
COM 5335 Network Security

Lecture 5

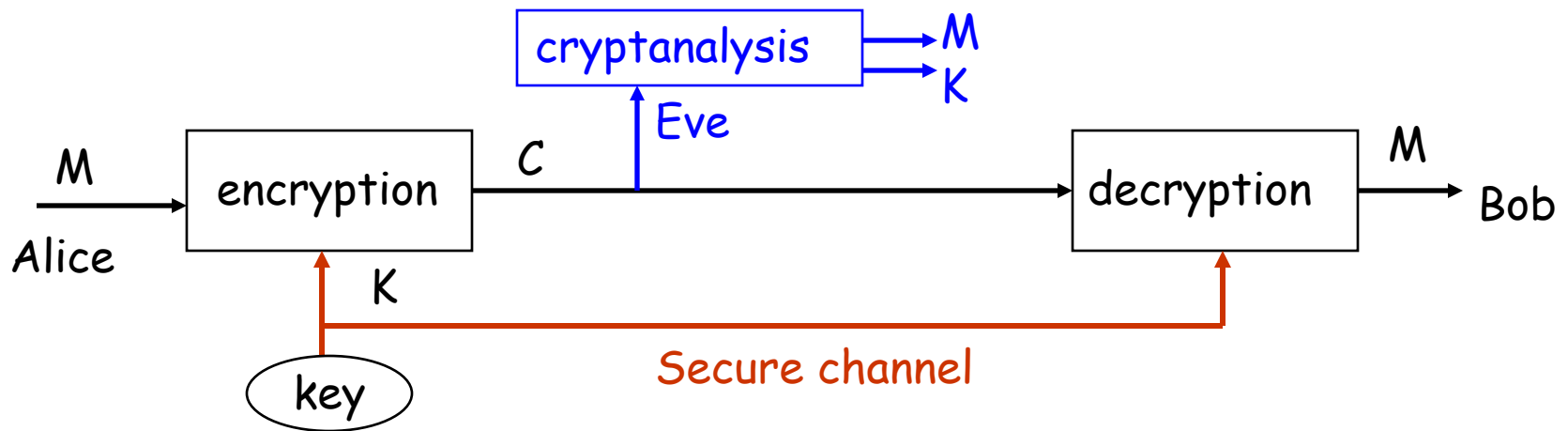
Introduction to Public-Key Cryptography

Scott CH Huang

Outline

- Symmetric Cryptographic System
- Key Management
- Centralized Key Management
- Public-Key Encryption
- Public-Key Cryptographic System
- Public-Key vs. Symmetric Key
- Digital Signature

