CPSC 317
COMPUTER NETWORKING

Module 8: Security – Day 2 - Encryption

1

Some slides based on Kurose/Ross original slides, found at https://gaia.cs.umass.edu/kurose_ross/ppt.htm

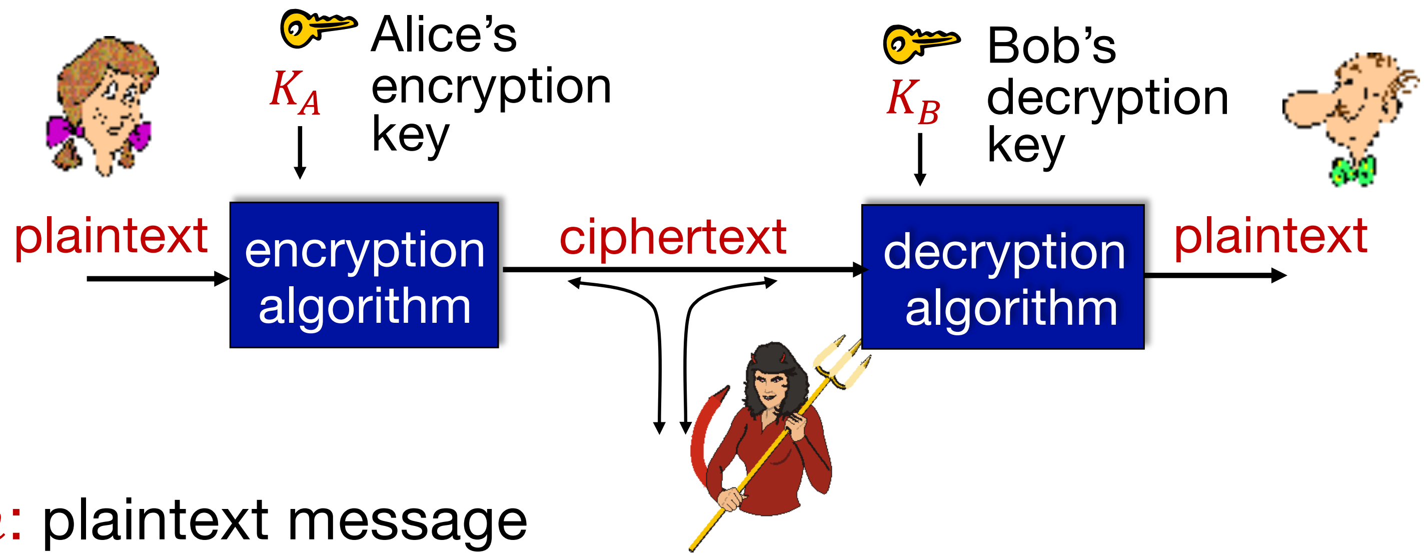
LEARNING GOALS

- Explain different classes of attacks on cryptography schemes
- Explain and use substitution ciphers
- Explain the uses and limitations of shared key or symmetric cryptography
- Explain how keys can be exchanged using Diffie-Hellman protocol

