Topic 4

Public-key Cryptography

Understand the principles of public-key (asymmetric) cryptography

Overview

- ❑ Background
- ❑ Some Basics in Number Theory
- ❑ RSA Algorithm
- ❑ Conclusion

Source: chapter 9 of Stalling's book:

Cryptography and Network Security

Background

- Up to this point, all cryptographic schemes are based on shared secret keys - symmetric (or conventional) cryptography.
- The problems with symmetric cryptography - motivations
  - A separate key is needed for each pair of users (or even for each ciphertext encryption – session key).
    - So an n-user system requires $n*(n-1)/2$ keys - the n^2 problem.
    - Generating and distributing these keys are a challenging problem.
    - Maintaining security for the keys already distributed is also challenging - can one remember so many keys?
  - As two parties share the same key, non-repudiation can not be achieved.
- In 1976, Diffie and Hellman first presented the concept of public key cryptography.

Background - PKC Features

- Public-key cryptosystems are based on mathematically hard problems rather than substitution/transposition (permutation) ciphers.
- A pair of keys used: One is private (secret), the other can be made public. The pair of private and public keys are related mathematically. It is infeasible to generate one from the other.
- Encryption generated with one key can only be decrypted with the other key in the pair.
- Exemplar PK ciphers: RSA, DSS (Digital Signature Standard), DH (Diffie-Hellman), etc.

Background - PKC Features

Key (Ke)

Plaintext (M) → Encrypt (E) → Ciphertext (C)

$C = E_{ke}(M)$

Key (Kd)

Ciphertext (C) → Decrypt (D) → Plaintext (M)

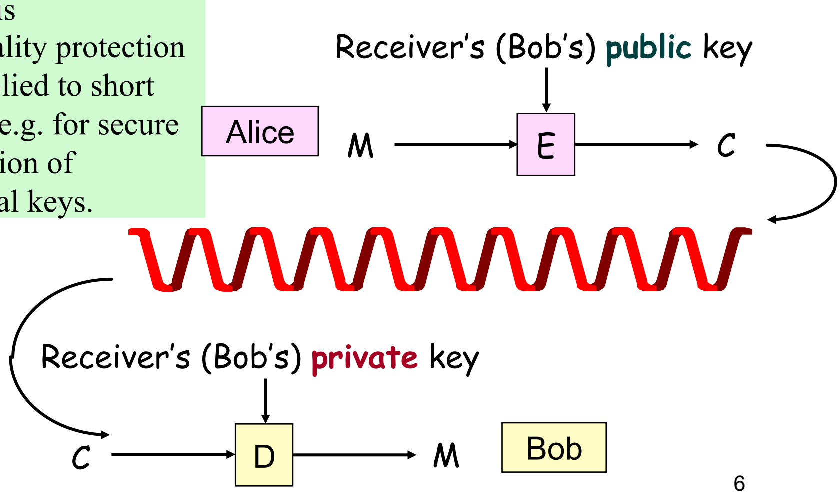
$M = D_{kd}(C)$

$M = D_{kd}\,(E_{ke}(M))$

We have a pair of keys here: {Ke, Kd}

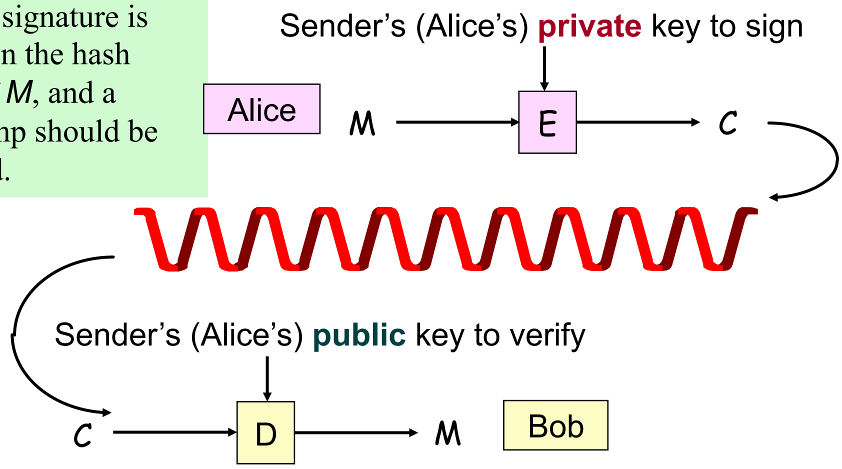
PKC Applications - Achieve Confidentiality (Secrecy)

- Usually this confidentiality protection is only applied to short messages, e.g. for secure transportation of symmetrical keys.