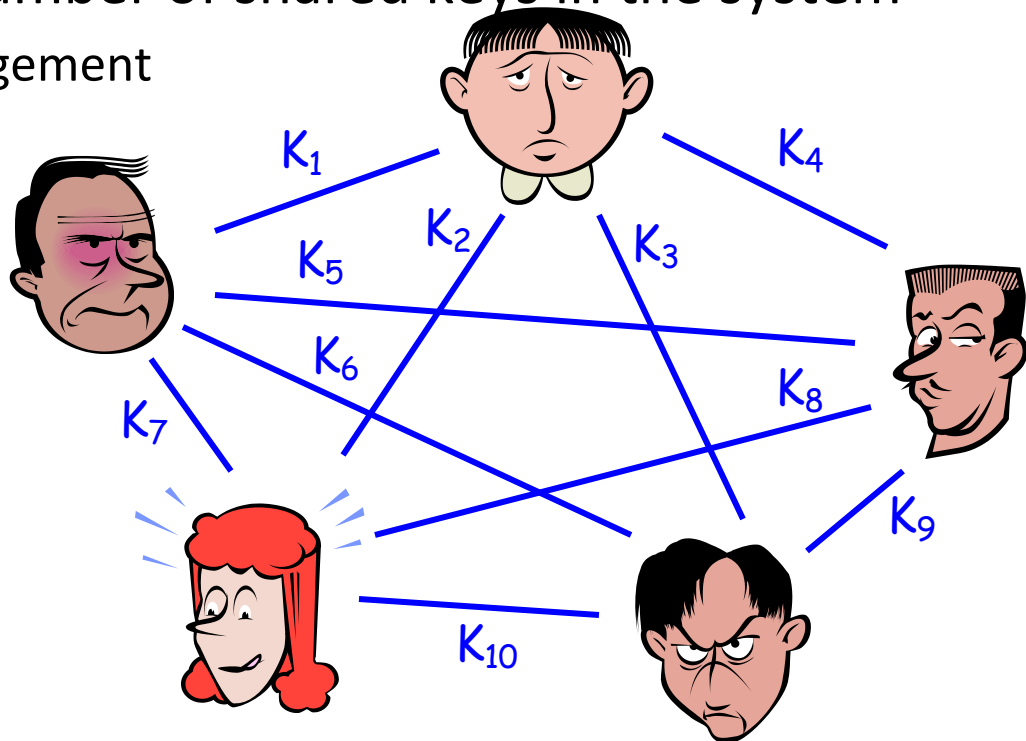
Symmetric (Private-Key) Cryptosystems



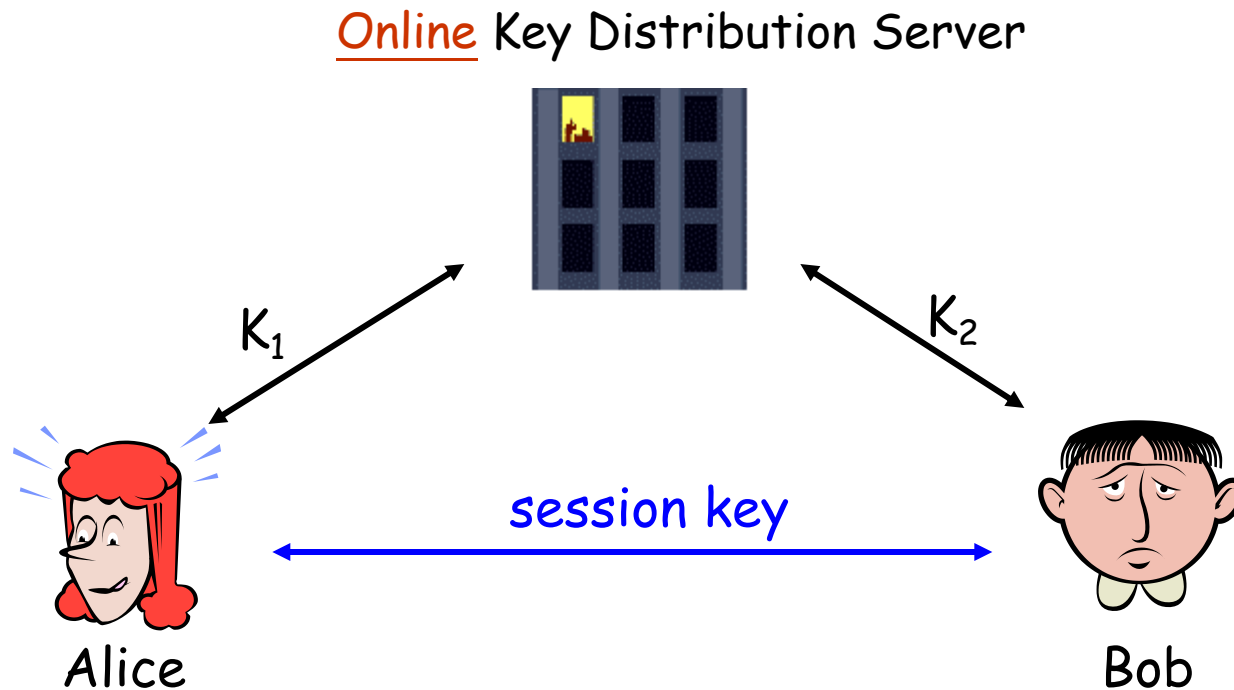
- Alice: sender
- Bob: receiver
- Eve: eavesdropper / Oscar : opponent
- Ciphertext $C = E_K(M)$
- Plaintext $M = E_K^{-1}(C)$
- One of the greatest difficulties: key management
- Algorithms: DES, IDEA, RC2/4/5/6, AES, ...

