b \leftarrow \{0, 1\}$ and send $\mathrm{Enc}(pk, m_b)$ to $\mathcal{A}$
4. $\mathcal{A}$ guess b' and win if $b' = b$

As the encryption key pk is made public, any secure public-key encryption is automatically CPA-secure.

Additionally, we define the CCA-security on Π by allowing $\mathcal{A}$ to have oracle access of $\mathrm{Dec}(sk, \cdot)$ on any ciphertext except the challenging one given to $\mathcal{A}$.

2 Key Encapsulation

With a secure key-exchange scheme Π*, we can combine* Π *with a secure private-key encryption scheme to encrypt arbitrary length messages, as known as the "encapsulation" trick.*

2.1 Encapsulation Trick

Consider the following scheme $\Pi = (\mathrm{Gen}, \mathrm{Encap}, \mathrm{Decap})$

- $\text{Gen}(1^\lambda) \to (pk, sk)$
- $\text{Encap}(pk) \to \{k, \text{cap}(k)\}$
- $\text{Decap}(sk, \text{cap}(k)) \to k$

that is correct (i.e. $\text{Decap}(sk, \text{cap}(k)) = k$ for any $k, \text{cap}(k)$ generated by Encap). If this scheme satisfies that, for any p.p.t. adversary $\mathcal{A}$, the following game satisfies $\Pr[\text{PubK}^{\text{encap}}_{\mathcal{A},\Pi} \to 1] \leq \frac{1}{2} + \text{negl}(\lambda)$ then we say Π is secure:

1. The challenger sample $(sk, pk) \leftarrow \text{Gen}(1^\lambda)$ and send pk to $\mathcal{A}$
2. The challenger generate $(k, \text{cap}(k)) \leftarrow \text{Encap}(pk)$ and send $\text{cap}(k)$ to $\mathcal{A}$
3. The challenger sample a random bit $b \leftarrow \{0, 1\}$, and send k to $\mathcal{A}$ if $b = 0$, or a random value r otherwise
4. $\mathcal{A}$ guess b' and win if $b' = b$

Also, we define the CCA-security by giving oracle access of $\text{Decap}(sk, \cdot)$ to the adversary $\mathcal{A}$ on any input except the $\text{cap}(k)$ given.

2.2 Construct Key Encapsulation Using Trapdoor Permutation

A *trapdoor permutation* is a pair of (f, t) generated by some efficient algorithm $\text{Gen}(1^\lambda)$ such that

- $f : \mathcal{D} \to \mathcal{D}$ where $\mathcal{D}$ is similar to $\{0, 1\}^\lambda$
- There exists an efficient algorithm invert based on Gen that $\text{invert}(t, f(x)) \to x$
- For any p.p.t. adversary $\mathcal{A}$, $\Pr[\mathcal{A}(f, f(x)) \to x] \leq \text{negl}(\lambda)$

With a trapdoor permutation, we can build a key encapsulation protocol

- $\text{Gen}(1^\lambda) \to (f, t)$
- $\text{Encap}(f) \to (h(k), c = f(k))$ where k is uniformly sampled from $\mathcal{D}$
- $\text{Decap}(t, c) \to h(\text{invert}(t, c))$

with some hardcore function h whose candidate may be

- a hardcore bit of f
- a hardcore function of f
- a random oracle

while in the first two cases the Encap and Decap process may need to run multiple times to carry long enough keys, and they're not CCA-secure.

2.3 Key Encapsulation to Public-Key Encryption

With a CPA-secure key encapsulation and a CPA-secure private-key encryption, we can construct a CPA-secure public-key encryption by attaching the value $\text{Enc}(k, m)$ after the result of $\text{Encap}(pk)$, where k is the encapsulated key.

Consider the real world W and the hybrid world W' in which Alice and Bob has a pair of pre-generated keys r and use r to encrypt their data. In W' the adversary has $\text{negl}(\lambda)$ advantage (as this is a CPA-secure private-key encryption), and by the security of key encapsulation, $\Pr[W' \to 1]$ and $\Pr[W \to 1]$ should also differ within $\text{negl}(\lambda)$.

Similar argument can be used to prove CCA-security of this hybrid.

3 Discrete Log and Diffie-Hellman

In real world secure encryption schemes, we must make some relatively reasonable assumptions on the computational difficulty of some specific problems. One among these is discrete log, which introduces Diffie-Hellman and its variants.

3.1 Discrete Log and Diffie-Hellman Assumptions

Dlog Assumption: (*Finding discrete log is hard.*) Given a group G with a generator g sampled by some algorithm Gen and an element $g^t \in G$, finding t is difficult, i.e. for any p.p.t. adversary $\mathcal{A}$

$$\Pr_{(G,g)\leftarrow\text{Gen}(1^\lambda),t\leftarrow\{1,2,\ldots,|G|\}}[\mathcal{A}(G, g, g^t) \to t] \leq \text{negl}(\lambda)$$

Based on Dlog assumption, we have two variants of Diffie-Hellman assumption.

