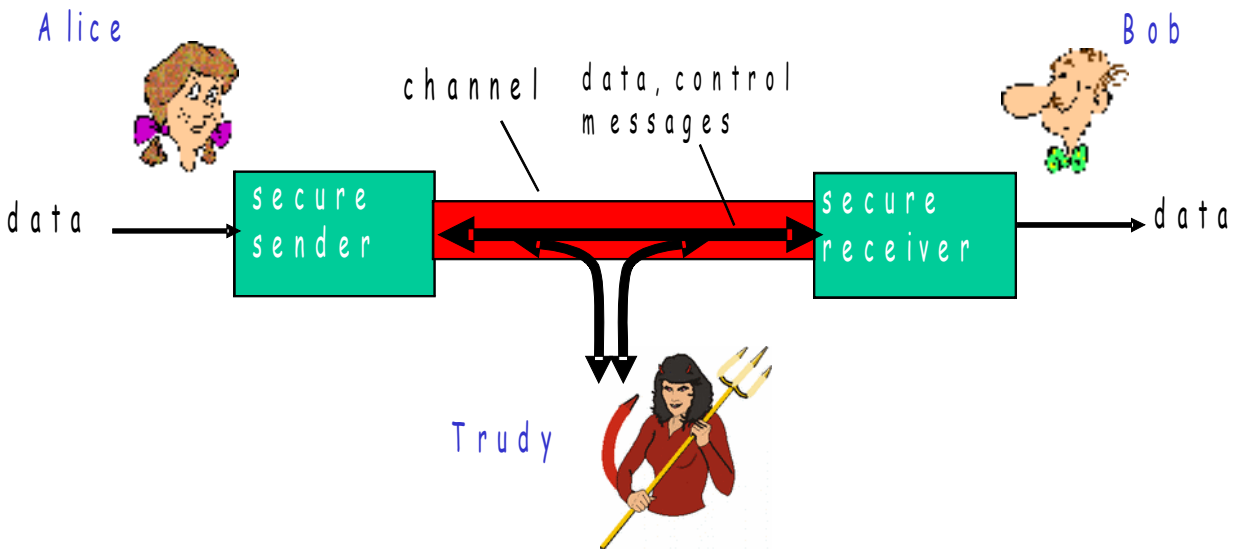
Summary of sections 8.2.1 Symmetric Key Cryptography and 8.2.2 Public Key Encryption[1]

Bob, Alice want to communicate "securely". Trudy (intruder) may intercept, delete, add messages.

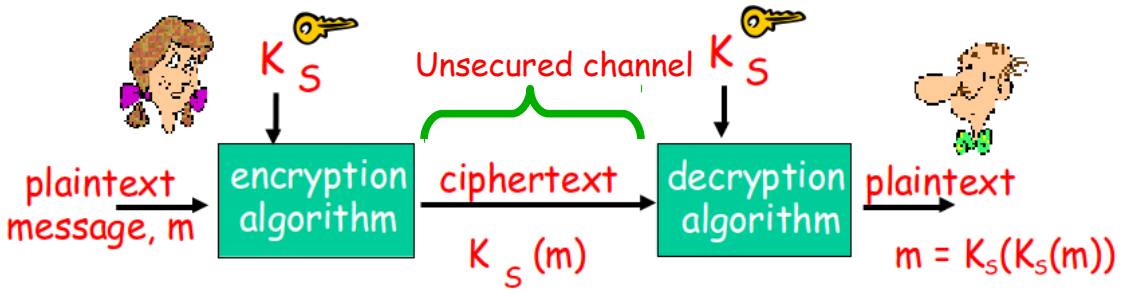


There are two classes of cryptography algorithms: symmetric-key cryptography and public-key cryptography.

8.2.1 Symmetric Key Cryptography

(more information about this topic in section 8.2.1 of the book).

With a symmetric cipher, a message and a key are supplied as input to an encryption algorithm, producing ciphertext. The receiver inputs the ciphertext and key into a decryption algorithm to recover the original plaintext. Importantly, the algorithm is not secret and is known to all. However, the key is known only to the communicating entities.