PKC Applications – Achieve Authenticity

- Usually signature is signed on the hash value of *M*, and a timestamp should be included.



Background – PKC Applications

- Applications of public-key ciphers
  - Confidentiality
    - Encrypt the plaintext *M* using recipient's public key;
    - As only the recipient has the corresponding private key, so *M* can only be read by the recipient.
  - Digital signature - message authenticity (message authentication and integrity) and non-repudiation of message origin
    - Sign *M* (actually the hash of *M*) using sender's private key;
    - As only the sender has this private key, so the message must have been signed by the sender.

Background - PKC Features

- Public Key Cryptography (PKC) is based on the idea of a trapdoor function, or mathematically "hard" problems, e.g. factoring large composites of primes, discrete logarithms.
- Easy to generate keys (public and private).
- Easy to encrypt and decrypt if the right key is known.
- Hard to compute private key from public key.
- Hard to recover plaintext from ciphertext without the right key.

One-way function, f

C = f (M) "Easy"

M = f^{-1} (C) "Infeasible"

Trap-door one-way function, f

C = f (K, M) "Easy" if K & M known

M = f^{-1} (K, C) "Easy" if K & C known

M = f^{-1} (K, C) "Infeasible" if K not known, C known

Background

- Since 1976, numerous public-key cryptographic algorithms have been proposed. Among those secure and practical public-key algorithms
  - some are suitable for **encryption** (+ **key distribution**);
  - some are only useful for **digital signatures**;
  - some are for **key agreement**;
  - only three algorithms, RSA, ElGamal and Rabin, works well for both encryption and digital signatures.

Mathematical Basics - Modulo Operator

- With the modulo operation you are interested in the remainder left over from division with an integer number.

- **Mathematical definition**

$$a \equiv b \bmod n$$

means there exists an integer number k such that a can be represented as

$$a \equiv k \cdot n + b$$

with the condition that: $0 \leq b \leq n-1$

Here we are not interested in the value of k; the important thing is its existence.

Some Basics in Number Theory - Modular arithmetic

- Given integers, a, b, and $n \neq 0$, *a is congruent* to *b* modulo *n if and only if* $a\text{-}b \equiv k \cdot n$ for some integer k, *i.e. n divides (a-b).*
- *Notation:* $\boldsymbol{a \equiv b \; (mod\; n)}$
- We call n the modulus, and b is remainder or residue of a modulo n.
- Examples:
  - 9 mod 5 = 4
  - 20 mod 9 = 2
  - 17 = 2 *mod 5 since 17-2 = 3 ·5*
- $x \equiv_n y$ if and only if (x mod n) = (y mod n)

An example
The modulo operator is commutative with the basic arithmetic operations. For example, it does not matter whether you **first multiply**

$$18 \cdot 13 = 234 \equiv 4 \; mod \; 10$$

or first calculate the modulus **and then multiply:**

$$18 \cdot 13 \equiv 8 \cdot 3 \; mod \; 10$$
$$= 24 \; mod \; 10 \equiv 4 \; mod \; 10$$

Some Basics in Number Theory - Modular arithmetic

- Properties
  - $a \equiv a \bmod n$
  - $a \equiv b \bmod n \Leftrightarrow b \equiv a \bmod n$
  - $a \equiv b \bmod n$ & $b \equiv c \bmod n \Rightarrow a \equiv c \bmod n$
  - $(a + b) \bmod n = ((a \bmod n) + (b \bmod n)) \bmod n$
  - $(a \cdot b) \bmod n = ((a \bmod n) \cdot (b \bmod n)) \bmod n$
  - $a \cdot (b + c) \bmod n = (a \cdot b + a \cdot c) \bmod n$
  - $a \cdot x \bmod n = 1$ where x is an integer and called the multiplicative inverse of a; in this case, x can be written *as* a^{-1}, *i.e.* $a \cdot a^{-1} \bmod n = 1$.

