Public Key Encryption

- Two difficult problems
  - Key distribution under conventional encryption
  - Digital signature
- Diffie and Hellman, 1976
  - Astonishing breakthrough
  - One key for encryption and the other related key for decryption
  - It is computationally infeasible to determine the decryption key using only the encryption key and the algorithm

Public Key Cryptosystem

- Essential steps of public key cryptosystem
  - Each end generates a pair of keys
  - One for encryption and one for decryption
  - Each system publishes one key, called public key, and the companion key is kept secret
  - If A wants to send message to B
    - Encrypt it using B’s public key
  - When B receives the encrypted message
    - Decrypt it using its own private key

Applications of PKC

- Encryption/Decryption
  - The sender encrypts the message using the receiver’s public key
  - Q: Why not use the sender’s secret key?
- Digital signature
  - The sender signs a message by encrypt the message or a transformation of the message using its own private key
- Key exchange
  - Two sides cooperate to exchange a session key, typically for conventional encryption

Conditions of PKC

- Computationally easy
  - To generate public and private key pair
  - To encrypt the message using encryption key
  - To decrypt the message using decryption key
- Computational infeasible
  - To compute the private key using public key
  - To recover the plaintext using ciphertext and public key
  - The encryption and decryption can be applied in either order

One Way Function

- PKC boils down to one way function
  - Maps a domain into a range with unique inverse
  - The calculation of the function is easy
  - The calculation of the inverse is infeasible
- Easy
  - The problem can be solved in polynomial time
- Infeasible
  - The effort to solve it grows faster than polynomial time
  - For example: $2^n$
  - It requires infeasible for all inputs, not just worst case
Trapdoor One-way Function

- Trapdoor one way function
- Maps a domain into a range with unique inverse
- $Y = f_k(X)$
- The calculation of the function is easy
- The calculation of the inverse is infeasible if the key is not known
- The calculation of the inverse is easy if the key is known

Possible Attacks

- Brute force
  - Use large keys
  - Trade-off: speed (not linearly depend on key size)
  - Confined to small data encryption: signature, key management
- Compute the private key from public key
  - Not proven that is not feasible for most protocols!
- Probable message attack
  - Encrypt all possible messages using encryption key
  - Compare with the ciphertext to find the matched one!
  - If data is small, feasible, regardless of key size of PKC

RSA Algorithm

- R. Rivest, A. Shamir, L. Adleman (1977)
- Block cipher using integers $0 \cdots n-1$
- Thus block size $k$ is less than $\log_2 n$
- Algorithm:
  - Encryption: $C = M^e \mod n$
  - Decryption: $M = C^d \mod n$
  - Both sender and the receiver know $n$

Requirements

- Possible to find $e$ and $d$ such that
  - $M = M^{ed} \mod n$ for all message $M$
- Easy to conduct encryption and decryption
- Infeasible to compute $d$
  - Given $n$ and $e$

Key Generation

- Recall Euler Theorem
  - $a^{\phi(n)} \equiv 1 \mod n$ for all $a$
  - Then $ed \equiv 1 \mod \phi(n)$ is sufficient to make algorithm correct
- RSA chooses the following
  - Integer $n = pq$ for two primes $p$ and $q$
  - Select $e$, such that $\gcd(e, \phi(n)) = 1$
  - Compute the inverse of $e \mod \phi(n)$
  - The result is set as $d$

Key Generation

- The prime numbers $p$ and $q$ must be sufficiently large
  - They are chosen by applying primality testing of randomly chosen large numbers
  - About $\ln n$ prime numbers less than $n$ implies needs to check about $2\ln n$ random numbers to find 2 primes numbers around $n$
  - Compute $n = pq$, keep $p$ and $q$ secret!
- Select random number $e$
  - Test $\gcd(e, \phi(n)) = 1$, and get $d$ if equation holds
Security of RSA
- Brute force: try all possible private keys
- Factoring integer $n$, then know $\phi(n)$
- Not proven to be NPC
- Determine $\phi(n)$ directly without factoring
  - Equivalent to factoring! (1996)
- Determine $d$ directly without knowing $\phi(n)$
  - Currently appears as hard as factoring
  - But not proven, so it may be easier!