In the figure above, Bob and Alice share same (symmetric) key: K_s

1 This text and the figures are based on the study guides and slides of the book Computer Networking: A Top Down Approach 5th edition & 6th Edition. Jim Kurose, Keith Ross

Two types of symmetric ciphers:

- Stream ciphers
    - Encrypt one bit at time
- Block ciphers
    - Break plaintext message in equal-size blocks
    - Encrypt each block as a unit

Among the symmetric key algorithms, the textbook focuses on so called "block ciphers," for which DES and 3DES are examples. In a block cipher, the message is chopped into blocks of K bits (for example, K=64 bits) and each block is separately encrypted, with a 1-to-1 mapping. Thus, (ignoring cipher block chaining), the sender inputs a block and the key into the encryption algorithm to generate an encrypted block. The receiver inputs the encrypted block and the same key into a decryption algorithm to decrypt the block. Example with k=3:

input	output	input	output
000	110	100	011
001	111	101	010
010	101	110	000
011	100	111	001

In general, 2k! mappings; huge for k=64

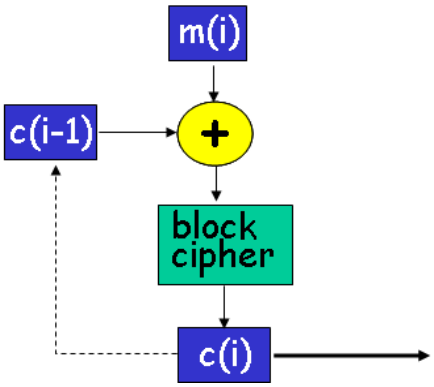
Problem: Table approach requires table with 264 entries, each entry with 64 bits. Instead use function that simulates a randomly permuted table.

Encrypting a large message: If same block of plaintext appears twice, will give same cyphertext.

Solution: Cipher Block Chaining.

Cipher Block Chaining generates its own random numbers

- Have encryption of current block depend on result of previous block
    - c(i) = KS(m(i) XOR c(i-1))
        - XOR ith input block, m(i), with previous block of cipher text, c(i-1)
    - m(i) = KS(c(i)) XOR c(i-1)
- How do we encrypt first block?
    - Initialization vector (IV): random block = c(0)
    - IV does not have to be secret
        - c(0) transmitted to receiver in clear
- Change IV for each message (or session)
    - Guarantees that even if the same message is sent repeatedly, the ciphertext will be completely different each time



DES: Data Encryption Standard

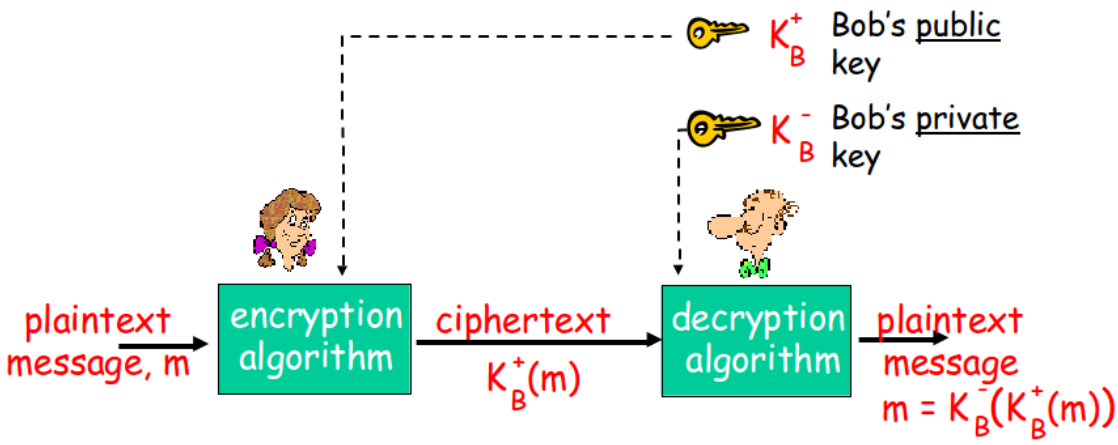
- US encryption standard [NIST 1993]
- 56-bit symmetric key, 64-bit plaintext input
- Block cipher with cipher block chaining
- How secure is DES?
  - DES Challenge: 56-bit-key-encrypted phrase decrypted (brute force) in less than a day
  - No known good analytic attack
- making DES more secure:
  - 3DES: encrypt 3 times with 3 different keys (actually encrypt, decrypt, encrypt)