Some Basics in Number Theory - Modular arithmetic

- Existence of **multiplicative inverse**
  - Given $a \in [0, n-1]$, find $x \in [0, n-1]$ such that $a \cdot x \bmod n \equiv 1$;
  - E.g. as $3 \cdot 4 \bmod 11 \equiv 12 \bmod 11 \equiv 1$, so we say, 3 and 4 are each other's multiplicative inverse mod 11.

- iff a and n are relative prime, i.e. $\gcd(a, n)=1$, then $a \in [0, n-1]$ has a unique inverse modulo n.

- An integer $p > 1$ is a **prime number** if it is divisible only by itself and 1, e.g. 7.
- a and b are said to be **relatively prime** if only 1 can divide each of them, e.g. are 8 and 15 relatively prime?

Some Basics in Number Theory - Multiplication table *Mod 11*

x \ *a*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
1	1	2	3	4	5	6	7	8	9	10
2	2	4	6	8	10	1	3	5	7	9
3	3	6	9	1	4	7	10	2	5	8
4	4	8	1	5	9	2	6	10	3	7
5	5	10	4	9	3	8	2	7	1	6
6	6	1	7	2	8	3	9	4	10	5
7	7	3	10	6	2	9	5	1	8	4
8	8	5	2	10	7	4	1	9	6	3
9	9	7	5	3	1	10	8	6	4	2
10	10	9	8	7	6	5	4	3	2	1

Another example - Multiplication table *Mod 4*

x \ *a*	*1*	*2*	*3*
1	1	2	3
2	2	0	2
3	3	2	1

The inverse of 2 (mod 4) does not exist, because there isn't another number x in the finite field that can satisfy

$a*x = 1 \bmod 4.$

Not surprising!

There are two common divisors between *a* (2) and *n* (4), as 2 and 4 are not relatively prime.

Some Basics in Number Theory - Multiplication table Mod 11

- The *Table* gives the multiplication results for mod 11 (11 is a prime), the following can be noted:
  - In each multiplication result row/column, we can find all the (positive integers) numbers less than 11.
  - Each multiplication result is found only **ONCE** in each row and column.
  - The two numbers a and a^{-1}, that fulfil the requirement: $a * a^{-1} \equiv 1 \bmod 11$, (or $a * x \equiv 1 \bmod n$) are the multiplicative inverse of each other.
  - For example, 1 and 1; 2 and 6; 3 and 4; etc.

There is only one solution to this equation, when a and n are co-prime.

RSA Algorithm - Preliminaries

- The algorithm was invented by Ron **R**ivest, Ali **S**hamir, and Leonard **A**dleman.
- It is by far the easiest to understand and implement.
- It has withstood years of cryptanalysis - remains by far most popular and well trusted scheme.
- The algorithm actually consists of two numbers, the modulus (represented by the letter n) and the public exponent (represented by the letter e).
- The modulus is the product of two very large prime numbers (100 to 400 digits), represented by the letters p and q. p and q need to be kept secret.

RSA Algorithm - Preliminaries

- It is a block cipher in which the plaintext and ciphertext are integers between 0 and n-1 for some n.

- The algorithm can be described in three steps:
  - **Key generation**
  - **Encryption**
  - **Decryption**

  Use the same mathematical function, but different keys.

