الجامعة الافتراضية السورية

أمن الشبكات
Network and Infrastructure Security

د. محمد العصورة

Dr. Mohammed Assora

دكتوراه في امن شبكات الحواسب

PhD. In Computer Network Security

Objectives of Lecture

- Understanding the needs of Asymmetric cryptosystem
- Understanding the main basics, advantages and disadvantages of Asymmetric cryptography
- Understanding the meaning of one way function and hard mathematical problem
- Understanding the meaning of public key certificate.

Contents

The Key Distribution Problem

- For symmetric ciphers, each pair of communicating agents needs a unique key.
- The generation, distribution and storage of the keys are called key management.
- In open systems, where number of users is big, secret key cryptography does not scale well.
- For example, if a cryptosystem has *n* users, each user must have *(n-1)* keys, and the total number of keys is $\frac{n^2 - n}{2}$
- If *n=1000*, each user must have *(999)* keys, and the total number of keys in the cryptosystem is 499500.

Asymmetric (or Public-key) Encryption

- Public key cryptography was invented in 1976 by Whitfield Diffie and Martin Hellman.
  - For this reason, it is sometime called Diffie-Hellman encryption.
- It is also called asymmetric encryption because it uses two keys instead of one key (symmetric encryption).
  - A public key known to everyone(widely distributed)
  - A private or secret key known only to the recipient of the message.

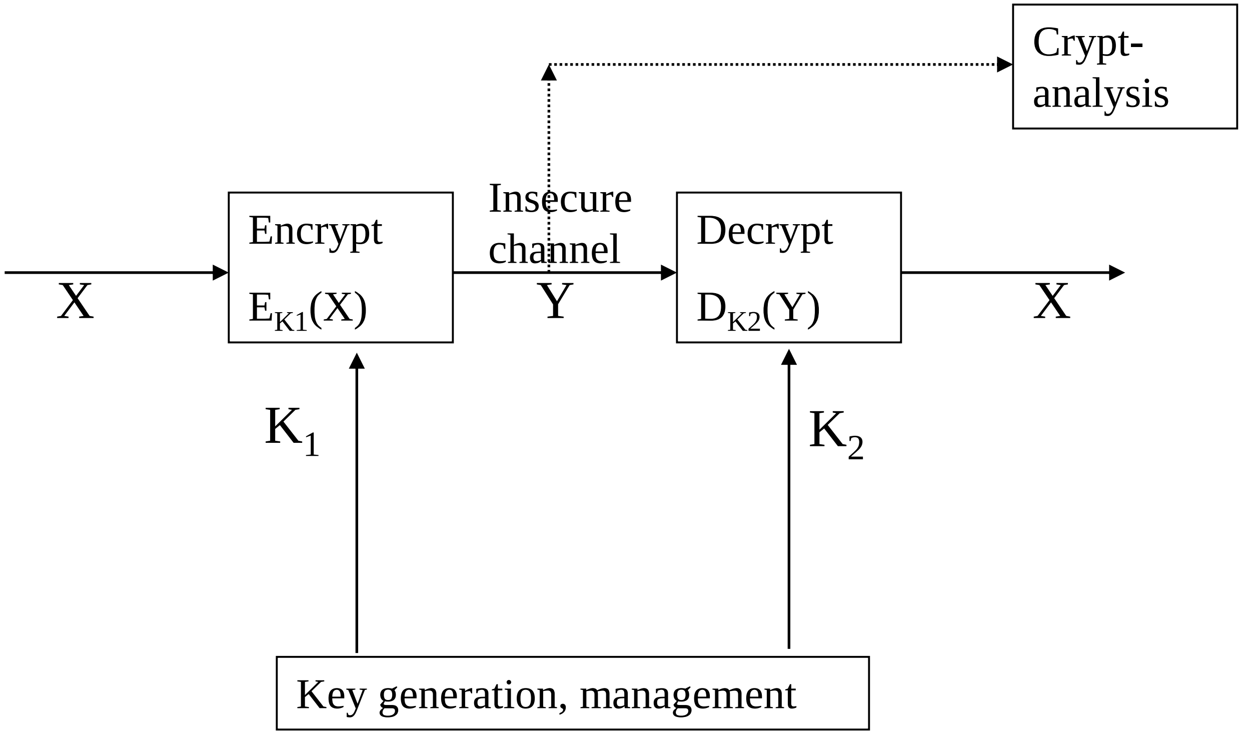
- When Bob wants to send a secure message to Alice,
  - Bob uses Alice's public key to encrypt the message.
  - Alice then uses her private key to decrypt it.
- The public and private keys are related in such a way that only the public key can be used to encrypt messages and only the corresponding private key can be used to decrypt them.

