CPSC 317
COMPUTER NETWORKING

Module 8: Security – Day 3 – Asymmetric Encryption

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

ADMINISTRATION

- PA5 due Apr 7th
- Please fill up the SEI survey (due Apr 15, 2024)
  - Your feedback will be valuable in improving future offering of the course
  - Read guidelines for how to provide feedback: https://seoi.ubc.ca/resources/resources-for-students/

LEARNING GOALS

- Describe the differences between shared key and private/public key cryptography (a.k.a. symmetric vs. asymmetric cryptography)
- Describe the power of public/private key cryptography
- Describe the message authentication problem and several solutions to it
- Explain what a MAC is and what it does
- Explain the role and use of a certificate

READING

- Reading: 8.4

ASYMMETRIC CRYPTOGRAPHY

- Some algorithms use a pair of keys (e.g., RSA)
- If one key is used for encryption, the other is used for decryption

$$K_1\big(K_2(m)\big) = K_2\big(K_1(m)\big) = m$$

- You can generate a pair of keys, but one key cannot be obtained from the other in reasonable computation time
- One key is kept private, one key is made public
  - Usually denoted K^- and K^+, respectively

CLICKER QUESTION

If Alice encrypts a message with Bob's public key, who can decrypt the message?

A. Alice
B. Bob
C. Alice and Bob
D. Alice, Bob and Trudy (and everybody else)

CLICKER QUESTION

If Alice encrypts a message with Alice's private key, who can decrypt the message?

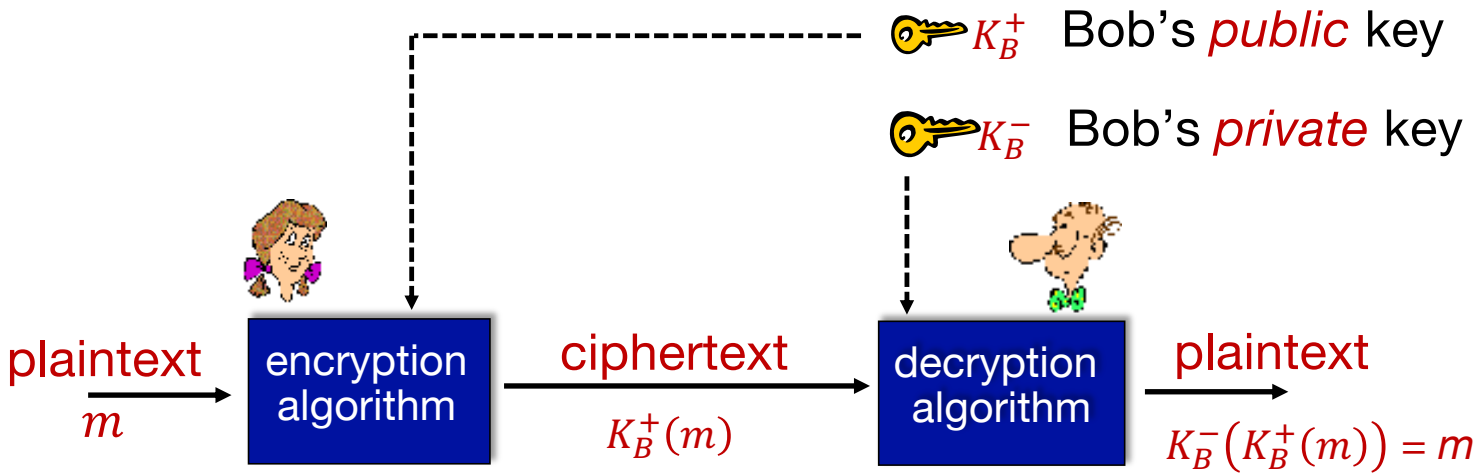
A. Alice
B. Bob
C. Alice and Bob
D. Alice, Bob and Trudy (and everybody else)