Symmetric Key Management

- Each pair of communicating entities needs a shared key
 - Why?
 - For an n -party system, there are $n(n-1)/2$ distinct keys in the system and each party needs to maintain $n-1$ distinct keys.
- How to reduce the number of shared keys in the system
 - Centralized key management
 - Public keys



Centralized Key Management



- Only n keys, instead of $n(n-1)/2$ in the system.
- The server may become the single-point-of-failure and the performance bottleneck.

Asymmetric (Public-Key) Cryptosystems

- First proposed in public by Diffie and Hellman at Stanford University in 1976.
 - known earlier in classified community
- Enable secure message exchange
 - between sender and receiver
 - without ever having to meet in advance to agree on a common secret-key.
- It is **asymmetric** because
 - Those who encrypt messages or verify signatures **may not be able to** decrypt messages or create signatures

Public-Key Cryptography

- Probably most significant advance in the 3000 year history of cryptography
- It uses **two** keys – a public & a private key
- **It is asymmetric:** parties are **not** equal
- It uses clever applications of number theoretic concepts to function
- It complements **rather than** replaces private key cryptography

Public-Key Cryptosystems

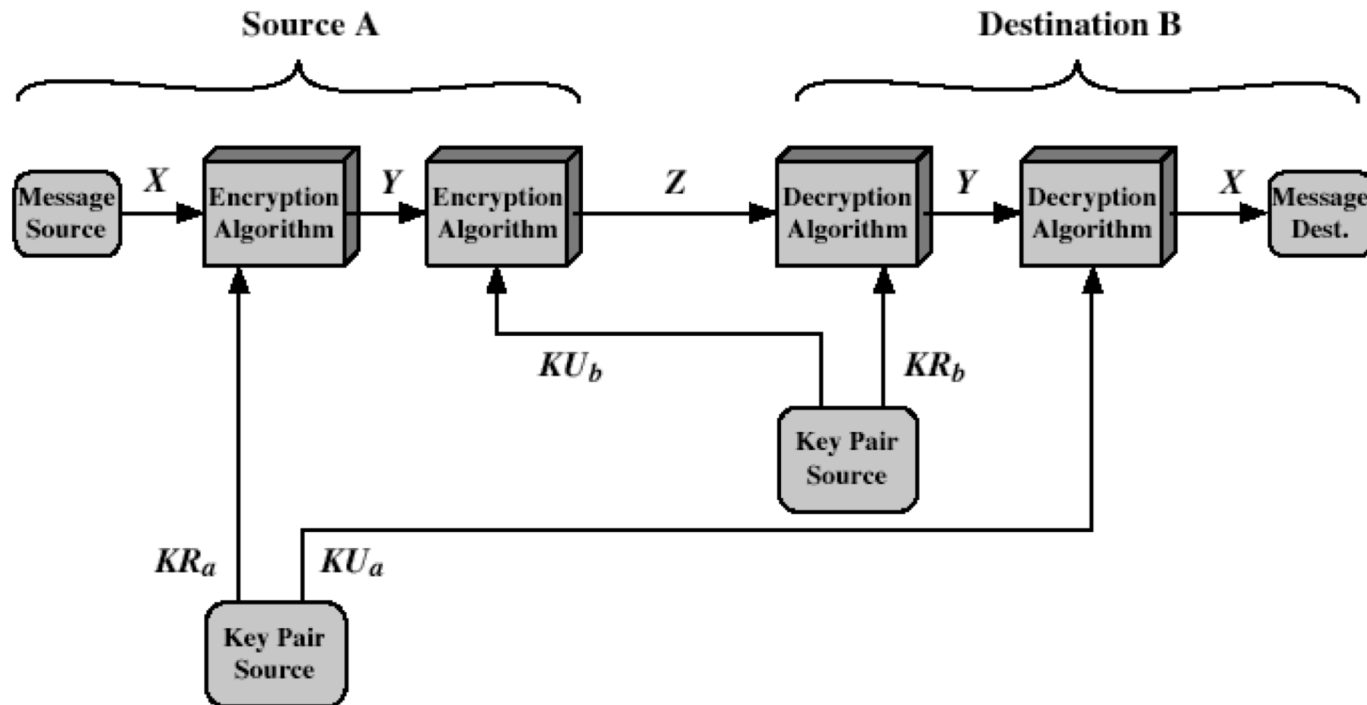
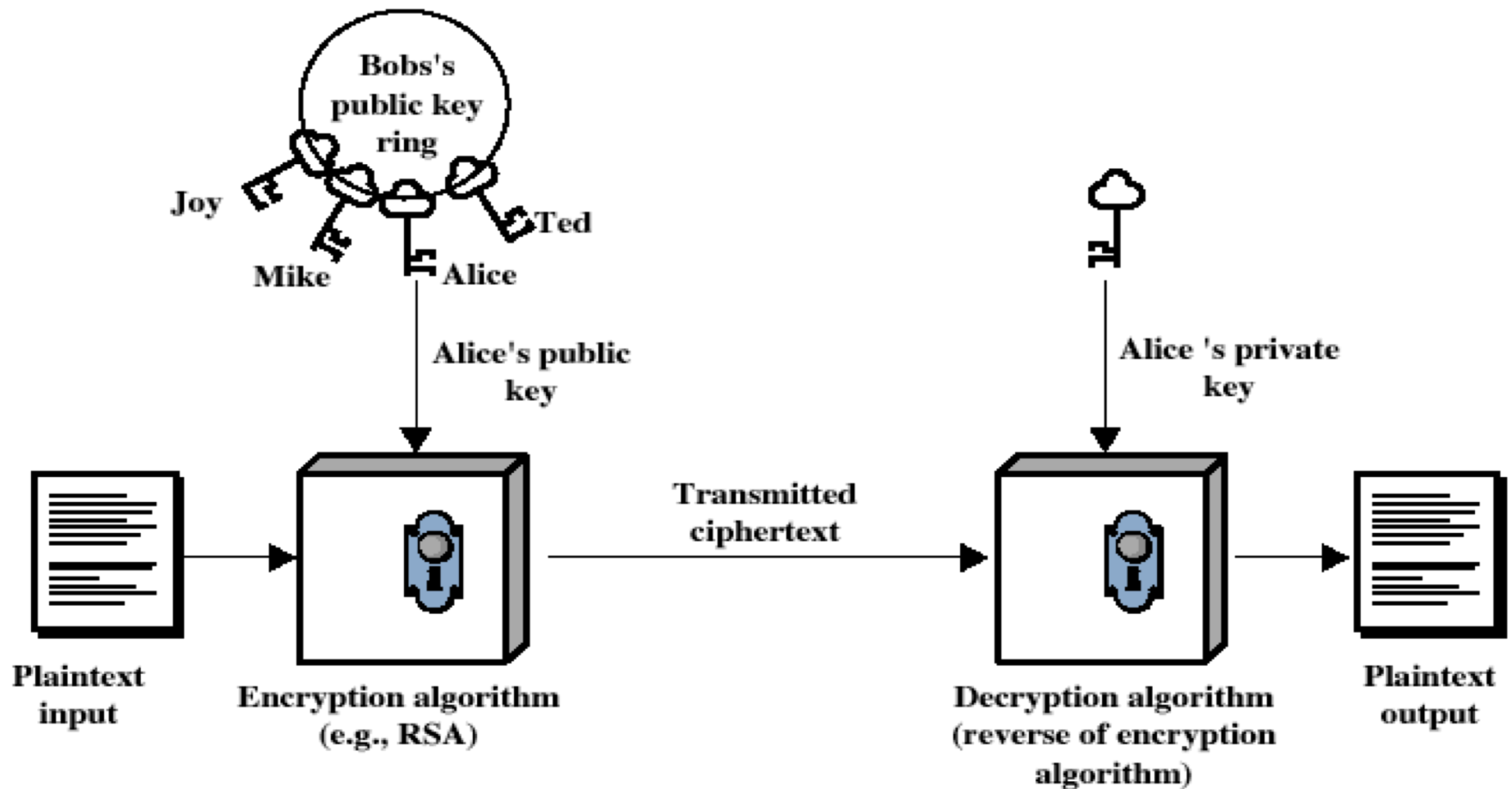