Public-key Encryption

- The keys are related mathematically, but the private key cannot be practically derived from the public key.
  - It is virtually impossible to deduce the private key if you know the public key.
- The only difficulty with public-key systems is that you need to know the recipient's public key to encrypt a message for him or her.
- Therefore What's needed is a global registry of public keys

Asymmetric cryptographic system

- Asymmetric encryption
  - Plaintext, X. Ciphertext, Y
  - Two keys K1, and K2. One is secret, other is public
  - One of them (secret or public) is used to encrypt, the other for decryption



Asymmetric cryptographic system

- There is no key distribution problem in public key cryptography.

- Some people have compared public key cryptography to a mailbox. Many people can put mail into the mailbox (in effect, using the public key), but only a postal worker with the appropriate key (corresponding to the private key) can retrieve the mail from the mailbox.

Asymmetric cryptographic system

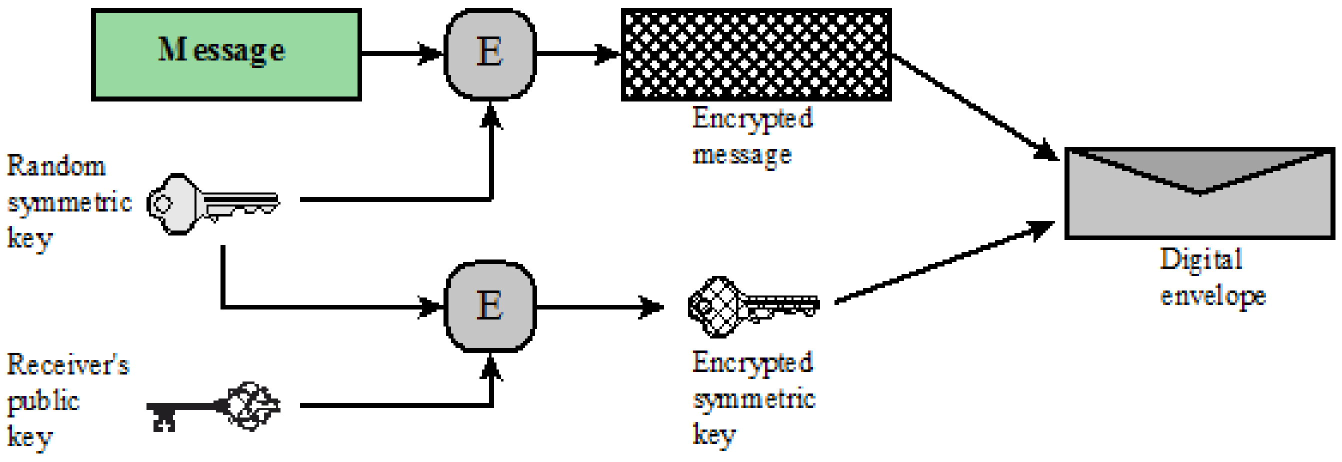
- Public key encryption and decryption algorithms tend to be incredibly slow relative to symmetric key algorithms.

- Public key algorithms tend to be about 100 times slower than DES.