READING

- Reading: 8.3

THE LANGUAGE OF CRYPTOGRAPHY

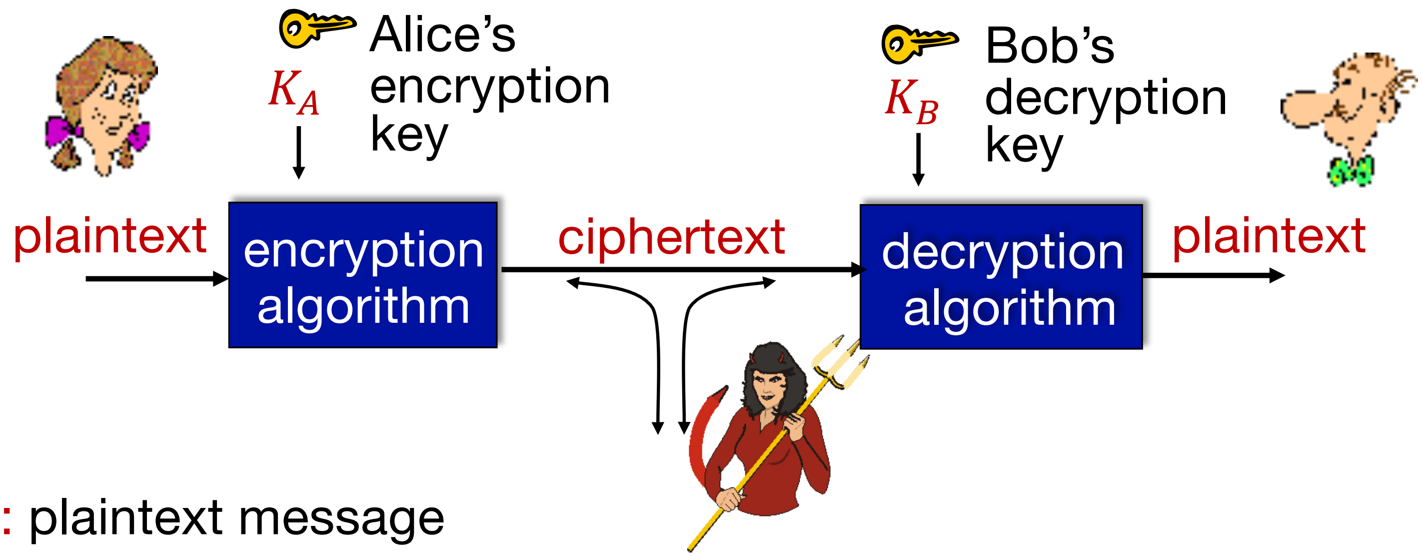


m: plaintext message

$K_A(m)$: ciphertext, encrypted with key K_A

$K_B\left(K_A(m)\right) = m$

THE LANGUAGE OF CRYPTOGRAPHY (ALTERNATE NOTATION)



m: plaintext message

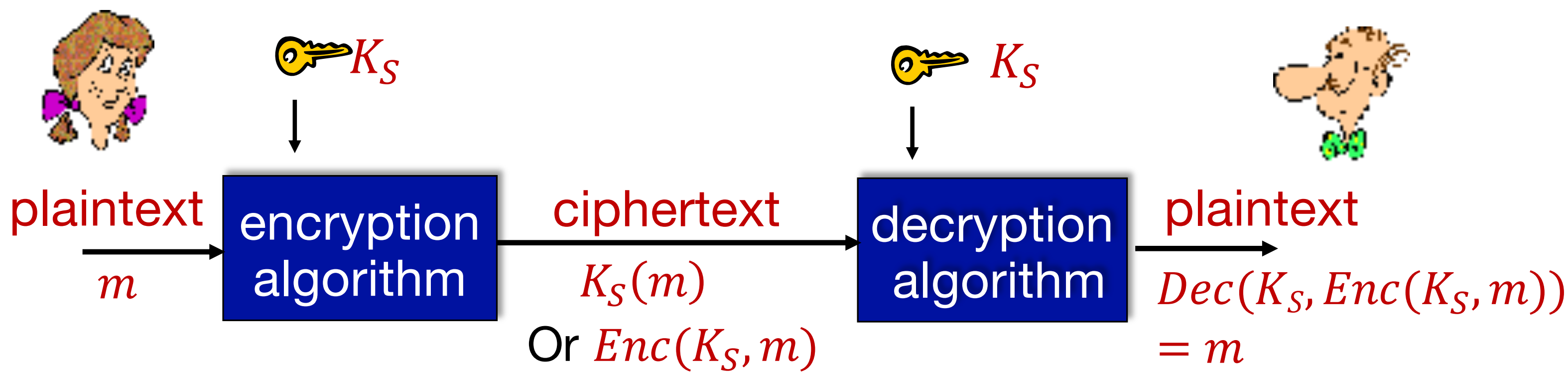
$Enc(K_A, m) = m'$: run algorithm Enc on m with key K_A to generate cipher m'

$Dec(K_B, m') = Dec(K_B, Enc(K_A, m)) = m$: run algorithm Dec with key K_B on cipher m' to retrieve m

BREAKING AN ENCRYPTION SCHEME

- Ciphertext-only attack:
  - Trudy has ciphertext, but not plaintext (e.g., knows $K_A(m)$ but not m)
  - Option 1: brute force, search through all keys
  - Option 2: statistical analysis (look for patterns)
- Known-plaintext attack:
  - Trudy has some ciphertext with its plaintext (e.g., for some m it knows $K_A(m)$), wants to break other ciphertexts
- Chosen-plaintext attack:
  - Trudy has the ability to encrypt any plaintext (e.g., knows K_A, or can trick Alice into encrypting any message), but doesn't have key for decryption

