Public key cryptography (asymmetric cryptography)

Review

- Secret-key cryptography (symmetric cryptography)
  - Shift cipher, substitution cipher, vigenere cipher, DES
  - Use the same key for both encryption & decryption (Z=Z')
  - Key must be kept secret
  - Weakness
    - Managing and distributing shared secret keys is so difficult in a model environment with too many parties and relationships
    - N parties ➔ n(n-1)/2 relationships ➔ each manages (n-1) keys
    - No way for digital signatures
      - No non-repudiation service

Diffie-Hellman new ideas for PKC

- In principle, a PK cryptosystem is designed for a single user, not for a pair of communicating users
  - More uses other than just encryption
- Proposed in Diffie and Hellman (1976) “New Directions in Cryptography”
  - public-key encryption schemes
  - public key distribution systems
    - Diffie-Hellman key agreement protocol
  - digital signature

Diffie-Hellman's proposal

- Each user creates 2 keys: a secret (private) key and a public key → published for everyone to know
  - The PK is for encryption and the SK for decryption
    $$X = D(z, E(Z, X))$$
  - The SK is for creating signatures and the PK for verifying these signatures
    $X = E(Z, D(z, X))$ → $D()$ for creating signatures, $E()$ → verifying
- Also, called asymmetric key cryptosystems
  - Knowing the public-key and the cipher, it is computationally infeasible to compute the private key

Principles for creating a PKC

- A PKC can be created based on a one-way (1 chiều), which satisfies the following properties
  - for all X, it is easy to compute Y = f(X)
  - But, it is almost computationally impossible to inversely determine X given Y
- Example: Let $p_1, p_2, \ldots p_n$ be primes →
  - it is easy to compute $n = p_1 \times p_2 \times \cdots \times p_n$
  - But, given n, it is difficult to factorize n into the product of $p_1 \times p_2 \times \cdots \times p_n$
- To build a PKC → we need a special one-way function, which has a so-called trap-door
  - if someone knows the trap-door, they can easily determine X given $f(X)$
  - otherwise, it is impossible to determine X from $f(X)$
- How to build a PKC from a one-way function with having trap ?
  - the encoding function E_z is the one-way function with the trap-door
  - Trap-door is the secret key
  - if one has the secret key → can decrypt the cipher text

PKC based on the Knapsack problem

- 1978, Merkle – Hellman proposed a PKC based on the knapsack problem as follows
- knapsack problem:
  - given a set of a_i, $1 \le i \le n$, and a positive number T.
  - find the indices $S \subset \{1,2,\ldots,n\}$ such that : $\sum_{i \in S} a_i = T$

→ this is a one-way function

- for the inverse problem, we need to perform the brute-force with the complexity of exponentiation of

- ex: $(a_1, a_2, a_3, a_4) = (2, 3, 5, 7)$ T = 7.

  → we have two solutions S = (1, 3) và S = (4).

Merkle – Hellman PKC

- what if a_i satisfy a_i so-called super-increasing property?
  - $a_{i+1} > a_1 + \cdots + a_i$
- → the inverse problem can be easily done

  ex: a=(1,2,4,8)

  T=11,

  $T=T_0$

$X_4=1$	$T_0=T_0-X_4=3$	→ $(X_1\ X_2\ X_3\ 1)$
$X_3=0$	$T_2=T_1=3$	→ $(X_1\ X_2\ 0\ 1)$
$X_2=1$	$T_3=T_2-2=1$	→ $(X_1\ 1\ 0\ 1)$
$X_1= 1$		→ (1 1 0 1)

- $a_1 < a_2 < \cdots < a_n$
- $a_i > a_{i-1} + \cdots + a_1$
- $T = \sum a_{k_1} + a_{k_2} + \cdots + a_{k_m}$
- $a_{q_1} | \, a_{q_1} < T < a_{q_1+1}$
- $T = a_{q_1} + (T \text{-} \, a_{q_1})$
- $T = a_q + \cdots ..$
- $a_{q_2} < T - a_{q_1} < a_{q_2+1}$
- $a_1 < a_2 < \cdots < a_{q_2} < a_{q_2+1} < \cdots < a_{q_1} < \cdots$
- $T = a_{q_1} + a_{q_2} + \cdots$