Figure 9.4 Public-Key Cryptosystem: Secrecy and Authentication

Public-Key Applications

- 3 major categories:
 - **encryption/decryption** (provide secrecy)
 - **digital signatures** (provide authentication)
 - **key exchange** (of session keys)
- Some algorithms are suitable for all uses, others are specific to one

PKC: Just Another Illustration



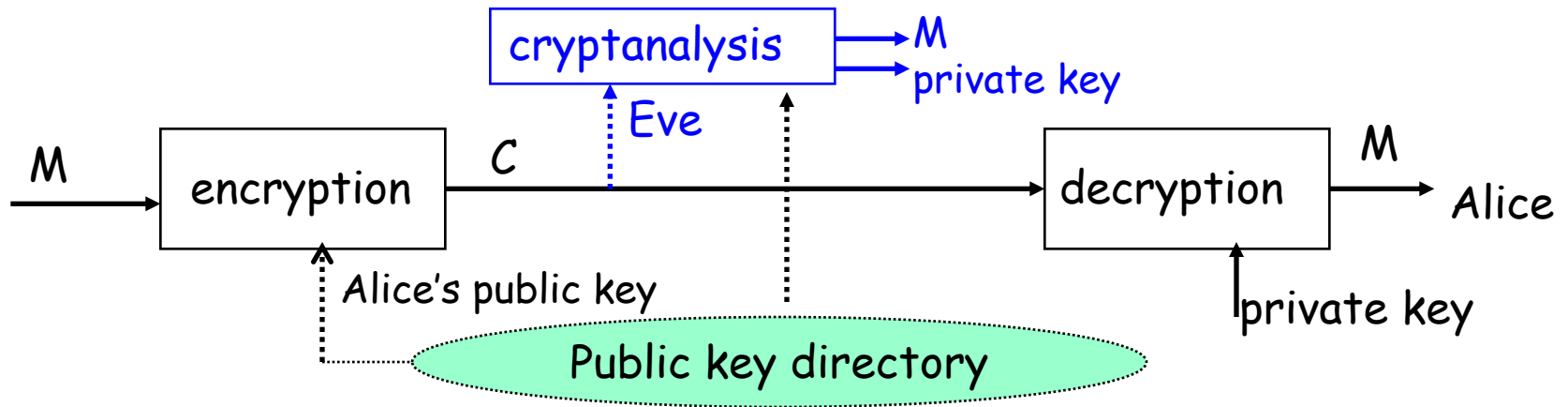
Security of Public Key Schemes

- Security relies on a **large enough** difference in difficulty between **easy** (en/decrypt) and **hard** (cryptanalysis) problems
- Similar to private key schemes, brute force **exhaustive search** attack is always theoretically possible
 - But keys used are too large (>512bits) to break that way
- It requires the use of **very large numbers**
 - **slow** when compared to private key schemes

PKC Computational Characteristics

- Public-Key algorithms rely on two keys with the characteristics:
 - computationally infeasible to find decryption key knowing only algorithm & encryption key
 - computationally easy to en/decrypt messages when the relevant (en/decrypt) key is known
 - either of the two related keys can be used for encryption, with the other used for decryption (in some schemes).