ASYMMETRIC CRYPTOGRAPHY

- If sender encrypts with public key, only private key can decrypt it
  - Used for confidentiality
  - $K^-\left(K^+(m)\right) = m$
  - Alternatively, $Dec(K^-, Enc\left(K^+, (m)\right)) = m$
- If owner encrypts with private key, public key can decrypt it
  - Used for authentication, so call sign/verify instead of encrypt/decrypt
  - $K^+\left(K^-(m)\right) = m$
  - Alternatively, $Dec(K^+, Enc\left(K^-, (m)\right)) = m$ OR

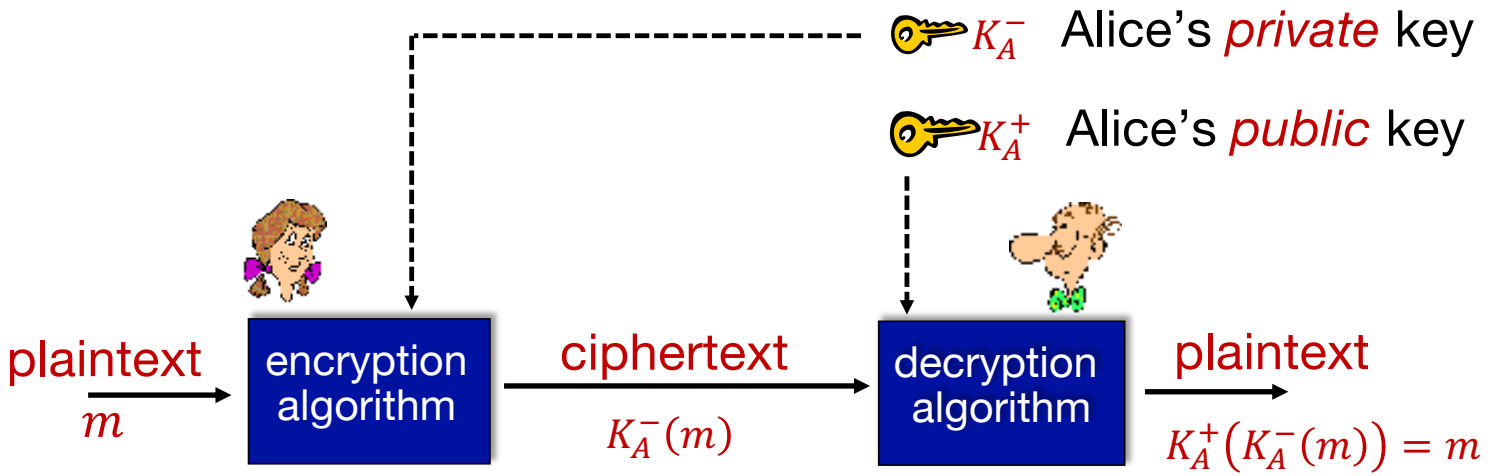
$$Verify(K^+, Sign\left(K^+, (m)\right)) = m$$

PUBLIC KEY CRYPTOGRAPHY (CONFIDENTIALITY)



- Similar ideas emerged at roughly same time, independently in US and UK (classified)

PUBLIC KEY CRYPTOGRAPHY (AUTHENTICATION)



- It works the other way around, too

ASYMMETRIC ENCRYPTION ALGORITHMS

- Need $K_B^+(x)$ and $K_B^-(x)$ such that $K_B^-\left(K_B^+(m)\right) = m$
- The opposite is ideal for authentication, but not required for encryption
- Given public key K_B^+, it should be computationally unfeasible to compute K_B^-

RSA

- Named after Ron **R**ivest, Adi **S**hamir, Leonard **A**dleman
- Relies on modular arithmetic
  - All values computed as remainders of division by common divisor
- Encryption is based on mathematical operations on an integer

