CPE5021
Advanced Network Security

--- Advanced Cryptography: RSA and its implementation ---

Lecture 1.1

Mathematical background

Modular operations

- "remainder"
  - 13 mod 5 = 3, 1 mod 7 = 1
  - 20 mod 5 = 0, 32 mod 7 = 4
- modular exponentiation
  - 2^2 mod 3 = 1, 3^2 mod 3 = 0
  - 2^2 mod 5 = 4, 10^2 mod 92 = 8
  - 4^6 mod 10 = 6, 3^{11} mod 10 = 7

Mathematical background

- *a* is relatively prime to *b* if the largest integer that divides both *a* & *b* is 1
  - E.g:
    - any *m* (<>0) is relatively prime to a prime number
    - is 9 relatively prime to 10?

Mathematical background

- Let ø(*n*) denote the total numbers that are *less than n* and relatively prime to *n*
  - If *n* is a prime number then ø(*n*) = *n* – 1
  - If *p*, *q* are prime numbers and *n*=*p***q*, then
    - Ø(*n*) = Ø(*p***q*) = *p***q* – (*p* + *q* -1) = (*p*-1)*(*q*-1)
    - *p* & *q* are prime numbers => only multiples of *p* and *q* are not relatively prime to p*q
    - That is: there are (*p* + *q* – 1) multiples [0 is counted once] of *p* and *q*
    - E.g: *p* = 3; *q*=7; {0, 3, 7, 6, 9, 12, 14, 15, 18} are not relatively prime to *p***q*
    - Ø(*n*) = ø(*p***q*) = 12 ; {1,2,4,5,8,10,11,13,16,17,19,20}

Mathematical background

- *y* & *n* are integers and *y* mod ø(*n*) = 1, for any *x* < *n*, x^y mod *n* = *x* (1)
  - E.g:
    - y=13 ; n=7; x = 4;
    - ø(*n*) = 6; y mod ø(*n*) = 13 mod 6 = 1;
    - $x^y = 4^{13}$; x^y mod n = 4^{13} mod 7 = 4 = x mod n;

Mathematical background

- The multiplicative inverse of *x* with modulo *n* is *y* such that: (*x***y*) mod *n* = 1 (2).
- The above multiplicative inverse can be used to create a simple public key cipher: either *x* or *y* can be thought of as a secret key and the other is the public key.

E.g: *x*=3; *n*=10; *y*=7; we have: (3*7) mod 10 = 1;

- M =5 ;
  - 3*5 (mod 10) = 5 ; 5*7 (mod 10) = 5 = M (message)
- M =6 ;
  - 3*6 (mod 10) = 8; 8*7 (mod 10) = 6 = M (message)

RSA (1)

- Bob:
  - chooses 2 large prime numbers: *p, q*
    multiplies p and q: *n = p*q*
  - finds out two numbers *e* & *d* such that
    - *(e * d) mod ø(n) = 1* [similar to (2)]

    Or *(e * d) mod [(p-1)*(q-1)] = 1*
  - public key (published in the phone book)
    - 2 numbers: *(e, n)*
    - encryption alg: modular exponentiation
  - secret key: *(d,n)*

RSA (2)

- Alice has a message *m* to be sent to Bob:
  - finds out Bob's public encryption key *(e, n)*
  - calculates

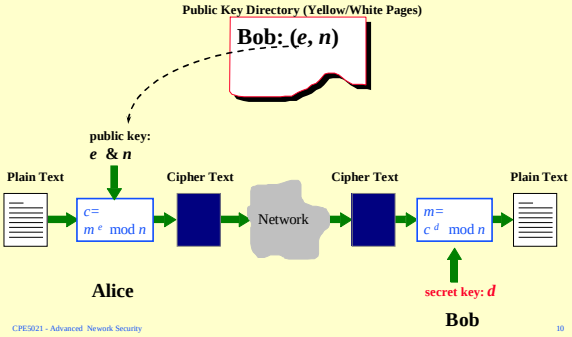
    $m^e \pmod n \rightarrow c$
  - sends the ciphertext *c* to Bob