SYMMETRIC KEY CRYPTOGRAPHY



- Symmetric key cryptography: Bob and Alice share same (symmetric) key: K_S
  - e.g., key is knowing substitution pattern in mono alphabetic substitution cipher
  - Method may be different (opposite) for decryption, but uses same key

SIMPLE ENCRYPTION SCHEME

- ***Substitution Cipher***: substitute one thing for another
  - "Thing" can be a byte, block, word, etc.
  - Monoalphabetic cipher: substitute one letter for another
  - Encryption key: mapping from one set to another
  - Example:

```
abcdefghijklmnopqrstuvwxyz
↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓
mnbvcxzasdfghjklpoiuytrewq
```

- Similar to the Caesar cipher, but the mapping is less regular

CLICKER QUESTION

Assuming the encryption mapping below:

```
abcdefghijklmnopqrstuvwxyz
↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓
mnbvcxzasdfghjklpoiuytrewq
```

Decrypt the following message:

LKUMUK

SLIGHTLY BETTER ENCRYPTION

- Several substitution ciphers (e.g., $M_1, M_2, \ldots, M_n$)
- Predictable pattern of ciphers, e.g.,
  - Cycling pattern (e.g., $M_1 \rightarrow M_3 \rightarrow M_3 \rightarrow M_2$)
  - Algorithm that decides next pattern
- For each new symbol, use next substitution pattern
- Encryption key: all ciphers, plus pattern

```
Plaintext: abcdefghijklmnopqrstuvwxyz
           ↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓
      M1:  mnbvcxzasdfghjklpoiuytrewq
      M2:  gclafmvqjdkerouwtisbphynxz
      M3:  xyzuvwrstpqmnojklghidefabc
```

CLICKER QUESTION

Assuming the encryption mapping below, where patterns M_1 and M_2 alternate (M_1 first):

```
Plaintext: abcdefghijklmnopqrstuvwxyz
           ↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓
      M1:  mnbvcxzasdfghjklpoiuytrewq
      M2:  gclafmvqjdkerouwtisbphynxz
```

Decrypt the following message:

KOSUJS

BLOCK CIPHERS

- Message is broken into blocks (e.g., 64-bit blocks)
- Each block is encrypted/decrypted separately
- Encryption method can be as simple as a substitution cipher
  - Substitution table for 64-bit blocks would require 2^{64} entries!
- An algorithm can create a substitution table based on a given key

BLOCK CIPHERS PLUS…

- Keeping the same substitution can be risky
  - Allows for statistical analysis of common substitutions
- To avoid this we can change the substitution for every block
- Option 1: change the key every time (e.g., cyclic pattern)

CIPHER-BLOCK CHAINING (CBC)

- Option 2: Do an additional operation with the plaintext
  - Viable if both parties know what the operation is
- First block is XOR'ed with an arbitrary (randomly chosen) number known by both parties (initialization vector or IV) and then encrypted using a substitution cipher with K_s
- Following blocks are XOR'ed with previous block, then encrypted
  - C[0]: IV
  - $C[1] = K_s(M[0] \oplus C[0])$
  - $C[i+1] = K_s(M[i] \oplus C[i])$
- Decryption: apply the (reverse) substitution using K_s, then XOR with previous block

DES: DATA ENCRYPTION STANDARD

- 56-bit symmetric key, 64-bit plaintext input blocks
  - Block cipher: substitution derived from symmetric key
  - Cipher block chaining: initial vector derived from symmetric key
- Not considered secure any longer
  - DES challenge: 56-bit-key encrypted phrase decrypted with brute force (1997 – 96 days, 1998 – 41 days then 56 hours, 1999 – 22 hours)
- 3DES: more secure
  - Encrypt 3 times with 3 different keys

AES: ADVANCED ENCRYPTION STANDARD

- Symmetric key, replaced DES as NIST standard in 2001
- 128-bit block cipher
- 128-, 192- or 256-bit key
  - Difference is just the number of rounds of translation
- Way more secure than DES
  - Brute force decryption that takes 1 second for DES would take 149 trillion years for 128-bit AES

WHO KNOWS THE KEY?

