

# Cryptosystems

Bob wants to send a message  $M$  to Alice.

Symmetric ciphers: Bob and Alice both share a secret key,  $K$ .

- $C = E(M, K)$ , Bob sends  $C$
- Alice receives  $C$ ,  $M = D(C, K)$  Use the same key to decrypt.

Public key cipher: Bob has a public key  $Pub_B$  and private key  $Pri_B$ . Alice has a public key  $Pub_A$  and private key  $Pri_A$ .

- $C = E(M, Pub_A)$ , Bob uses Alice's public key to encrypt
- $M = D(C, Pri_A)$ . Alice uses her own secret key to decrypt
- So  $M = D(E(M, Public\_key), Private\_key)$ .
- Usually also true:  $M = D(E(M, Private\_key), Public\_key)$ , so we can provide a signature use my own private key, and the receiver can verify it using my public key.

# The Comparisons

Public Key is 100-1000 times slower than symmetric key.

- Public key are not generally used to encrypt long messages. They are often used to encrypt “keys”
- Many protocols combine them such as SSL.

The algorithms are usually based on different approaches.

- Public key cryptosystems are usually based on simple-to-describe mathematical functions. Their security depends on certain computational problem that are infeasible, or believed to be infeasible such as given a large  $n$ , find two primes  $p$ , and  $q$  such that  $n=p*q$ .
- Symmetric key cryptosystems are usually based on a sequence of simple operations (substitutions and permutations).

# Cryptosystems

Signature: for example using public key systems.

- Bob side:  $S = E(M, \text{private\_key of Bob})$ , then  $C = E(M || S, \text{public\_key of Alice})$ . Sends C.
- Alice side: decrypt C with private key of Alice to get  $M || S$  and then decrypt S again with public key of Bob and then compare it with M.
- Why this message must be sent from Bob? How about data integrity (message authentication)? Can someone modify the message in the middle?
- Problem: the message to be encrypted and decrypted is too long.
- Dispute: bring M and S to the judge.

Another one. Bob side:  $Y = E(M, \text{public\_key of Alice})$ ,  $S = E(Y, \text{Private key of Bob})$ . Sends  $Y || S$

Alice side:  $Y' = D(S, )$ , compare Y with Y',  $M = D(Y, )$ .

- Any potential problem? Repudiation? Alice needs to get the additional encrypted one Y for dispute resolution.

# Hash function

Better to represent a long message by a short bit string, called message digest (eg. 128 bits). This is done through hash function such as MD5.

Bob:  $H = \text{Hash}(M)$ , then  $C = E(M || H, \text{shared secret key})$ .

Hash is fast and  $H$  is very short.

Alice: decrypt  $C$  using shared secret key to get  $M || H$ . Then  $\text{Hash}(M)$  and compare it with  $H$ .

This provide data integrity (message authentication). But how about digital signature?

Bob:  $H = \text{Hash}(M)$ ,  $S = E(H, \text{private key of Bob})$ ,  $C = E(M || S, \text{public\_key of Alice or shared secret key})$ .

Alice: decrypt  $C$  get  $M || S$ , decrypt  $S$  to get  $H$ , and  $\text{Hash}(M)$ , compare this  $\text{Hash}(M)$  with  $H$ .

# Cryptographic Tools

Encryption schemes: are used to achieve confidentiality

Signature schemes: are used to “sign” data/ A signature helps to ensure data integrity and data origin authentication, and it can also provide non-repudiation.

Message authentication codes (MAC): a message authentication code provides data integrity. Usually use secret key to create a “signature” of the message. Both sender and receiver must know this key. It does not permit public verification of the signature or non-repudiation (because both know the key).

Cryptographic hash functions: used to provide random, unpredictable short “digest” for a message. Keyless MAC. It is hard to reverse back from a digest.

# Cryptographic Tools

Key agreement protocols: used to establish a common secret key known to two or more specified parties. Usually this key is to be later used for another cryptographic purpose such as symmetric key encryption or message authentication codes.

Identification schemes: provides entity authentication. Used in both the secret key and public key setting. It is harder to do this in a public key setting- may need third party.

Pseudorandom number generators: used in many cryptographic contexts, for example, in the generation of keys.

# Secure Socket Layer

Practical tools need incorporate many tools. An SSL session is used, for example, to facilitate on-line purchase from a company's web page. Suppose a client (Bob) wants to purchase something from a server (Alice). They need to build up an SSL session.

- First Bob and Alice needs to introduce themselves. Exchange SSL version number and what cryp tools are okay.
- Server Alice authenticates herself to Bob. She sends him her public key PK, signed by a trusted certification authority (CA). Then Bob verifies the CA's signature using the CA's public key (which would have been bundled with the web browser in Bob's computer).
- Alice and Bob will determine two secret keys (K1 and K2): Bob first generates a random master secret, MS, using a pseudo random generator. He encrypt it using Alice's PK. They then both use a hashing function to generate two keys. K1 is used for MAC, and K2 is used for encrypt/decrypt data messages using a symmetric key cryptosystems.