Public Key Cryptography

Prof. Ashok K Bhateja, IIT Delhi

Secret-Key or Symmetric Cryptography

- Alice and Bob agree on an encryption method and a shared key.
- Alice uses the key and the encryption method to encrypt (or encipher) a message and sends it to Bob.
- Bob uses the same key and the related decryption method to decrypt (or decipher) the message.

Public Key Cryptography

Definition: A system where the enciphering and deciphering operations, as well as the generation of inverse key, all take polynomial time, but the deduction of deciphering key from enciphering key needs exponential time.

Public-Key Cryptography

- Developed to address two key issues:
  - Key distribution – how to have secure communications in general without having to trust a KDC
  - Digital signatures – how to verify a message comes intact from the claimed sender
- Public invention due to Diffie & Hellman at Stanford University in 1976
  - known earlier in classified community

Public key v/s Symmetric key

- Public key algorithms are significantly slower
- Key size is large
- Key secrecy and key distribution are not serious problems
- Keys may remain unchanged for considerable period of time

- At least 1000 times faster than public key
- Key size is short
- Key secrecy and key distribution are the serious problems
- The key should be changed frequently, and perhaps for each communication session

One Way Function

- A function f is called **one way** if for a given x it is easy to compute $f(x)$, but for a given $f(x)$ it is hard to compute x.
  - Ex 1: It is easy to smash a watch into hundreds of tiny pieces. However, it is not easy to put all of those tiny pieces back together into a functional watch.
  - Ex 2: It is easy to compute x^2, but much hard to compute $\sqrt{x}$.
- **Trapdoor One Way Function**: It is a special type of one-way function with a secret trapdoor. It is easy to compute $f(x)$ and hard to compute its inverse, but it becomes easy if one knows the trapdoor (secret)
  - Ex: Letter box

RSA Cryptosystem

- Public Key Cryptosystem invented by Ron Rivest, Adi Shamir and Leonard Adleman in 1978.
- Based on the dramatic difference between the ease of multiplying two large numbers and the difficulty of factoring a composite number.
- The public and the private keys are functions of a pair of large prime numbers.
- Security of the system lies in the difficulty of the problem of factoring large integers.

RSA – Key Generation

Select p & q both primes $p \neq q$

Calculate $n = p \times q$

Calculate $\varphi(n) = (p-1) \times (q-1)$

Select integer e s.t. $\gcd(e, \varphi(n)) = 1$; $1 < e < \varphi(n)$

Calculate $d = e^{-1} \bmod \varphi(n)$

Public key: e & n

Private key: d

RSA Algorithm (continue)

- Encryption:
  - Divide the plaintext into numerical blocks such that each block is just under n. Let M be a block of the plaintext.
  - Compute $C = M^e \pmod{n}$
- Decryption
  - Take each encrypted block C and compute

    $M = C^d \pmod{n}$

Example: RSA

$p = 71,\ q = 401 \Rightarrow n = p \cdot q = 28471,\ \varphi(n) = 28000$

$e = 3,\quad d = e^{-1} \pmod{\varphi(n)} = 18227$

Plaintext : INDIA IS MY COUNTRY

Let the number of alphabets be 29.

Encryption block size be 3.

Plaintext blocks IND IA IS MY COU NTR Y.

Encryption: IND $\rightarrow$ 8 13 3

$\therefore M = 3 + 13 \cdot 29 + 8 \cdot 29^2 = 7108$

$C = (7108)^3 \bmod 28471 = 14105$

$= 11 + 22 \cdot 29 + 16 \cdot 29^2$

i.e., C is 11 22 16 0

L W Q A

Decryption: $L\ W\ Q\ A \rightarrow$ 11 22 16 0

$\therefore C = 11 + 22 \cdot 29 + 16 \cdot 29^2 = 14105$

$M = (14105)^{18227} \pmod{28471} = 7108$

$= 3 + 13 \cdot 29 + 8 \cdot 29^2$

= 8 13 3

I N D

Efficient Decryption

- decryption uses exponentiation to power d
  - this is likely large, insecure if not
- can use the Chinese Remainder Theorem (CRT) to compute mod p & q separately. then combine to get desired answer
  - approx 4 times faster than doing directly
- only owner of private key who knows values of p & q can use this technique

RSA Security

- possible approaches to attacking RSA are:
  - brute force key search (infeasible given size of numbers)
  - mathematical attacks (based on difficulty of computing $\varphi(n)$, by factoring modulus n)