Merkle – Hellman PKC

- Idea of the Merkel-Hellman PKC
  - Choose a so-called cargo vector a = $(a_1, a_2, \ldots, a_n)$
  - encryption
    - for a plain text $X = (X_1,X_2,X_3 \ldots, X_n)$ → encrypt by $T= \sum a_iX_i$ (*)
  - decryption
    - for a cipher text T, and the cargo vector a, determine X_i satisfying (*).
- → decryption is a one-way function ➔ we need to design a trap-door

Merkle – Hellman PKC

How to hide the trap-door

- *key generation*:

  Alice chooses a super-increasing vector :

  $a' = (a_1', a_2', \ldots, a_n')$

  a' is kept as a part of the secret key

  - Alice chooses
    - $m > \sum a_i'$, named as the congruent modulothe
    - a random integer ω, named as gọi là multiplier, which is co-prime with *m*.
  - public key is $a = a'.\omega$

    $a = (a_1, a_2, \ldots, a_n)$

    $a_i = \omega \times a_i' \pmod{m}; \ i=1,2,3 \ldots n$
  - secret key: (a', m, ω)

Merkle – Hellman PKC Details

- *Encryption:*
  - When Bob wants to send a message X to Alice, he encrypted X :

$$T = \sum a_i X_i$$

- *Decryption*:
  - When Alice receives T, she decrypts as follows:

  she first calculates ω^{-1} satisfying gaht $\omega \times \omega^{-1} = 1\ (mod\ m)$,

  then determines $T' = T \times \omega^{-1} (mod\ m)$

  - Alice knows that $T' = a'.X$, and because a' is a super-increasing vector, Alice can determine X given T' and a'

- Note

$$T' = T \times \omega^{-1} = \sum a_i X_i \omega^{-1} = \sum a_i{}' \omega X_i \omega^{-1}$$
$$= \sum (a_i{}' \omega \omega^{-1}) X_i = \sum a_i{}' X_i = a'.X$$

RSA Algorithm

- Invented in **1978** by Ron **R**ivest, Adi **S**hamir and Leonard **A**dleman
  - Published as R L Rivest, A Shamir, L Adleman, "*On Digital Signatures and Public Key Cryptosystems*",Communications of the ACM, vol 21 no 2, pp120-126, Feb 1978
  - Security relies on the difficulty of factoring large composite numbers

Main idea

- Encryption and decryption functions are modulo exponential in the field $Z_n = \{0,1,2,..n-1\}$
  - Encryption : $Y \equiv X^e \ (mod\ n)$
  - Decryption: $X \equiv Y^d \ (mod\ n)$
  - The clue is that e & d must be selected such that
    - $X^{ed} \equiv X \ (mod\ n)$

Main idea

- Euler theorem: $X^{\varphi(n)} \equiv 1\ (mod\ n)$
  - $\varphi(n)$: the number of $k: 0 < k < n |\ gcd(k, n) = 1$
  - If $n = p \times q$ (p, q are primes) → $\varphi(n) = (p-1)(q-1)$
- First choose e then compute d s.t. $ed \equiv 1(mod\ \varphi(n))$
  - $d \equiv e^{-1}(mod\ \varphi(n))$
  - $X^{ed} \equiv X^{k\varphi(n)+1} \equiv (X^{\varphi(n)})^k \times X \equiv X\ (mod\ n)$
- Note this works because we know n's factorization
  - From e we compute $d \equiv e^{-1}\ mod\ \varphi(n)$ since we know $\varphi(n)$, otherwise it is computational infeasible to compute d s.t. $X^{ed} \equiv$ $mod\ n$

RSA public key cryptography

- **Key generation:**
  - Select 2 large prime numbers of about the same size, p, q
  - Compute $n = pq$, and $\varphi(n) = (q-1)(p-1)$
  - Select a random integer e, $1 < e < \varphi(n)$, s.t. $\gcd(e, \varphi(n)) = 1$
  - Compute d, $1 < d < \varphi(n)$ s.t. $ed \equiv 1 \ mod \ \varphi(n)$
  - **Public key: $(\boldsymbol{e}, \boldsymbol{n})$ and Private key: $\boldsymbol{d}$**
    - **Note: $\boldsymbol{p}$ and $\boldsymbol{q}$ must remain secret**

RSA public key cryptography

- **Encryption**
  - Given a message $M, 0 < M < n$
  - Use public key (e, n) compute :

    $C = M^e \ (mod\ n)$
- **Decryption**
  - Given a ciphertext C, use private key (d) và compute:
    - $M = C^d \ (mod\ n)$
- Why work?
  - $C^d \ (mod\ n) \equiv M^{ed} (mode\ n) \equiv M (mod\ n)$