Computational Diffie-Hellman Assumption: For any p.p.t. adversary $\mathcal{A}$, $\Pr[\mathcal{A}(g, g^x, g^y) \to g^{xy}] \leq \text{negl}(\lambda)$.

Decisional Diffie-Hellman Assumption: $(G, g, g^x, g^y, g^{xy}) \approx_c (G, g, g^x, g^y, g^r)$. i.e. any p.p.t. distinguisher $\mathcal{D}$ can't distinguish them with noticeable advantage.

3.2 Break Discrete Log with Specific Primes

If all the factors of $p - 1$ are small, then the discrete log on $\mathbb{Z}_p^*$ can be calculated efficiently using Chinese Remainder Theorem.

For example, let $p = 2 \times 3 \times 5 \times 7 + 1 = 211$, which is a prime. The group $\mathbb{Z}_p^*$ is isomorphic with $\mathbb{Z}_2 \times \mathbb{Z}_3 \times \mathbb{Z}_5 \times \mathbb{Z}_7$, which means both g^{30} and g^{30t} are 0 on the first 3 dimensions, and $x = t \mod 7$ is the only solution in $\mathbb{Z}_7$ that $g^{30x} = g^{30t}$ on the last dimension. Thus we can guess $t \mod 7$ in 7 guesses. The same trick can then be applied to guess $t \mod 2, 3, 5$ and use CRT to recover t.

Table 3-1: A visualized chart of the argument above

projection	$\mathbb{Z}_2$	$\mathbb{Z}_3$	$\mathbb{Z}_5$	$\mathbb{Z}_7$
g	g_2	g_3	g_5	g_7
g^t	tg_2	tg_3	tg_5	tg_7
g^{30}	0	0	0	$30g_7$
g^{30t}	0	0	0	$30tg_7$

3.3 Diffie-Hellman Based Encryption

Based on Diffie-Hellman assumption, we can construct a encapsulation scheme

- $\text{Gen}(1^\lambda) \to (pk = (G, g, g^x), sk = x)$ where x is uniformly sampled from $\mathbb{Z}^*_{|G|}$
- $\text{Encap}(pk) \to (h(g^{xy}), c = g^y)$ where y is uniformly sampled from $\mathbb{Z}^*_{|G|}$
- $\text{Decap}(sk, c) \to h(c^x = g^{xy})$ recovers the encapsulated key

in which we need a random oracle h to achieve CCA-security.

And we can construct a similar encryption scheme

- $\text{Gen}(1^\lambda) \to (pk = (G, g, g^x), sk = x)$ where x is uniformly sampled from $\mathbb{Z}^*_{|G|}$
- $\text{Enc}(pk, m) \to (g^y, h(g^{xy}) \cdot m)$ where y is uniformly sampled from $\mathbb{Z}^*_{|G|}$
- $\text{Dec}(sk, (c_k, c_m)) \to (h(c_k^x))^{-1} \cdot c_m$

that is CCA-secure with random oracle h under the DDH assumption.

4 RSA-based Encryption

RSA was invented based on the difficulty of factorization.

4.1 Intuition

Factorization problem: Given $N = pq$ where p, q are primes, find p, q.

Assumption: (*Factorization is hard.*) For any p.p.t. adversary $\mathcal{A}$

$$\Pr_{p,q \leftarrow \text{primes}(\lambda)}[\mathcal{A}(pq) \to (p, q)] \leq \text{negl}(\lambda)$$

Consider the group $\mathbb{Z}^*_N$. The size of this group is $\varphi(N) = (p-1)(q-1)$.

Conditioning on $N = pq$, calculating $\varphi(N)$ is as hard as factorizing N, as $p + q = N - \varphi(N) + 1$ and we can solve (p, q) using a quadratic equation.

4.2 Plain RSA

RSA Assumption: Given (N, e, m^e), then m is hard to find, i.e. For any p.p.t. adversary $\mathcal{A}$

$$\Pr_{p,q\leftarrow\text{prime}(\lambda),N=pq}[\mathcal{A}(N,e,m^e)\to m]\leq \text{negl}(\lambda)$$

Strong RSA Assumption: Given (N, y), then non-trivial pair (m, e) such that $m^e = y$ is hard to find, i.e. For any p.p.t. adversary $\mathcal{A}$

$$\Pr_{p,q\leftarrow\text{prime}(\lambda),N=pq}[\mathcal{A}(N,y)\to (m,e)\wedge e\neq 1\wedge m^e=y]\leq \text{negl}(\lambda)$$

Both two assumptions above can derive a weaker variant by replacing prime with safe-prime.