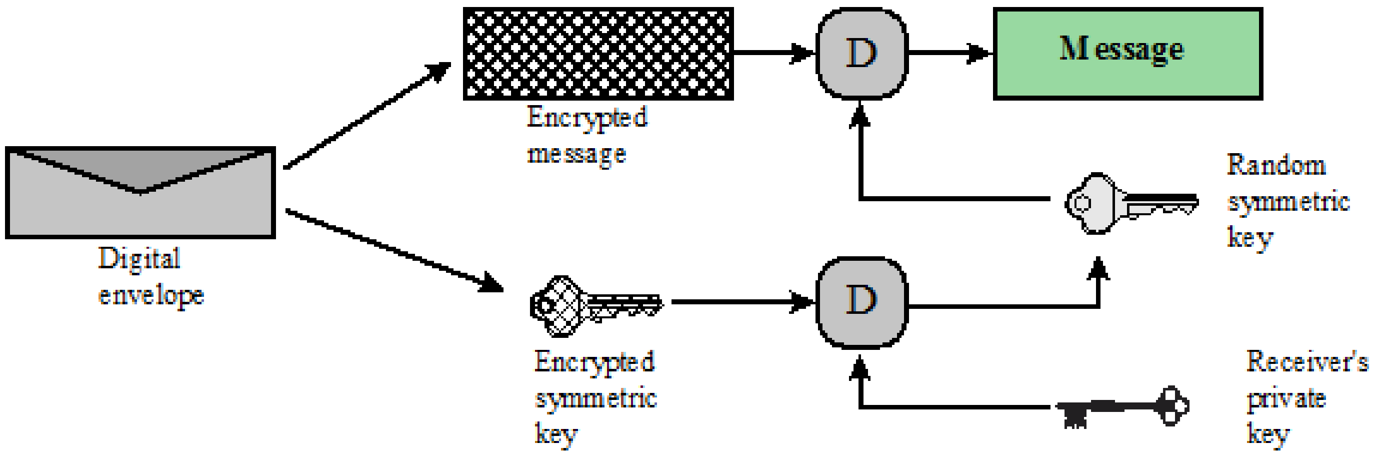
- In general, encrypting large messages using public key cryptography is not considered practical.

Hybrid Encryption Systems

- All known public key encryption algorithms are much slower than the secret-key algorithms.
- In a ***hybrid*** system, Alice uses Bob's public key to send him a secret shared ***session key***.
- Alice and Bob use the session key to exchange information.



(a) Creation of a digital envelope



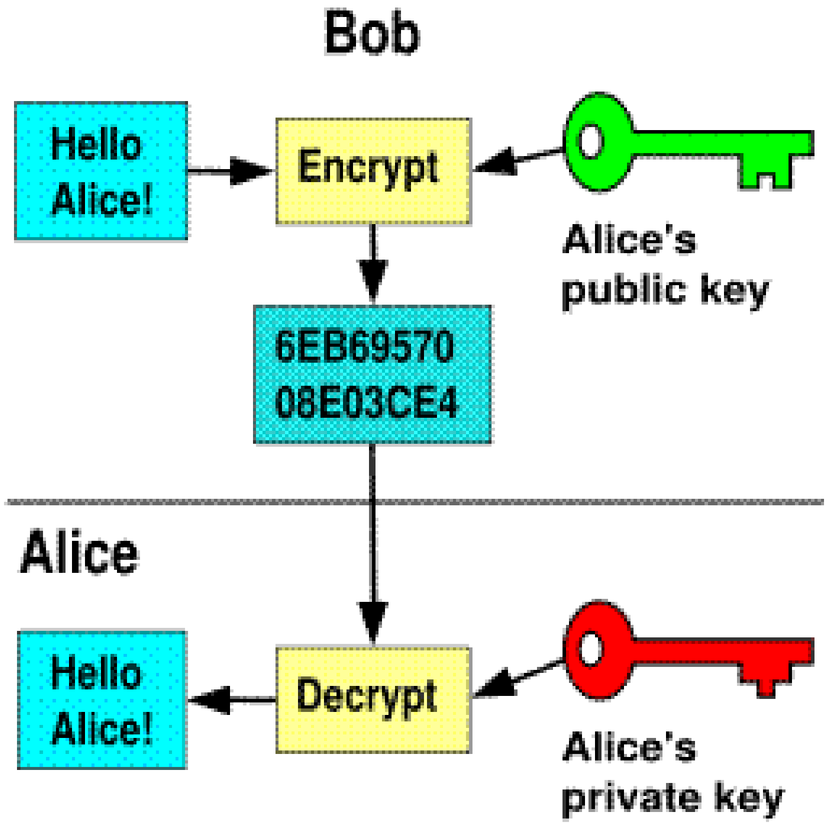
(b) Opening a digital envelope

Cryptography Hash function

- A hash function H is a transformation that takes an input m and returns a fixed size string.
- When employing in cryptography, the function H must have the following properties:
  - The input m can be of any length.
  - The output has a fixed length.
  - The function H(m) is easy to compute for any given m
  - H(m) is a one-way function i.e. it is computationally infeasible to find m such that H(m)=h for any given h
  - H is collision free.

