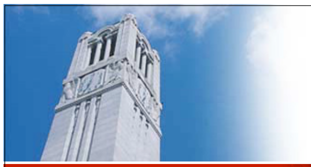


NC STATE UNIVERSITY Computer Science

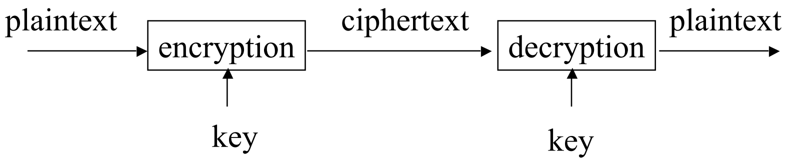
CSC 774 Advanced Network Security

Topic 2. Review of Cryptographic Techniques

Outline

- Encryption/Decryption
- Digital signatures
- Hash functions
  - One-way hash chain
  - Merkle hash tree
- Pseudo random functions
- Key exchange/agreement/distribution

Encryption/Decryption



- Plaintext: a message in its original form
- Ciphertext: a message in the transformed, unrecognized form
- Encryption: the process that transforms a plaintext into a ciphertext
- Decryption: the process that transforms a ciphertext to the corresponding plaintext
- Key: the value used to control encryption/decryption.

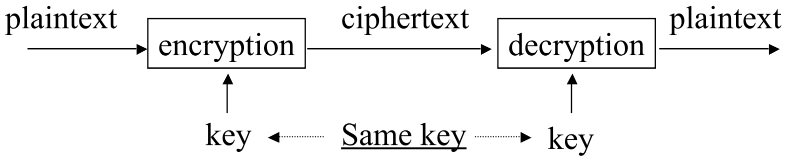
Cryptanalysis

- Ciphertext only:
  - Analyze only with the ciphertext
  - Example: Exhaustive search until “recognizable plaintext”
  - Smarter ways available
- Known plaintext:
  - Secret may be revealed (by spy, time), thus <ciphertext, plaintext> pair is obtained
  - Great for mono-alphabetic ciphers

Cryptanalysis (Cont'd)

- Chosen plaintext:
  - Choose text, get encrypted
  - Useful if limited set of messages
- Chosen ciphertext:
  - Choose ciphertext
  - Get feedback from decryption, etc.

Secret Key Cryptography



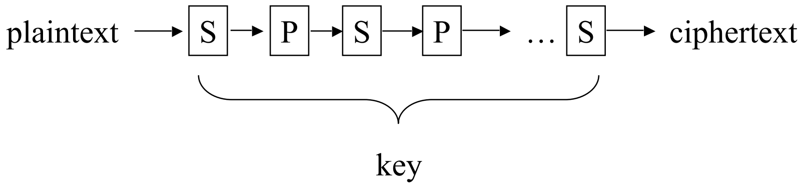
plaintext → encryption → ciphertext → decryption → plaintext

key ← Same key → key

- Same key is used for encryption and decryption
- Also known as
  - Symmetric cryptography
  - Conventional cryptography

Secret Key Cryptography (cont'd)

- Basic technique (block cipher)
  - Product cipher:
  - Multiple applications of interleaved substitutions and permutations

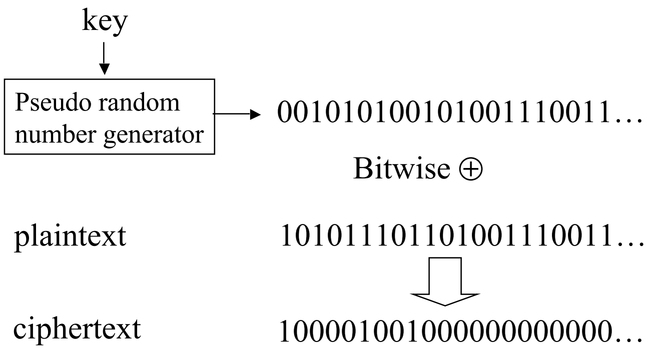


plaintext → S → P → S → P → … S → ciphertext

key

Secret Key Cryptography (cont'd)