RSA: CREATING KEY PAIR

- Choose two large prime numbers $\boldsymbol{p}, \boldsymbol{q}$ (e.g., 1024-bits each)
- Compute $\boldsymbol{n} = pq, \boldsymbol{z} = LCM(p-1)(q-1)$
- Choose $1 < \boldsymbol{e} < z$ that has no common factors with z
- Choose $\boldsymbol{d} < z$ such that $ed \bmod z = 1$
- Public key is $\mathrm{K}_\mathrm{B}^+ = (n, \boldsymbol{e})$, private key is $\mathrm{K}_\mathrm{B}^- = (n, \boldsymbol{d})$
- To encrypt message m (i.e.,$\mathrm{K}_\mathrm{B}^+(m)$), compute $c = m^e \bmod n$
- To decrypt ciphertext c (i.e.,$\mathrm{K}_\mathrm{B}^-(\mathrm{K}_\mathrm{B}^+(m))$), compute $m = c^d \bmod n$

WHY IS RSA SECURE?

- Suppose you know the public key (n, e), you want to determine d
- This requires you to compute the factors of n, which is a hard problem
- Bonus feature:

$$K_B^-\left(K_B^+(m)\right) = m = K_B^+\left(K_B^-(m)\right)$$

MESSAGE INTEGRITY

- Problem: Sender needs to prove authenticity of a document
- The recipient wants to know that:
  - The sender sent the document
  - The document hasn't been changed since it was sent

MESSAGE INTEGRITY: SIGNATURES

- Simple digital signature for message m:
  - Bob signs m by encrypting with his private key: $K_B^-(m)$
  - Bob sends both the original message m and the signature $K_B^-(m)$
  - Alice can check if $\mathrm{m} = K_B^+\left(K_B^-(m)\right)$: if they match, message was signed with K_B^-

MESSAGE DIGEST

- Computing the signature by applying a private key to a long message is computationally expensive
- Alternative: compute a fixed-length "fingerprint"
  - Apply hash function H to message m, giving a fixed size message digest, $H(m)$

SIGNED MESSAGE DIGEST

- Instead of signing entire message, sign only the hash result
  - Bob sends message m and signed digest $K_B^-\big(H(m)\big)$
  - Alice receives m and computes $H_{new}(m)$
  - Alice receives $K_B^-\big(H(m)\big)$ and computes $K_B^+\left(K_B^-\big(H(m)\big)\right)$
  - If $K_B^+\left(K_B^-\big(H(m)\big)\right) = H_{new}(m)$, message is considered signed

SECURE HASH

- Must find hash function such that:
  - Given x, it is computationally infeasible to find m such that $H(m) = x$
  - Given m, it is computationally infeasible to find $m' \neq m$ such that $H(m) = H(m')$

SIMPLE CHECKSUMS AS HASH

- Simple checksums have some properties of hash
  - Produce fixed-length digest of message
- But given a message, it's easy to find another message with same hash value
- For example, if hash is the XOR of all characters:
  - $H(\text{"IOU100.99BOB"}) = H(\text{"IOU900.19BOB"})$

BETTER HASH FUNCTION ALGORITHMS

- MD5 hash function widely used
  - Compute a 128-bit message digest in 4-step process
  - Hard to compute message from hash
- SHA-1 is more secure
  - US NIST standard
  - 160-bit digest
- Newer alternatives: SHA-256, SHA-512