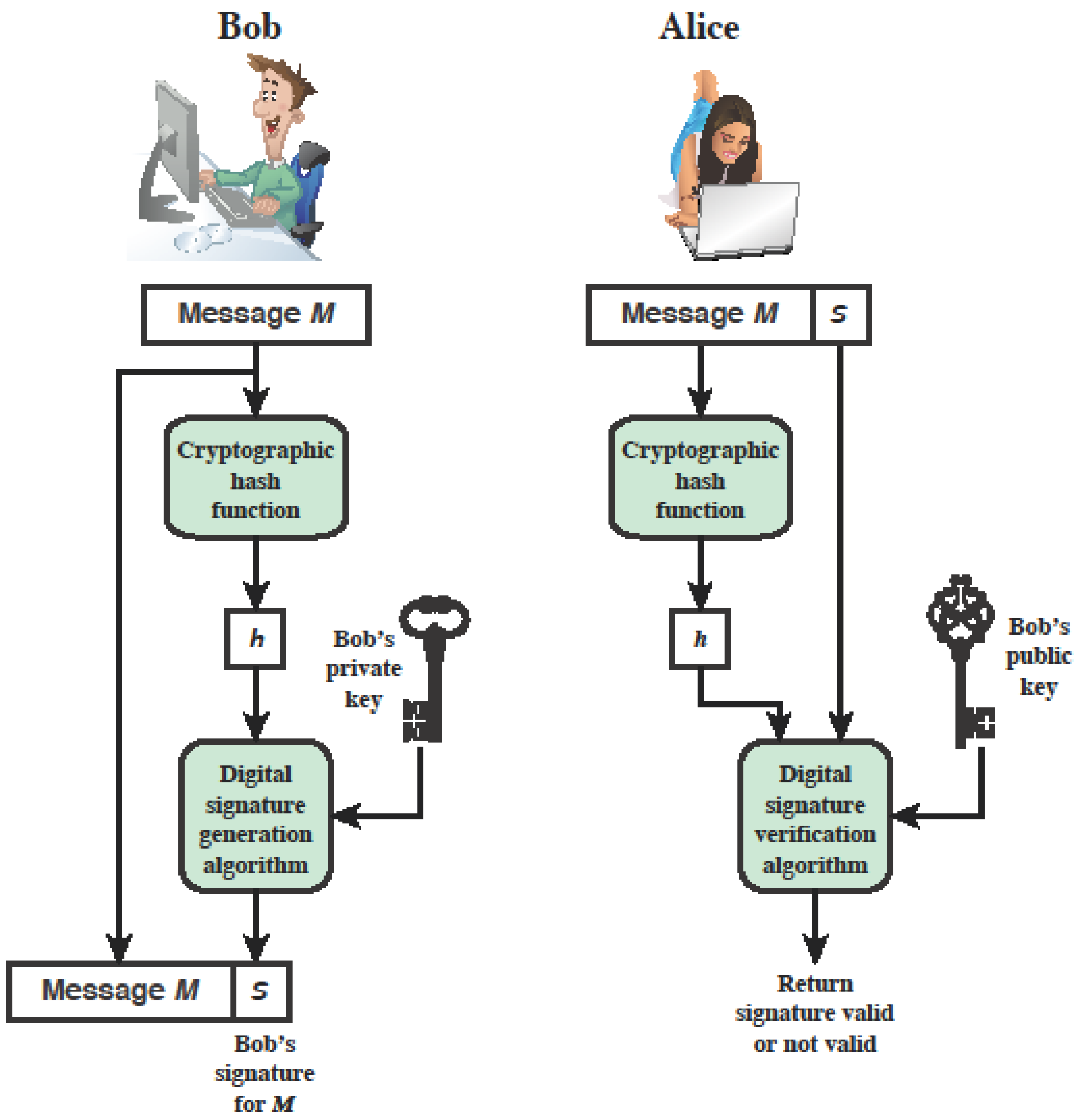
Public-key (Encryption)

Anyone can encrypt using the public key, but only the holder of the private key can decrypt. Secrecy depends on the secrecy of the private key.



Public-key (Digital signature)

Using a private key to encrypt (thus **signing**) a message; anyone can check the **signature** using the public key. Validity depends on private key security.