More Constraints
- Primes $p$ and $q$ should be in similar scale
- Both $p-1$ and $q-1$ should have large prime factor
- The gcd($p-1,q-1$) should be small
- The decryption key $d$ should larger then $n^{1/4}$

Timing Attacks
- Keep track of how long a computer takes to decrypt a message!
  - Paul Kocher, 1996
  - Stunning attack strategy and cipher only attack!
- Guessing the key bit by bit
- Countermeasures
  - Constant exponentiation time
  - Random delay
  - Blinding

Other Public Key Systems
- Rabin Cryptosystem
  - Decryption is not unique
- Elgamal Cryptosystem
  - Expansion of the plaintext (double)
- Knapsack System
  - Already broken
- Elliptic Curve System
  - If directly implement Elgamal on elliptic curve
  - Expansion of plaintext by 4, Restricted plaintext
  - Menezes-Vanston system is more efficient

Rabin Cryptosystem
- Procedure
  - Let $n=pq$ and $p=3 \text{ mod } 4, q=3 \text{ mod } 4$
  - Publish $n$, and a number $b<n$
  - For message $m$
    - $C=m(m+b) \text{ mod } n$
  - The receiver decrypt ciphertext $C$
    - $(b^{(n/4)}C)^{(1/2)-b/2}$

Analysis
- For receiver, need solve equation
  - $x^2+bx=C \text{ mod } n$
  - $x_1=x+b/2, c=b^{(n/4)}+C$, then need
    - Solve $x_1^2=c \text{ mod } n$
  - Chinese Remainder Theorem implies that
    - $x_1^2 \equiv x \text{ mod } p$
    - $x_1^2 \equiv x \text{ mod } q$
  - When $p=3$ and $q=3 \text{ mod } 4$
    - Solution $x_1 \equiv m^{(n/4)} \text{ mod } p$ and $x_2 \equiv m^{(n/4)} \text{ mod } q$
    - Then Chinese Remainder Theorem again to combine solution
Security

- Secure against
- Chosen plaintext attack

- Not secure against
- Chosen ciphertext attack

ElGamal Cryptosystem

- Based on Discrete Logarithm
  - Find unique integer \( a \) such that \( a^a \mod p \)
    - Here \( a \) is a primitive element in \( \mathbb{Z}_p \) \( p \) is prime

- Procedure
  - Make \( p, ?, \) ? public, keep \( a \) secret
  - Encryption:
    - \( E_k(x) = (x^k \mod p, x^k \mod p) \)
  - Decryption
    - \( D_k(y_1, y_2) = (y_1^a)^{-1} \mod p \)

Knapsack Cryptosystem

- Based on subset sum problem
- Given a set, find a subset with half summation value
- It is NPC problem generally
- Superincreasing set if \( s_i > ? j<i s_j \)
- The subset problem over superincreasing set can be solved in polynomial time!
- Been broken by Shamir, 1984
- Using integer programming tech by Lenstra

Knapsack System

- Procedure
  - Select a superincreasing set \( s \)
  - Let \( p \) be prime larger than set summation of \( s \)
  - Select integer \( a \), keep \( s, a, p \) secret
  - Make \( t = (as_1, as_2, ..., as_n) \mod p \) public
  - Encryption
    - \( E(x_1, x_2, ..., x_n) = x_i t_i \mod p \)
  - Decryption
    - Solve the subset summation problem \( s, a^C \mod p \)

Solve Subset Problem

- Let \( T \) be the half summation, \( t=T \)
- For \( i=n \) downto \( 1 \) do
  - If \( s_i \) then
    - Set \( x_i = 1 \)
    - Else \( x_i = 0 \)
  - If \( \sum x_i = T \) then \( (x_1, x_2, ..., x_n) \) is the solution
  - Else, there is no solution