RSA (3)

- Bob:
  - receives the ciphertext *c* from Alice
  - uses his matching secret decryption key *d* to calculate

    $c^d \pmod n \rightarrow m$

RSA Public Key Cryptosystem



RSA --- 1st small example (1)

- Bob:
  - chooses 2 primes: *p=5, q=11*
    multiplies p and q: *n = p*q = 55*
  - finds out two numbers *e=3* & *d=27* which satisfy

    *(3 * 27) mod 40 = 1*
  - Bob's public key
    - 2 numbers: *(3, 55)*
    - encryption alg: modular exponentiation
  - secret key: *(27,55)*

RSA --- 1st small example (2)

- Alice has a message *m=13* to be sent to Bob:
  - finds out Bob's public encryption key *(3, 55)*
  - calculates c:

    $c = m^e \pmod n$

    $= 13^3 \pmod{55}$

    $= 2197 \pmod{55}$

    $= 52$
  - sends the ciphertext *c=52* to Bob

RSA --- 1st small example (3)

- Bob:
  - receives the ciphertext *c=52* from Alice
  - uses his matching secret decryption key *27* to calculate m:
    $m = 52^{27} \pmod{55}$
    $= 13$ *(Alice's message)*

How does RSA work?

- $n = p*q$ => $\emptyset(n) = \emptyset(p*q) = (p-1)*(q-1)$
- We choose *d* & *e* such that
  - $(e * d) \bmod \emptyset(n) = = 1$; similar to (2)
  - for any $m < n$: $m^{de} = m \bmod n$; from (1)
  - an RSA encryption consists of taking m and raising it to *e*; and decrypting the ciphertext by raising the result of the encrytion to *d*:
    - We have $(a*b) \bmod n = ((a \bmod n) * (b \bmod n)) \bmod n$
    - hence : $(m^e \bmod n)^d \bmod n = (m^e)^d \bmod n = (m^{ed}) \bmod n = m \bmod n = m$ [from (1)]

RSA Signature --- an eg (1)

- Bob:
  - chooses 2 primes: *p=5, q=11*
    multiplies p and q: $n = p*q = 55$
  - finds out two numbers *e=3* & *d=27* which satisfy
    $(3 * 27) \bmod 40 = 1$
  - Bob's public key
    - 2 numbers: (*3, 55*)
    - encryption alg: modular exponentiation
  - secret key: (*27,55*)

RSA Signature --- an eg (2)

- Bob has a document *m=19* to sign:
  - uses his secret key *d=27* to calculate the digital signature of *m=19*:
    $s = m^d \pmod{n}$
    $= 19^{27} \pmod{55}$
    $= 24$
  - appends 24 to 19. Now *(m, s) = (19, 24)* indicates that the doc is 19, and Bob's signature on the doc is 24.

RSA Signature --- an eg. (3)

- Cathy, a verifier:
  - receives a pair *(m,s)=(19, 24)*
  - looks up the phone book and finds out Bob's public key *(e, n)=(3, 55)*
  - calculates $t = s^e \pmod{n}$
    $= 24^3 \pmod{55}$
    $= 19$
  - checks whether *t=m*
  - confirms that (*19,24*) is a genuinely signed document of Bob if *t=m*.