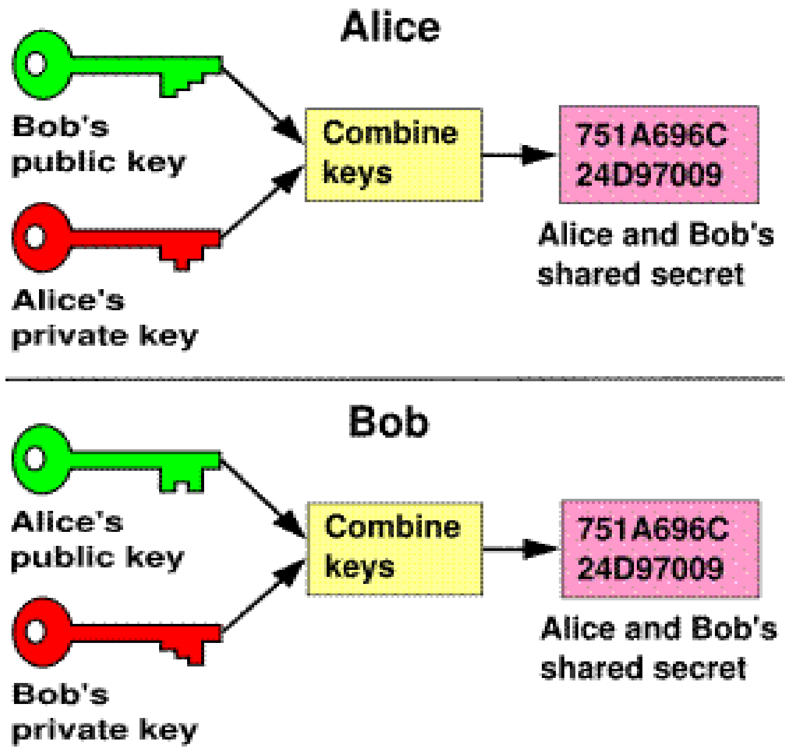


Bob
Alice
Message M
Message M
S
Cryptographic hash function
Cryptographic hash function
h
h
Bob's private key
Bob's public key
Digital signature generation algorithm
Digital signature verification algorithm
Message M
S
Bob's signature for M
Return signature valid or not valid

(a) Bob signs a message

(b) Alice verifies the signature

Public-key (key exchange)



By combining your own private key with the other user's public key, you can calculate a shared secret that only the two of you know. The shared secret can be used as the key for a symmetric cipher.

Public-key cryptography

- The Three main branches of public key cryptography are:
  - Public key encryption — a message encrypted with a recipient's public key cannot be decrypted by anyone except the recipient possessing the corresponding private key. This is used to ensure Confidentiality.
  - Digital signatures — a message signed with a sender's private key can be verified by anyone who has access to the sender's public key, thereby proving that the sender signed it and that the message has not been tampered with. This is used to ensure data origin authentication and data integrity.
  - Key agreement(or key exchange) is a public key technique that allows two or more parties to exchange a secret key over an insecure channel without any prior shared secret.

A trap-door one way functions

A **one-way function** is a function that is “easy” to compute and “difficult” to reverse.

- A trapdoor one-way function is a family of invertible functions f_k, such that

  $Y = f_k(X)$ easy, if k and X are known

  $X = f_k^{-1}(Y)$ easy, if k and Y are known

  $X = f_k^{-1}(Y)$ infeasible, if Y is known but k is not known

OWF: Multiplying two primes

It is easy to take two prime numbers and multiply them together.

- If they are fairly small we can do this in our heads, on a piece of paper, or on a calculator.
  - As they get bigger and bigger it is fairly easy to write a computer program to compute the product.
    - Multiplication runs in polynomial time.
      - Multiplication of two primes is easy.

OWF: Multiplying two primes

Multiplication of two prime numbers is **believed** to be a one-way function.

We say **believed** because nobody has been able to **prove** that it is hard to factorise.

Maybe one day someone will find a way of factorising efficiently.

OWF: Modular exponentiation

The process of **exponentiation** just means raising numbers to a power.

Raising **a** to the power **b**, normally denoted a^b just means multiplying **a** by itself **b** times. In other words:

$$a^b = a \times a \times a \times \ldots \times a$$

Modular exponentiation means computing a^b modulo some other number **n**. We tend to write this as

$$a^b \bmod n.$$

Modular exponentiation is “easy”.

OWF: Modular exponentiation

However, given **a**, **b**, and **a^b mod n** (when **n** is prime), calculating **b** is regarded by mathematicians as a hard problem.

This difficult problem is often referred to as the **discrete logarithm problem**.