19

RSA Algorithm - Key generation

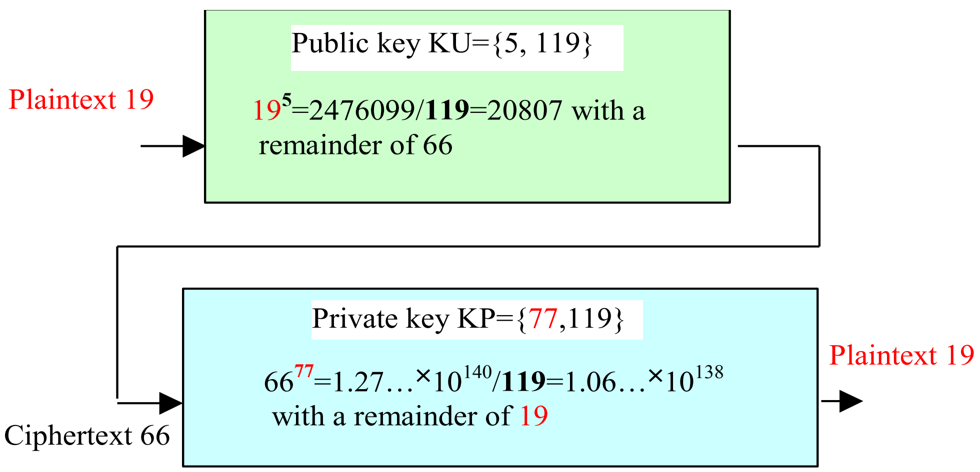
- *Step 1* - Key generation:
  - select two large primes (e.g. 200 digits) p and q
  - calculate $n = p*q$ and $\varphi(n) = (p - 1)*(q - 1)$
  - select integer e relatively prime to $\varphi(n)$ & $1<e<\varphi(n)$
  - calculate $d = e^{-1} \bmod \varphi(n)$ (or $de = 1 \bmod \varphi(n)$)
  - **public key = {e, n }**
  - **private key ={d, n}**
  - *To summarise*:
    - p, q are private & chosen;
    - $n = p*q$ is public & calculated (but keep p, q secret);
    - e is public & chosen, and d is private & calculated.

20

RSA Algorithm - Encryption & Decryption

- *Step 2* - Encryption:
  - represent the plaintext as an integer M in [0, n-1], i.e. $M < n$;
  - ciphertext: $C = M^e \bmod n$
- *Step 3* - Decryption:
  - ciphertext: C
  - plaintext: $M = C^d \bmod n$
- An example of using RSA to encrypt a message
  - select p=7 and q=17
  - calculate $n = p*q = 119$ and $\varphi(n) = (p-1)*(q-1) = 96$
  - select $e = 5$, relatively prime to $\varphi(n)$=96 and less than $\varphi(n)$
  - calculate d=77, such that $de = 1 \bmod \varphi(n)$ (= 96) and $d < 96$
  - let $M = 19$, then ciphertext $C = 19^5 \bmod 119 = 66$.

RSA Algorithm - An example



RSA Algorithm - Standard

- PKCS#1 standard defines the use of RSA algorithm. It defines the key generation, encryption, decryption, digital signatures, verification, public key format, padding, and several other issues with RSA. It is probably the most widely used RSA standard, and most of the security protocols using RSA are also compatible with the PKCS#1 standard.
  - PKCS#1 standard - http://www.rsasecurity.com/rsalabs/pkcs/pkcs-1/index.html

RSA Algorithm - Some facts for the RSA

- Security of RSA relies on difficulty of finding d given $\{e, n\}$.
  - the problem of computing d from $\{e, n\}$ is computationally equivalent to the problem of factoring n
    - If one can factorise n, then he can find p and q, and hence calculated d;