Mathematical Justification of RSA algorithm

$$e \cdot d \equiv 1 \bmod \varphi(n) \Rightarrow e \cdot d \equiv t \cdot \varphi(n) + 1$$

Let M be the plaintext and C be its ciphertext

$$C = M^e \pmod{n}$$

$$C^d = M^{ed} \pmod{n}$$

$$= M^{t \cdot \varphi(n) + 1} \pmod{n}$$

$$= M^{t \cdot \varphi(n)} \cdot M \pmod{n}$$

$$= 1 \cdot M \pmod{n} \quad \text{because } M^{\varphi(n)} \equiv 1 \pmod{n}$$

$$= M \pmod{n}$$

Selection of primes

- p and q should be about the same bit length, and sufficiently large.
- difference $p - q$ should not be too small, otherwise if $p \approx q$ and hence $p \approx \sqrt{n}$. Thus, n could be factored efficiently simply by trial division by all odd integers close to $\sqrt{n}$.
- p and q be strong primes. i.e.
  - $p - 1$ has a large prime factor, denoted r;
    - to foil Pollard's $p - 1$ attack
  - $p + 1$ has a large prime factor, denoted s;
    - to foils the $p + 1$ attack
  - $r - 1$ has a large prime factor, denoted t.
    - to foil cycling attacks

Time Complexity of RSA

- Three major components of the RSA algorithm are exponentiation, inversion and modular operation.
- Modular addition takes O(log$n$).
- Modular multiplication takes O((log$n$)^2).
- Modular multiplication using squaring and multiply to find $m^e \bmod n$ multiplying the digits of m, log$e$ times.
- Complexity of encryption O((log$n$)^3).
- Complexity of decryption O((log$n$)^3).
- Time to find inverse O((log$n$)^2).

  (using Euclidean extended algorithm)

Rabin public-key encryption

- The Rabin encryption scheme is based on the fact that it is easy to compute square roots modulo a composite number n if the factorization of n is known, yet it appears to be difficult to compute square roots modulo n when the factorization of n is unknown.
- Algorithm - Rabin public-key
  - Generate two large random (and distinct) primes p and q, each roughly the same size.
  - Compute $n = pq$. A's public key is n; A's private key is (p, q).
  - Encryption: Compute $c = m^2 \bmod n$.
  - Decryption: find the 4 square roots m_1, m_2, m_3, and m_4 of c modulo n, and decide which of these is m.

Computing Square Roots Modulo n

- Compute $a_p = a \bmod p$ and $a_q = a \bmod q$.
- Compute a square root x_p of a_p modulo p
- Compute a square root x_q of a_q modulo q
- Use Chinese Remainder Theorem & solve

$$x \equiv x_p \bmod p$$

$$x \equiv x_q \bmod q$$

- Compute the square root of 3 modulo 143

  3 modulo 143 = 3 mod (11 ×13)

  $\sqrt{3} \pmod{11} = \pm 5$ i.e., 5 & 6

  $\sqrt{3} \pmod{13} = \pm 4$ i.e., 4 & 9

- Using the Chinese Remainder Theorem

  5 mod 11 & 4 mod 13 $\Rightarrow$ 82 mod 143

  5 mod 11 & 9 mod 13 $\Rightarrow$ 126 mod 143

  6 mod 11 & 4 mod 13 $\Rightarrow$ 17 mod 143

  6 mod 11 & 9 mod 13 $\Rightarrow$ 61 mod 143

  the 4 square roots of 3 mod 143 are 82, 126, 17 and 61.

Computing Square Roots Modulo n

- Compute $a_p \equiv a \bmod p$ and $a_q \equiv a \bmod q$.
- Compute r and $-r$ square roots of a_p modulo p
- Compute s and $-s$ square roots of a_q modulo q
- Use the extended Euclidean algorithm to find integers c and d such that $cp + dq = 1$.
- Set $x \leftarrow (rdq + scp) \bmod n$

  & $y \leftarrow (rdq - scp) \bmod n$.
- Return($\pm x \bmod n, \pm y \bmod n$).