In other words, given a number **a** and a prime number **n**, the function

$$f(b) = a^b \bmod n$$

is believed to be a one-way function.

Diffie-Hellman

- The **Diffie–Hellman (DH) key exchange** technique was first defined in their seminal paper in 1976.
- DH key exchange is a method of exchanging public (i.e. non-secret) information to obtain a shared secret.
- **DH is not an encryption algorithm.**

- DH key exchange has the following important properties:
- The resulting shared secret cannot be computed by either of the parties without the cooperation of the other.
- A third party observing all the messages transmitted during DH key exchange cannot deduce the resulting shared secret at the end of the protocol.

Diffie-Hellman (cont.)

- Here is how DH works allowing Alice and Bob to establish a secret message key. Assume that n is some prime number and g is a base number:
  1. Alice generates a secret key, a.
  2. Bob generates a secret key, b.
  3. Alice computes a public key $A = g^a \bmod n$.
  4. Bob computes a public key $B = g^b \bmod n$.
  5. Bob and Alice exchange their public keys.
  6. Alice now computes the message key, K, as $K = B^a \bmod n$
  7. Bob now computes the message key, K, as $K = A^b \bmod n$
  8. Bob and Alice end up with the same key: $K = g^{a * b} \bmod n$

common parameters:

g, *n*: two large primes

Alice choices *a*

Bob choices *b*

$A = g^a \bmod n \longrightarrow$

$\longleftarrow B = g^b \bmod n$

Alice calculate $((B)^a \bmod n)$, result is $(g^{ab} \bmod n)$

Bob calculate $((A)^b \bmod n)$, result is $(g^{ab} \bmod n)$

Session key = $g^{ab} \bmod n$

Diffie-Hellman (Eample)

In practice, very large numbers are used (several hundred DIGITS each), but here is an example using small numbers:

- $n = 11, g = 5, a = 2, b = 3.$
- Alice computes her public key $A = 5^2 \bmod 11 = 3.$
- Bob computes his public key $B = 5^3 \bmod 11 = 4.$
- Bob and Alice exchange their public keys.
- Alice computes the message key, K, as $K = 4^2 \bmod 11 = 5.$
- Bob computes the message key, K, as $K = 3^3 \bmod 11 = 5.$
- Both end up with the same message key, namely:
- $K = 5^{2*3} \bmod 11 = 15{,}625 \bmod 11 = 5.$

Diffie-Hellman (cont.)

- Diffie-Hellman are used in several network protocols and commercial products.
- With Diffie-Hellman, keys can be generated as needed (on the fly) and they can be discarded at the end of the conversation.

RSA

The **RSA** public key encryption algorithm was the first practical implementation of public key encryption discovered.

It remains the most used public key encryption algorithm today.

It is named after the three researchers Ron Rivest, Adi Shamir and Len Adleman who first published it.

Setting up RSA

- Let **n** be the product of two large primes **p** and **q** and $\phi(n)=(p-1)(q-1)$
  - By “large” we typically mean at least 512 bits.
- Select a special number **e**
  - greater than 1 and less than $\phi(n)$ .The precise mathematical property that e must have is that there must be no numbers that divide neatly into e and into $\phi(n)$ except for 1.
- Publish the pair of numbers (**n**,**e**)
- Compute the private key **d** from **p**, **q** and **e**

Computing the private key

The private key **d** is computed to be the unique inverse of **e** modulo $\phi(n)$.

In other words, **d** is the unique number less than $\phi(n)$ that when multiplied by **e** gives you 1 modulo $\phi(n)$.

Written mathematically:

$$ed = 1 \bmod \phi(n)$$

The **Euclidean Algorithm** is the process that you need to follow in order to compute **d**.

Setting up RSA: example

Step 1: Let p = **47** and q = **59**. Thus n = **47** x **59** = **2773**

Step 2: Select e = **17**

Step 3: Publish (n,e) = (**2773**, **17**)

Step 4: $\phi(n)$ = (p-1) x (q-1)

= **46** x **58** = **2668**

Use the Euclidean Algorithm to compute the modular inverse of **17** modulo **2668**. The result is d = **157**

<< Check: **17** x **157** = 2669 = 1(mod **2668**) >>

Public key is (**2773**,**17**)

Private key is **157**

Encryption and decryption

The first job is to represent the plaintext as a series of numbers modulo n.