Example

- Parameters:
  - Select $p = 11$ và $q = 13$
  - $n = 11 * 13 = 143; m = (p-1)(q-1) = 10 * 12 = 120$
  - Choose $e = 37$ ➔ $\gcd(37,120) = 1$
  - Find d such that: $e \times d \equiv 1\ (mod\ 120)$ ➔ $d = 13\ (e \times d = 481)$
- To encrypt a binary string
  - Split it into segments of u bits, $2^u \le 142$ ➔ $u = 7$
    - each segment presents a number from 1 to 127
  - Compute $Y = X^e (mod\ n)$

  E.g.: for $X = (0000010) = 2$, we have $Y \equiv X^{37} \equiv 12\ (mod\ 143) \rightarrow Y = (00001100)$
- Decryption : $X \equiv 12^{13}\ (mod\ 143) = 2 \rightarrow X = 0000010$

RSA implementation

- n, p, q
  - The security of RSA depends on how large n is, which is often measured in the number of bits for n. Current recommendation is 1024 bits for n.
  - p and q should have the same bit length, so for 1024 bits RSA, p and q should be about 512 bits.
  - $p - q$ should not be small
  - Way to select p and q
    - In general, select large numbers (some special forms), then test for primality
    - Many implementations use the Rabin-Mille test, (probabilistic test)

Modular multiplicative inverse

- Bézout lemma:
  - Let a and b be integers with greatest common divisor d. Then, there exist integers x and y such that $ax + by = d$. More generally, the integers of the form $ax + by$ are exactly the multiples of d
- Diophantine equation: $ax+by=c$
  - This equation has solution if and only if $c \vdots \gcd(a, b)$
- If $1 = GCD(e, n) \rightarrow 1 = xe + yn \rightarrow xe \equiv 1 (mod\ n) \rightarrow x \equiv e^{-1} (mod\ n)$

Modular multiplicative inverse

- Euclidean algorithm for determining $\text{GCD}(r_0, r_1)$

$$
\begin{aligned}
r_0 &= q_1 r_1 + r_2, & 0 < r_2 < r_1 \\
r_1 &= q_2 r_2 + r_3, & 0 < r_3 < r_2 \\
&\vdots \\
r_{m-2} &= q_{m-1} r_{m-1} + r_m, & 0 < r_m < r_{m-1} \\
r_{m-1} &= q_m r_m .
\end{aligned}
$$

  - It can be proved that: $\gcd(r_0, r_1) = \gcd(r_1, r_2) = \cdots = \gcd(r_{m-1}, r_m) = r_m$

Modular multiplicative inverse

- Example
  - Determine gcd(252, 198)

$$
\begin{aligned}
252 &= 198 \times 1 + 54 \\
198 &= 54 \times 3 \;\; + 36 \\
54 \;\; &= 36 \times 1 \;\; + 18 \\
36 \;\; &= 18 \times 2 \;\; + 0
\end{aligned}
$$

Gcd(252, 198) = 18

Modular multiplicative inverse

- Example
  - Solve: $252x+198y=18$

$$
\begin{aligned}
252 &= 198 \times 1 + 54 \\
198 &= 54 \times 3 + 36 \\
54 &= 36 \times 1 + 18 \\
36 &= 18 \times 2 + 0
\end{aligned}
$$

$$
\begin{aligned}
18 &= 54 - 36 \\
18 &= 54 - (198 - 54 \times 3) \\
18 &= 54 \times 4 - 198 \\
18 &= (252 - 198) \times 4 - 198 \\
18 &= 252 - 198 \times 5
\end{aligned}
$$

(x, y) = 1, -5

Modular multiplicative inverse

- Example
  - Solve: $252x+198y=18$

$$
\begin{aligned}
252 &= 198 \times 1 + 54 \\
198 &= 54 \times 3 + 36 \\
54 &= 36 \times 1 + 18 \\
36 &= 18 \times 2 + 0
\end{aligned}
$$

$$
\begin{aligned}
18 &= 54-36 \\
18 &= 54-(198-54\times 3) \\
18 &= 54\times 4-198 \\
18 &= (252-198)\times 4-198 \\
18 &= 252-198\times 5
\end{aligned}
$$