Example: Rabin public-key cryptosystem

- $p = 7$, $q = 11$, $n = 77$
- Encryption: Let $m = 20$

  $c = m^2 \bmod 77 = 400 \bmod 77 = 15$
- Decryption: $a_p = 1$, $a_q = 4$

  Square roots of a_p and a_q: $r = 1$, $s = 9$

  Using Euclidean Algo, $cp + dq = (-3 \cdot 7) + (2 \cdot 11) = 1$

  $x = (-3 \cdot 7 \cdot 9 + 2 \cdot 11 \cdot 1) \bmod 77 = 64$

  $y = (-3 \cdot 7 \cdot 9 - 2 \cdot 11 \cdot 1) \bmod 77 = 20$

∴ Plaintext candidates: 64, -64, 20, -20

i.e., 64, 13, 20, 57

Knapsack public-key encryption

- Merkle-Hellman Knapsack public-key encryption schemes are based on the subset sum problem, which is NP-complete
- The basic idea is to select an instance of the subset sum problem that is easy to solve, and then to disguise it as an instance of the general subset sum problem which is difficult to solve.
- The original knapsack set can serve as the private key, while the transformed knapsack set serves as the public key.

Knapsack problem

Let $A = \{a_1, a_2, \ldots, a_r\}$ i.e., $\mathbf{a} = [a_1, a_2, \ldots, a_r]$

Message $\mathbf{m} = \{\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_s\}$, where $\mathbf{m}_i \in \{0, 1\}^r \ \forall\ i$

Cipher $C = \{c_1, c_2, \ldots, c_s\}$, where $c_i = \mathbf{a} \,.\, \mathbf{m}_i$

- Def: Given an integer t and a set A of positive integers $a_1, a_2, \ldots, a_r$, is there a subset of A whose elements add up to t ?
- Example: A = {2106, 880, 1320, 974, 2388, 1617, 1568, 2523, 48, 897}

  $\mathbf{m}$ = 01110011000110100101100000 1010

  $\mathbf{m}_1$ = 0111001100, $\mathbf{m}_2$ = 0110100101, $\mathbf{m}_3$ = 1000001010

  c_1 = 7265, c_2 = 8008, c_3 = 3722

Thus, the complete message becomes 7265, 8008, 3722

Knapsack public-key encryption

- Super increasing sequence:
  A set of + ve integers $A' = \{a'_1, a'_2, \ldots, a'_r\}$ s.t. $a'_{k+1} > \sum_{i=1}^{k} a'_i$
- Example: $A' = \{2, 6, 9, 19, 41, 79, 159, 320, 643, 1310\}$
- Key generation
  - An integer r size of the knapsack
  - Choose a super increasing sequence of r integers
  - Choose integer n s.t. $n > a'_1 + a'_2 + \ldots + a'_r$
  - Select a random integer w, $1 \leq w \leq n$, s.t. gcd $(w, n) = 1$
  - Select a random permutation π of the integers $\{1, 2, \ldots, r\}$
  - Find $A = \{a_1, a_2, \ldots, a_r\}$ s.t. $a_i = w\, a'_{\pi(i)} \pmod{n}$
- Private key: n, w, A' , π
- Public key: A

Knapsack public-key encryption

- Private key: n, w, A', π
- Public key: A
- Encryption: Let $\mathbf{m} = \{x_1, x_2, \ldots, x_r\}$ be a message block

  $c = (x_1 a_1 + x_2 a_2 + \ldots + x_r a_r) \bmod n$
- Decryption: Compute $w^{-1} \cdot c \bmod n$

  $d = w^{-1} c \bmod n$

  Using A' and permutation π ,

  find $x_1, x_2, \ldots, x_r$ by solving super increasing subset sum problem

Knapsack Cryptosystem: Example

- Super increasing sequence, A′ = (12, 17, 33, 74, 157, 316), $r = 6$, $n = 737$, $w = 635$
- Permutation $\pi(1) = 3$, $\pi(2) = 6$, $\pi(3) = 1$, $\pi(4) = 2$, $\pi(5) = 5$, and $\pi(6) = 4$.
- Public key: A = (319, 196, 250, 477, 200, 559), (use $a_i = w\, a'_{\pi(i)} \pmod{n}$)
- Private key: n, w, A′
- Encryption: message **m** = 101101,

  c = (319 + 250 + 477 + 559) mod 737 = 131
- Decryption: w^{-1} mod 737 = 513, Computes $d = w^{-1}\, c \bmod n = 136$,

  solves the super increasing subset sum problem

  136 = 12 + 17 + 33 + 74.
- Applying inverse permutation, message bits: $x_3 = 1$, $x_4 = 1$, $x_1 = 1$, $x_6 = 1$.

  message **m** = 101101