The encryption process to obtain the ciphertext C from plaintext M is very simple:

$$C = M^e \bmod n$$

The decryption process is also simple:

$$M = C^d \bmod n$$

Encryption and decryption: example

Public key is (**2773**,**17**)

Private key is **157**

Plaintext block represented as a number: M = 31

Encryption using Public Key: $C = 31^{17} \pmod{2773}$

$= \mathbf{587}$

Decryption using Private Key: $M = \mathbf{587}^{157} \pmod{2773}$

$= 31$

Security of RSA

We will look at two different strategies for trying to “break” RSA:

1. Trying to decrypt a ciphertext without knowledge of the private key
2. Trying to determine the private key

Decrypting ciphertext without the key

The encryption process in RSA involves computing the function $C = M^e \bmod n$, which is regarded as being easy.

An attacker who observes this ciphertext *c*, and has knowledge of *e* and *n*, needs to try to work out what *m* is.

i.e., find *m* such that $\boldsymbol{m^e} = c \bmod n$

In other words, find the e^{th} root of *c* mod *n*

- Computing *m* from *c*, *e* and *n* is regarded as a hard problem and known as ***RSA problem***

Obtain the private key from the public key

If the attacker knows the public key of a user (e,n), what would she/he need to do in order to obtain the corresponding private key?

- He/she needs to find *d* such that $ed \bmod \phi(n) = 1$
- *i.e.*, needs to know *p* and *q*
- In other words, he/she must factor *n* (problem of prime factorization)

Recommended size of n:

- 2008:
  - the largest factorized number: 663 bits
  - Typical keys: 1024 and 2048 bits
  - Recommended size of *n* is 2048 bits and larger

Attacking RSA

RSA key length (bits)	Symmetric key length (bits)
1024	80
2048	112
3072	128
15360	256

- RSA claims that 1024-bit keys are likely to become crackable some time between 2006 and 2010 and that 2048-bit keys are sufficient until 2030.
- An RSA key length of 3072 bits should be used if security is required beyond 2030.

RSA security summary

There are two one-way functions involved in the security of RSA.

One-way function	Description
Encryption function	The encryption function is a trapdoor one-way function, whose trapdoor is the private key. The difficulty of reversing this function without the trapdoor knowledge is **believed** (but not known) to be as difficult as factoring.
Multiplication of two primes	The difficulty of determining an RSA private key from an RSA public key is **known** to be equivalent to factoring n. An attacker thus cannot use knowledge of an RSA public key to determine an RSA private key unless they can factor n. Because multiplication of two primes is believed to be a one-way function, determining an RSA private key from an RSA public key is believed to be very difficult.

Cryptographic Key Infrastructure

- Goal: bind identity to key
- Symmetric Cryptography
  - Not possible as all keys are shared
- Public key Cryptography
  - Bind identity to public key
  - Crucial as people will use key to communicate with principal whose identity is bound to key
  - Assume principal identified by an acceptable name