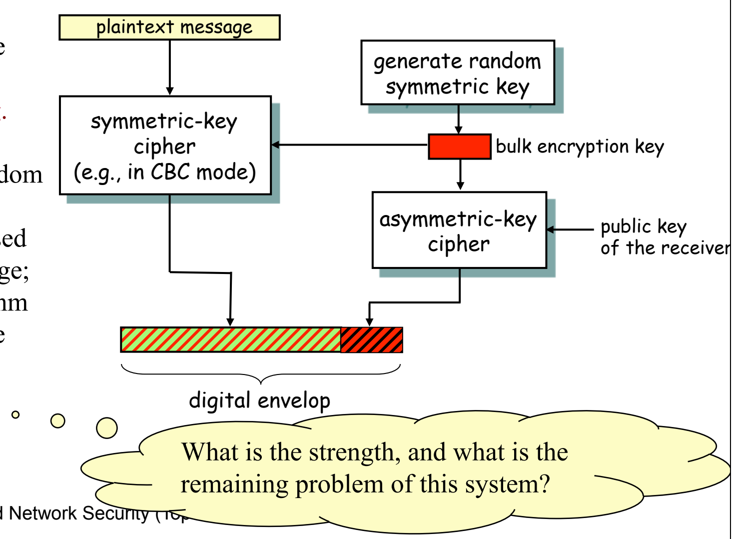
- p and q should differ in length by only a few digits, and both should be on the order of 100 - 200 digits or even larger.
  - n with 150 digits could be factored in about 1 year.
  - factoring n with 200 digits could take about 1000 years (assuming about 10^{12} operations per second).

Hybrid Cryptosystems

- Public key ciphers are much slower than symmetric key ciphers.
  - E.g. 1000 times slower in hardware, and 100 times slower in software, than DES.
- Symmetric ciphers
  - have key management problem.
  - can not provide non-repudiation service without the involvement of a trusted third party.
- So, usually, we combine them to get the strengths of both – this leads to the hybrid cryptosystem
  - Public cipher for symmetric key establishment/transportation and/or for digital signature generation.
  - Symmetric cipher for bulk encryption.

Hybrid Cryptosystems

- To speed the things up, hybrid cryptosystems are used - the technique is called digital enveloping.
  - A symmetrical algorithm with a random session key (bulk encryption key) is used to encrypt the message;
  - A public-key algorithm is used to encrypt the random session (symmetric) key.



Exercise 4 (1/3)

- 4(a):

  You are a recipient of p = 5, q = 7. You make the modulus n = 35 public. You also choose an exponent e = 5 and make that public too.

  Messages are sent to you, one letter at a time. Letters are coded into numbers as: A -> 0, B -> 1, and so on.

  Now, the following message has arrived for you:

  17 19 7 9 0 12 24

  Decrypt this message.

Exercise 4 (2/3)

- 4(b):
  - Use the RSA Demonstration facility in CrypTool to familiarize yourself with the RSA algorithm; the facility is available via Menu: “Indiv. Procedures” \“RSA Cryptosystem”.
  - Use the Hybrid encryption visualization facility in CrypTool to familiarize yourself with the RSA-AES encryption/decryption process; the facility is available via Menu: “Encrypt/Decrypt” \“Hybrid”.
  - Use the RSA encryption facility in CrypTool to encrypt two files with two different sizes (i) 1000 KB and (ii) 2000 KB, and record the encryption times; the facility is available via Menu: “Encrypt/ Decrypt” \“Asymmetric”.

Exercise 4 (3/3)

- 4(c):
  - Generate two pairs of RSA keys, one 1028-bits long and the other 2048-bits long. Record the key generation times. The facility is available via Menu: "Digital Signature\PKI" \"PKI"\ "generate/ import keys".
  - Create a big file, say around 2 Mbytes, encrypt this file using different crypto algorithms (symmetric and asymmetric) and different RSA keys you have generated, and record the encryption times (the 'encryption time' facility is available for RSA, but not available for symmetric algorithms. However you can tell their differences).
  - What observations can you make? Try to explain your observations.

Conclusion

- **Two primary use** of public key cryptography are
  - **Key establishment**
    - Key exchange (or key transportation)
      - *A* generates a symmetric key and transport it to *B* using *B*' s public key.
      - RSA can be used for key exchange.
    - Key agreement
      - Both A and B co-operate to generate a shared key.
      - DH is a key agreement algorithm (*another public-key algorithms to be presented in Lecture: Key Management*).
  - **Digital signatures**
    - Often using RSA or DSS (Digital Signature Standard) – see topic 6.
- Public key cryptography provides capabilities that can not be attained with symmetric cryptography, but it is too inefficient to be used alone - for large text encryption.