8.2.2 Public-key Encryption

(more information about this topic in section 8.2.2 of the book).

In a networking environment, with communicating entities often residing in different continents, it is a non-trivial task to distribute the symmetric shared key among the communicating entities. Public-key cryptography can be used for this task. In public-key cryptography, each communicating entity independently generates two keys: a private key and a public key. Each entity makes its public-key available publicly (for example, on a Web page), but keeps its private key private, not showing it to any other entity.

Public key cryptography is a radically different approach [Diffie-Hellman76, RSA78] from symmetric key cryptography. Has two different keys instead of sender and receiver sharing the same secret key: a **public key** that is available to everyone in the world and a **private key** that is known only by the receiver.



Requirements of public key encryption algorithms are:

1. Need K^+_B ( ) and K^-_B ( ) such that $K^+_B(K^-_B(m)) = m$
2. Given public key K^+_B, it should be impossible to compute private key K^-_B

In addition: $K^+_B(K^-_B(m)) = K^-_B(K^+_B(m)) = m \rightarrow$ Result is the same!

Result is the same, but providing different security services:

$K^-_B(K^+_B(m))$: use public key first, followed by private key

- Confidentiality: Only Bob (owner of K^+_B) can decrypt the ciphertext.
- Integrity: No one else, except Bob, can decrypt the ciphertext in order to modify the message and encrypt it again.

$K^+_B(K^-_B(m))$: use private key first, followed by public key

- Sender authentication: ciphertext can only be generated by Bob. Anyway, everybody can decrypt it, since K^+_B is public.
- Integrity: No one else, except Bob, can modify the text and encrypt it again.

RSA (*Rivest, Shamir, Adelson*) is a popular public key encryption algorithm. A message is a bit pattern. A bit pattern can be uniquely represented by an integer number. Thus encrypting a message is equivalent to encrypting a number.

Given the Public key (K^+_B) is the pair (n,e), and the Private key (K^-_B) is the pair (n,d).

1. To encrypt message m (<n), compute

   $c = m^e \bmod n$

2. To decrypt received bit pattern, c, compute

   $m = c^d \bmod n$

Refer to textbook for details on how to calculate the key pairs (n, e) and (n, d). The important result is:

$$m = \overbrace{(m^e \bmod n)}^{c}{}^d \bmod n$$

RSA also has the property: $K^-_B(K^+_B(m)) = m = K^+_B(K^-_B(m))$

$$m = (m^e \bmod n)^d \bmod n = m^{ed} \bmod n = m^{de} \bmod n = (m^d \bmod n)^e \bmod n$$

While encrypting and decrypting, RSA performs several exponentiation operations of large numbers, what is a rather time-consuming process. As a result, RSA is often used in practice in combination with symmetric key cryptography. For example, if Alice wants to send Bob a large amount of encrypted data, she could do the following. Alice first creates a random key, which will be the shared symmetric key. She now needs to send this key to Bob over the network in such a way that no one else can see it. To this end, she encrypts the symmetric key with Bob's public key, and sends the encrypted message to Bob. To decrypt the message and extract the symmetric key, one needs to apply to the message Bob's private key, which only Bob has. Bob decrypts the key, so that Alice and Bob finally share the same symmetric key. Distributing a symmetric key among communicating entities is a common application of public-key cryptography. There are many applications of public-key cryptography, some of which are discussed in the textbook.