Public-Key Cryptosystem



- $C = E_{PK}(M)$
- $M = D_{SK}(C) = D_{SK}(E_{PK}(M))$
- Public keys are published.
- Each private key is known to the receiver only.
- Difficult for Eve to find out SK from PK.

Why Public-Key Cryptography?

- Initially proposed to address two key issues:
 - **key distribution** – how to have secure communications in general without having to trust a KDC with your key
 - **digital signatures** – how to verify a message comes intact from the claimed sender
- Ripple Effect: Make e-commerce possible.

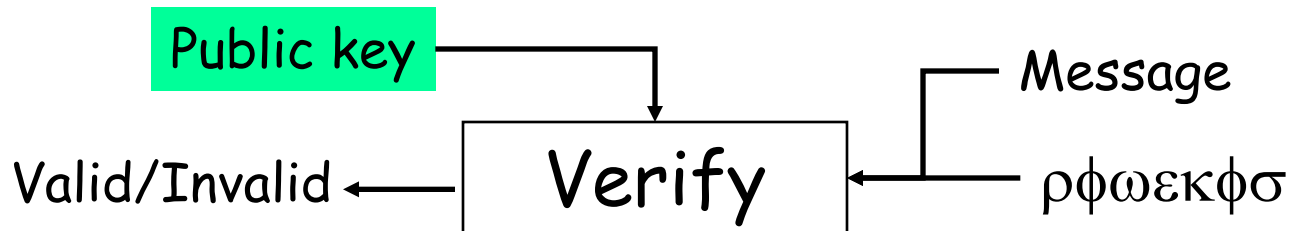
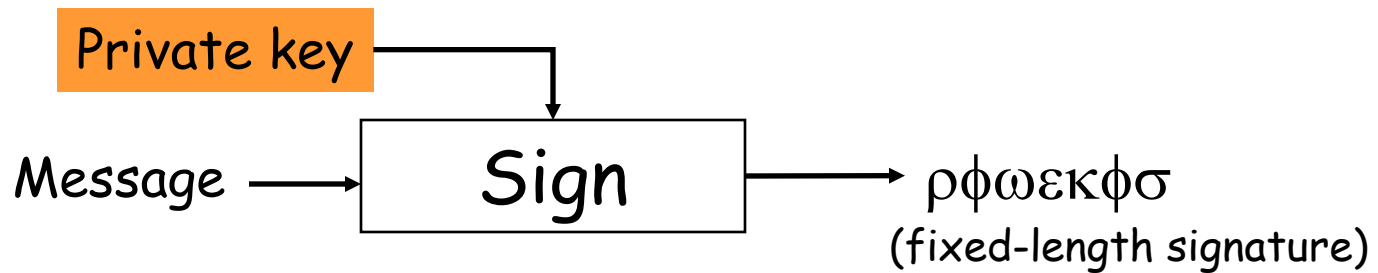
Private-Key vs. Public-Key

Private-Key	Public-Key
Two parties <i>MUST</i> trust each other	Two parties <i>DO NOT</i> need to trust each other
Both share same key (or one key is computable from the other)	Two separate keys: a public and a private key
Typically faster	Typically slower
Examples: DES, IDEA, RC5, AES, ...	Examples: RSA, ElGamal Encryption, ECC...

Digital Signature

- Is there a functional equivalence to a handwritten signature?
 - Easy for legitimate user to sign
 - But hard for anyone else to forge
 - Easy for anyone to verify
 - Dependent on message & signer (key)
- Public key!
 - Sign: “invert” function using private key
 - Verify: compute function using public key

Digital Signatures



- **Only** the signer (who has a private key) can generate a valid signature
- Everyone (since the corresponding public key is published) can verify if a signature with respect to a message is valid