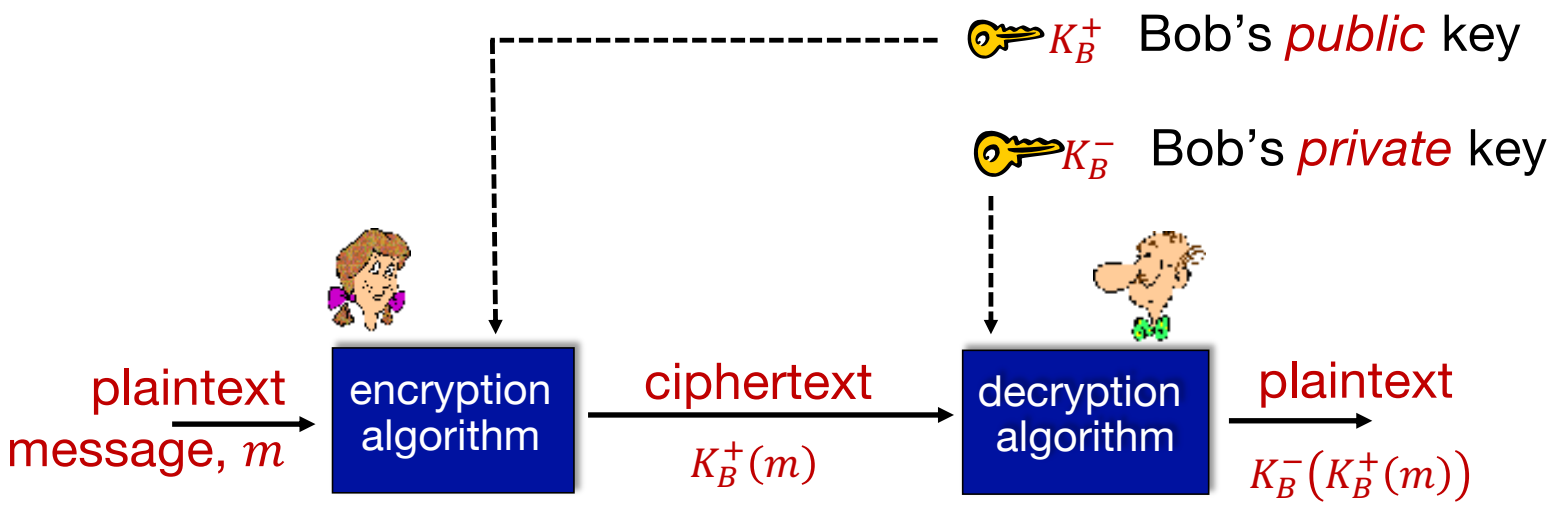
MESSAGE AUTHENTICATION CODE (MAC)

- Alternative to signed message digest
- Shared secret s is used between parties
  - Similar to symmetric cryptography
- Hash is computed not on message m, but on $m + s$
  - Bob sends message m and $h = H(m + s)$
  - Alice receives (m, h) and computes $H(m + s)$
  - If $h = H(m + s)$, message is considered signed
- Faster, since no encryption necessary
- HMAC: popular MAC standard

CONFIDENTIALITY, INTEGRITY, AUTHENTICATION

- Sometimes integrity without confidentiality is acceptable
- Confidentiality without integrity is not useful in practice
  - CBC vulnerability?
- Authenticated encryption
  - MAC-then-Encrypt (SSL)
  - Encrypt-then-MAC (IPSec)
  - Encrypt-and-MAC (SSH)

HOW DO YOU TRUST THE KEYS?



What if Alice accidentally uses someone else's public key (Trudy's) rather than Bob's?

CERTIFICATION AUTHORITY (CA)

- Entity (person, website, etc.) registers public key with CA, provides some "proof of identity"
  - This process is usually performed offline
- Certification Authority provides a certificate:
  - Binds public key to particular entity
  - Signed by CA's private key
- CA's public key is known ("trusted") to public

CHECKING A CERTIFICATE

- When Alice wants Bob's public key:
  - Bob (or someone else) provides a certificate
  - Certificate is signed by CA
  - Alice applies CA's public key to confirm certificate's authenticity
- Certificate contains Bob's public key

ASYMMETRIC CRYPTOGRAPHY: PROBLEMS

- Main problem: asymmetric algorithms are expensive
  - Encryption and decryption methods are computation-heavy
  - Fine for occasional communication
  - Too slow for extensive data transfer
- Solution: use asymmetric keys only to establish secure connection and for authentication
  - Messages sent after key exchange are encrypted/decrypted with shared key (“session key”)
  - Session key is exchanged with asymmetric crypto (e.g., RSA) or generated through a common algorithm (e.g., Diffie-Hellman)

IN-CLASS ACTIVITY

- ICA83