RSA Algorithm (1)

General description of RSA algorithm

- Choose 2 large prime numbers: *p, q*; then multiplie p and q: $n = p*q$
- Find out two numbers *e* & *d* such that
  - $(e * d) \bmod [(p-1)*(q-1)] = 1$
- Encrypt a message M using e: $M^e \pmod{n}$ -> c
- Decrypt using d: $c^d \pmod{n}$ -> M

RSA Algorithm (2)

1. **Choose 2 large prime numbers:** ***p and q*;**
2. **Calculate** $n = p*q$; $m = (p-1)*(q-1)$;
3. **Find** e **such that** gcd(e,m) **= 1;**
4. **Find** d **such that : k*m + d*e = 1;**
5. **Encrypt a message M using e:**
   M^e (mod n) -> c

RSA – How to choose p and q

Finding *p* and *q* which are prime and large enough is important.
There is no guarantee that we can find p and q correctly.
However, we can use Miller-Rabin test to test if an integer n is probably a prime number by:

1.Select a random number *b* from the set of integers [1, (*n*-1)] (pick n as large as you want).
2. Find q and the odd number *k* such that $n - 1 = 2^q k$
3. Test if either of the following condition holds:
(a) b^k mod n = 1; or
(b) if there exists *i* in [0, (q-1)] such that $b^{k2^i} \bmod n = -1$
4. If neither* (a) *nor* (b) *is satisfied, then n is composite;
Else
n is probably prime (Inconclusive)
(The probability of *n* being a prime number after *i* inconclusive tests is $(1 - (1/4)^i)$)

* There are other ways to test if an integer is probably a prime number

Miller-Rabin test – E.g.

- **Test if n = 15 is a prime number**
  1. **Choose random b = 8**
  2. **Solve (15 -1) = $2^q k$; => *q =1 ;k=7;***
  3. (a) **Is $8^7 \bmod 15 = 1$? NO**
     (b) Does any *i* **in [0, (q-1)] satisfy:** $b^{k2^i} \bmod n = -1$
     i = 0 **satisfies ; but from (a)** $8^{7*2^0} \bmod 15 = -1$ **is false;**
     => *n* is composite.

Miller-Rabin test – E.g.

- **Test if n = 13 is a prime number**
  1. **Choose random b = 3**
  2. **Solve (13 -1) = $2^q k$; => *q =2 ;k=3;***
  3. (a) **Is $3^3 \bmod 13 = 1 \Leftrightarrow 27 \bmod 13 = 1$? YES**
     $\Rightarrow$ ***n* is probably prime number**

Miller-Rabin test – E.g.

- **Test if n = 13 is a prime number**
  1. **Choose random b = 2**
  2. **Solve (13 -1) = $2^q k$; => *q =2 ;k=3;***
  3. **(a) Is $2^3 \bmod 13 = 1? \Leftrightarrow 8 \bmod 13 = 1$? NO**

  (b) Does any *i* in [0, (q-1)] satisfy: $b^{k2^i} \bmod n = -1$
  ***i* = 0**

  $$2^{3*2^0} \bmod 13 = 8 \bmod 13 = 8$$

  i = 1;

  $$2^{3*2^1} \bmod 13 = 64 \bmod 13 = -1$$

  $\Rightarrow$ ***n* is probably prime number with the probability of 75%**

RSA – find e and d

- **Find two numbers *e* and *d* such that (*e* * *d*) *mod* [(*p-1*)*(*q-1*)] = *1***
  - **First calculate** e **relatively prime to** m: gcd(e,m) **= 1 using Euclid's algorithm**

```
int gcd(int e, int m) {
    if (m == 0) return e;
    else return gcd(m, e%m) ;
}
```

RSA – find e and d (con't)

- **Then find** d **using extended Euclid's algorithm (EEA).**
  - **d*e mod m = 1**
    => 1 **= d***e **+ k***m**; where k is an integer.**
  - 1 **= d***e **+ k***m ⇔ a**x** + b**y** = 1;

Note: **We can also use a general EEA to find** e **and** d **at the same time**.

Research for the implementation of RSA

- **There are many free source codes implementing RSA in C, C++ and Java. (you can study the source codes but not copy them).**
- **Study the Euclidian algorithms and implement them if you need to.**
- **Implement the Miller Rabin test.**
- **Implement the random number generator.**
- **Find a way to convert a message into a number before you encrypt it and convert the encrypted number back to the text format (original message).**