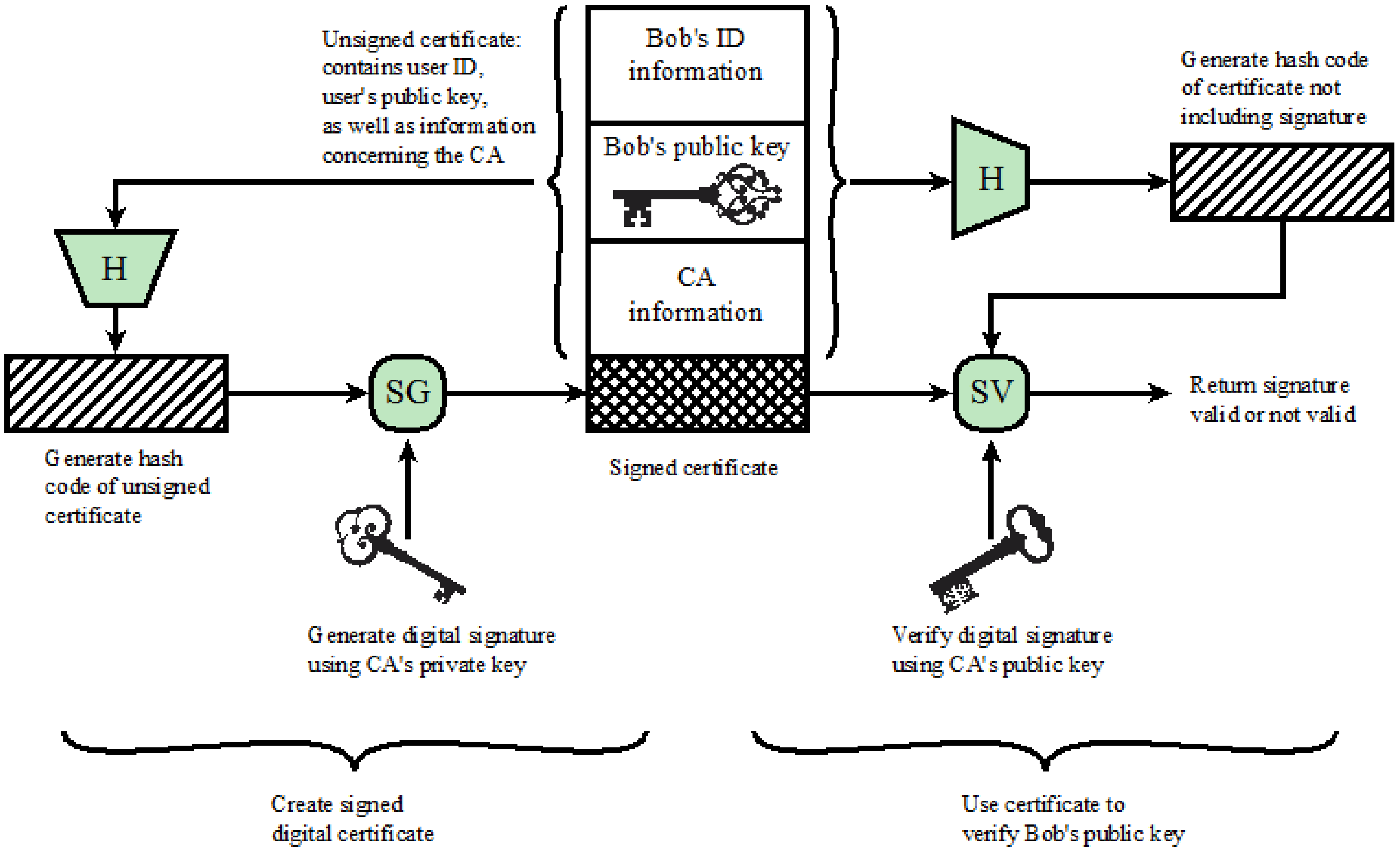
Certificates

- A **certificate** is a token (message) containing
  - Identity of principal (e.g., Alice)
  - Corresponding public key
  - Timestamp (when issued)
  - Other information (perhaps identity of signer)
  - Signature of a trusted authority (e.g., Cathy)

$$C_A = \{\text{Alice}, K_{u\_a}, DS_{kv}(K_{u\_a} \,||\, \text{Alice} \,||\, T)\,\}$$

Kv Cathy's private key

C_A is A's certificate


Unsigned certificate: contains user ID, user's public key, as well as information concerning the CA
Bob's ID information
Bob's public key
CA information
H
Generate hash code of unsigned certificate
SG
Signed certificate
Generate digital signature using CA's private key
H
Generate hash code of certificate not including signature
SV
Return signature valid or not valid
Verify digital signature using CA's public key
Create signed digital certificate
Use certificate to verify Bob's public key

Certificate Use

- Bob gets Alice's certificate
  - If he knows Cathy's public key, he can validate the certificate
    - When was certificate issued?
    - Is the principal Alice?
  - Now Bob has Alice's public key
- ✘ Problem:
  - Bob needs Cathy's public key to validate *Alice*'s certificate
  - Many solutions:
    - Public Key Infrastructure (PKI),
    - Trust-based certificates (PGP)

X.509 certificate

- Key certificate fields in X.509v3: (RFC 5280)
  - Version
  - Serial number (unique)
  - Signature algorithm identifier: hash algorithm
  - Issuer’s name; uniquely identifies issuer
  - Interval of validity
  - Subject’s name; uniquely identifies subject
  - Subject’s public key
  - Signature

- Issuer is called ***Certificate Authority*** (CA)

PKI

- ***A Public Key Infrastructure*** (PKI) is a set of services and policies that lays the framework for binding a public key to an identity and distributing that binding
  - Or in other words, a PKI is a system that provides authentic public keys to applications

- A PKI has three basic processes: Certification, Validation, and Certificate Revocation.