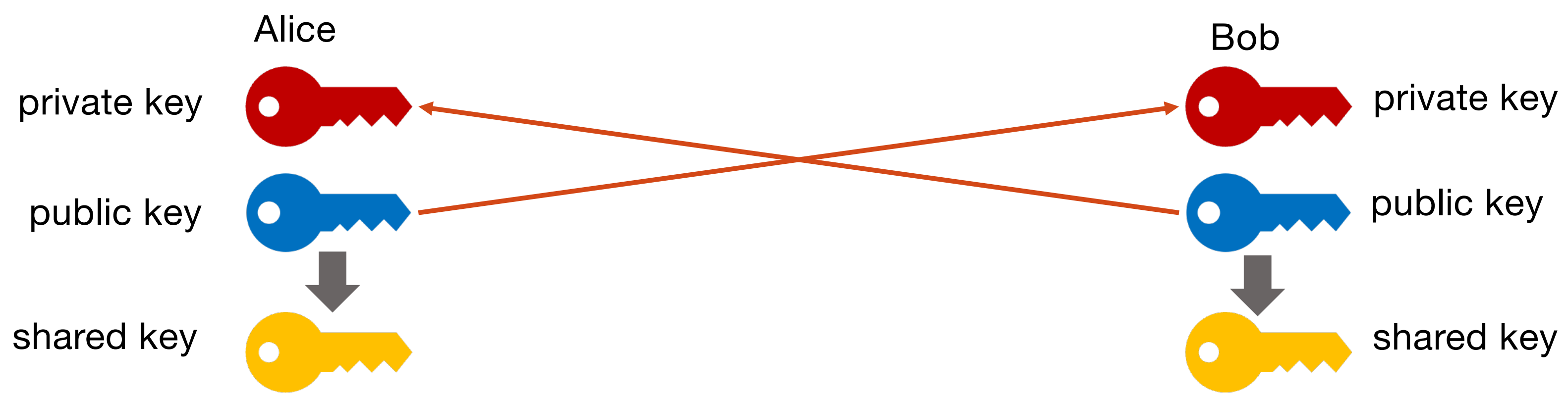
- Both receiver and sender must know the key
  - Sender needs it for encryption
  - Receiver needs it for decryption
- Each connection pair must have its own key
  - Sharing keys with other peers dilutes the trust

WHO KNOWS THE KEY?

- What if sender and receiver never negotiated a key before?
- Key can be generated when a connection first starts
  - How can peers share the key with each other?

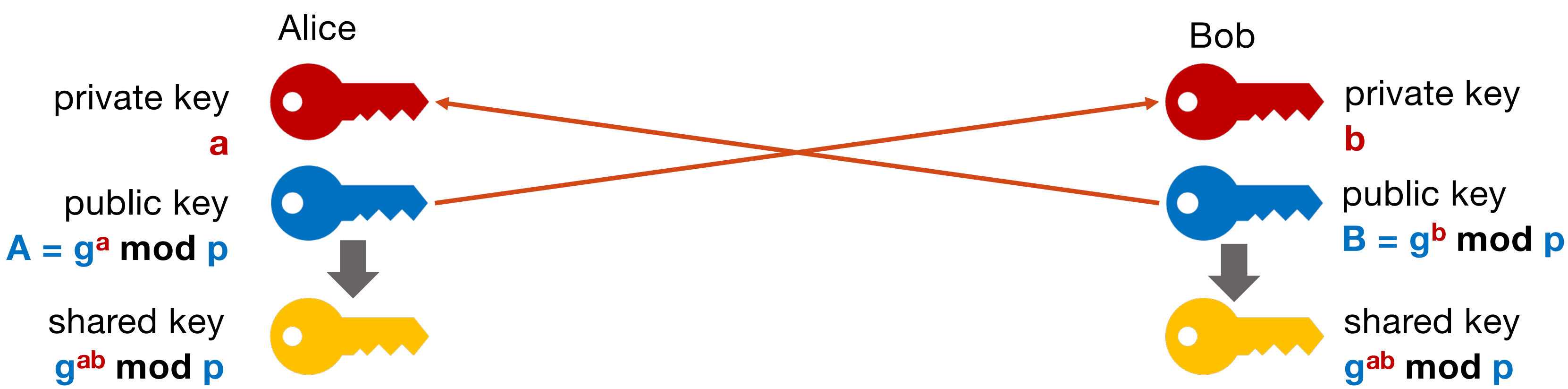
KEY EXCHANGE

- Two parts of a key: public and private
- Combine public key of Alice with private key of Bob and vice versa
- You can easily get public key from private key but you cannot get the private key from the public key alone



DIFFIE-HELLMAN KEY EXCHANGE

- Exponentiation modulo algorithm
- Alice and Bob agree on key generators:
  p (prime number) and **g** (exponentiation base)



IN-CLASS ACTIVITY

- ICA82