$(x, y) = 1, -5$

Modular multiplicative inverse

- Example
  - Determine $28^{-1}\ mod\ 75$
  - Correspond to solving equation $28x + 75\ y = 1$

$$75 = 28 \times 2 + 19$$
$$28 = 19 \times 1 + 9$$
$$19 = 9 \times 2 + 1$$

$$1 = 19 - 9 \times 2$$
$$1 = 19 - (28 - 19 \times 1) \times 2 = -28 \times 2 + 19 \times 3$$
$$1 = -28 \times 2 + (75 - 28 \times 2) \times 3 = 75 \times 3 - 28 \times 8$$

$$28^{-1}\ mod\ 75 = -8\ mod\ 75 = 75 - 8 = 67$$

Modular exponentiation

- compute $x^a (mod\ n)$
- Naïve method:
  - $x^a\ (mod\ n) = x\ (mod\ n) \times x\ (mod\ n) \times\ ...\times\ x\ (mod\ n)$
  - → repeating modular multiplication for a times
- Square and multiply algorithm

Square and multiply algorithm

- Representing a in binary notation : $a = \sum_{i=0}^{l} a_i 2^i$

$z \leftarrow 1$
For $i = l$ down to 0
  $z \leftarrow z^2 \ mod\ n$
  if $a_i = 1$ then
    $z \leftarrow (z \times x)\ (mod\ n)$
  end if
End for
Return z

E.g. Compute $x^{19}\ mod\ n$
$19 = 16 + 2 + 1 = 2^4 + 2^1 + 2^0 = 10011$
$z \leftarrow 1$
$i = 4: a_4 = 1; z \leftarrow z^2 \times x \equiv 1^2 \times x \equiv x$
$i = 3; a_3 = 0; z \leftarrow z^2 \equiv x^2$
$i = 2; a_2 = 0; z \leftarrow z^2 \equiv x^4$
$i = 1; a_1 = 1; z \leftarrow z^2 \times x \equiv x^8 \times x \equiv x^9$
$i = 0; a_1 = 1; z \leftarrow z^2 \equiv x^{18} \times x \equiv x^{19}$

E.g. Compute $3^{19}\ mod\ 5$
$19 = 10011$
$z \leftarrow 1$
$i = 4: a_4 = 1; z \leftarrow 1^2 \times 3 \equiv 3$
$i = 3; a_3 = 0; z \leftarrow 3^2 \equiv -1$
$i = 2; a_2 = 0; z \leftarrow (-1)^2 \equiv 1$
$i = 1; a_1 = 1; z \leftarrow 1^2 \times 3 \equiv 3$
$i = 0; a_1 = 1; z \leftarrow 3^2 \times 3 \equiv -3 \equiv 2$

Exercises

1. Compute
    1. $17^{-1}\ mod\ 101$
    2. $357^{-1}\ mod\ 1234$
    3. $3125^{-1}\ mod\ 9987$
    4. $9726^{3533}\ (mod\ 11413)$
    5. $127296^{907}(mod\ 186101)$
2. Given $p = 61, q = 53$
    1. create a RSA system with these values
    2. Suppose the value of plaintext is 123 -> find the cipher text
    3. Decrypt the cipher text to obtain the plain text
3. Prove that: $X^{(p-1)(q-1)} \equiv 1\ (mod\ pq)\ p, q$ are primes
4. Write pseudo code for Extended Euclidean algorithm
    1. The ones for computing modular multiplicative inverse
5. Prove the correctness of square and multiply algorithm

Projects

1. Cryptanalysis for substitution cipher
2. Cryptanalysis for vigenere cipher
3. A program for encryption and cryptanalysis of RSA as follows.
    1. Encryption:
        1. Input: plain text, and public key (e, n)
        2. Output: cipher text
        3. Encryption flow
            1. The plaintext is an English document. Each word of the plaintext is encoded as follows
                - DOG → $3 \times 26^2 + 14 \times 26 + 6 = 2398$
                - CAT → $2 \times 26^2 + 0 \times 26 + 6 = 19$
            2. Each encoded word then is encrypted using RSA with the public key (e, n)
                - Applying square and multiply for determining modular exponent
    2. Cryptanalysis
        1. Input: cipher text, and public key (e, n)
        2. Output: plaintext
        3. Hint:
            1. Determine primes p, q, s.t. n = p x q
            2. Calculate $\varphi(n)$
            3. Determine private key d
                - By using extended Euclidean algorithm
            4. Decrypt with private key d
                - Applying square and multiply for determining modular exponent