The intuition of introducing strong RSA assumption is to remove the randomness requirement on e.

Consider the following protocol $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ where

- $\text{Gen}(1^\lambda)\to (pk=(N=pq,e), sk=d=e^{-1} \mod \varphi(N))$
- $\text{Enc}(pk, m)\to m^e \mod N$
- $\text{Dec}(sk, c)\to c^d \mod N$

which is a plain RSA scheme. It's insecure as Enc is deterministic.

4.3 Encapsulation RSA

The following protocol $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ where

- $\text{Gen}(1^\lambda)\to (pk=(N=pq,e), sk=d=e^{-1} \mod \varphi(N))$
- $\text{Enc}(pk, m)\to (r^e \mod N, h(r)\oplus m)$ where r is uniformly sampled from $\mathbb{Z}_N^*$
- $\text{Dec}(sk, (c_k, c_m))\to h(c_k^d \mod N)\oplus c_m$

(with a random oracle h) is an encapsulation RSA scheme.

In real life, we can't directly use random oracles; instead we can let h be a hardcore bit such as $h(r) := \text{lsb}(r)$, where lsb denotes *least significant bit* (i.e. the value of $r \mod 2$).

However, a problem is that this scheme is only capable of encrypting 1 *bit of message. The following argument gives a solution to this problem.*

Suppose that lsb is a hardcore bit of $f(r) = r^e$. Consider the following process

$$\begin{array}{ccccc} r & \to & c^{d^\lambda} & \to & b_0 = \mathrm{lsb}(c^{d^\lambda}) \\ r^e & \to & c^{d^{\lambda-1}} & \to & b_1 = \mathrm{lsb}(c^{d^{\lambda-1}}) \\ r^{e^2} & \to & c^{d^{\lambda-2}} & \to & b_2 = \mathrm{lsb}(c^{d^{\lambda-2}}) \\ \vdots & & \vdots & & \vdots \\ r^{e^{\lambda-1}} & \to & c^d & \to & b_{\lambda-1} = \mathrm{lsb}(c^d) \\ r^{e^\lambda} & \to & c & & \end{array}$$

By hybrid argument (replacing $b_0, b_1, b_2, \ldots, b_{i-1}$ with real random in the i-th hybrid), $\{b_i\}$ is close to uniform when c is known. i.e. $\mathrm{lsb}(r)$ is a hardcore bit of $f(r) = r^e$ implies $h'(r) := \{b_0, b_1, \ldots, b_{\lambda-1}\}$ is a hardcore function of $f'(r) = r^{e^\lambda}$.

5 Rabin Encryption

Many other encryption schemes used are also based on the difficulty of factorization.

5.1 Intuition

Consider the quadratic residue group $\mathbb{QR}_p = \{a^2 | a \in \mathbb{Z}_p^*\} \cong \mathbb{Z}_{p'}$ for safe prime $p = 2p' + 1$.

For $N = pq$, where $p = 2p' + 1$ and $q = 2q' + 1$, define the Jacobi Symbol of $a \in \mathbb{Z}_N^*$ as

$$\left(\frac{a}{N}\right) = \left(\frac{a}{p}\right)\left(\frac{a}{q}\right)$$

and the Jacobi Symbol of $a \in \mathbb{Z}_p$ is

$$\left(\frac{a}{p}\right) = \begin{cases} 0 & a = 0 \\ 1 & \exists x \in \mathbb{Z}_p^*, a = x^2 \\ -1 & \text{otherwise} \end{cases}$$

Jacobi Symbol can be efficiently calculated.

Define $\mathbb{QR}_N := \{a \in \mathbb{QR}_p \wedge a \in \mathbb{QR}_q | a \in Z_N^*\}$, and pseudo-$\mathbb{QR}_N := \{a \notin \mathbb{QR}_p \wedge a \notin \mathbb{QR}_q | a \in Z_N^*\}$

The Jacobi Symbol of $a \in \mathbb{Z}_N^*$ is 1 if and only if a is in $\mathbb{QR}_N$ or pseudo-$\mathbb{QR}_N$.

Assumption: $\mathbb{QR}_N$ and pseudo-$\mathbb{QR}_N$ are indistinguishable.

5.2 Rabin Encryption

Based on the assumptions above, we have the Rabin Encryption described as $\Pi = (\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})$ below:

- $\mathrm{Gen}(1^\lambda) \to (pk = (N = pq, g \in \text{pseudo-}\mathbb{QR}_N), sk = (p, q))$

- $\mathrm{Enc}(pk, m) \to r^2 \cdot g^m$ where r is uniformly sampled from $\mathbb{Z}_N^*$
- $\mathrm{Dec}(sk, c)$ checks if c is in $\mathbb{QR}_N$, which indicates $m = 0$ and otherwise $m = 1$.

Sadly, this scheme is also only capable of encrypting 1 *bit of message.*