- Basic technique (stream cipher)



key

Pseudo random number generator → 0010101001010011100 11…

Bitwise $\oplus$

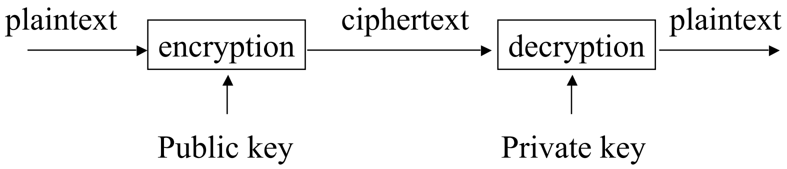
plaintext 1010111011010011100 11…

ciphertext 1000010010000000000 00…

Secret Key Cryptography (cont'd)

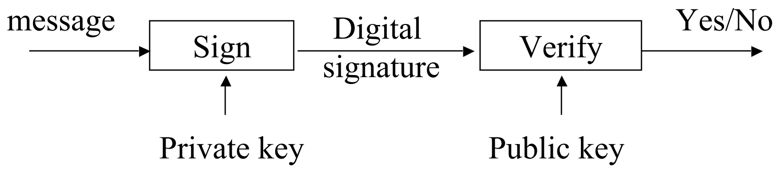
- Cipher-text approximately the same length as plaintext
- Examples
  - Stream Cipher: RC4
  - Block Cipher: DES, IDEA, AES

Public Key Cryptography



- Invented/published in 1975
- A public/private key pair is used
  - Public key can be publicly known
  - Private key is kept secret by the owner of the key
- Much slower than secret key cryptography
- Also known as
  - Asymmetric cryptography

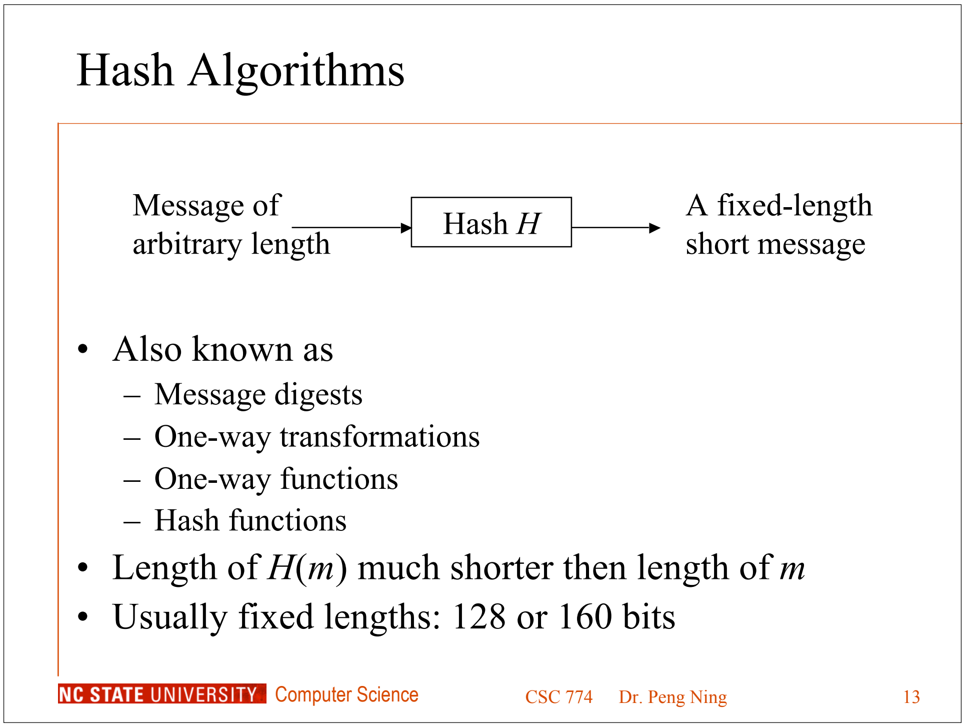
Public Key Cryptography (Cont'd)



- Another mode: digital signature
  - Only the party with the private key can create a digital signature.
  - The digital signature is verifiable by anyone who knows the public key.
  - The signer cannot deny that he/she has done so.

Public Key Cryptography (Cont'd)

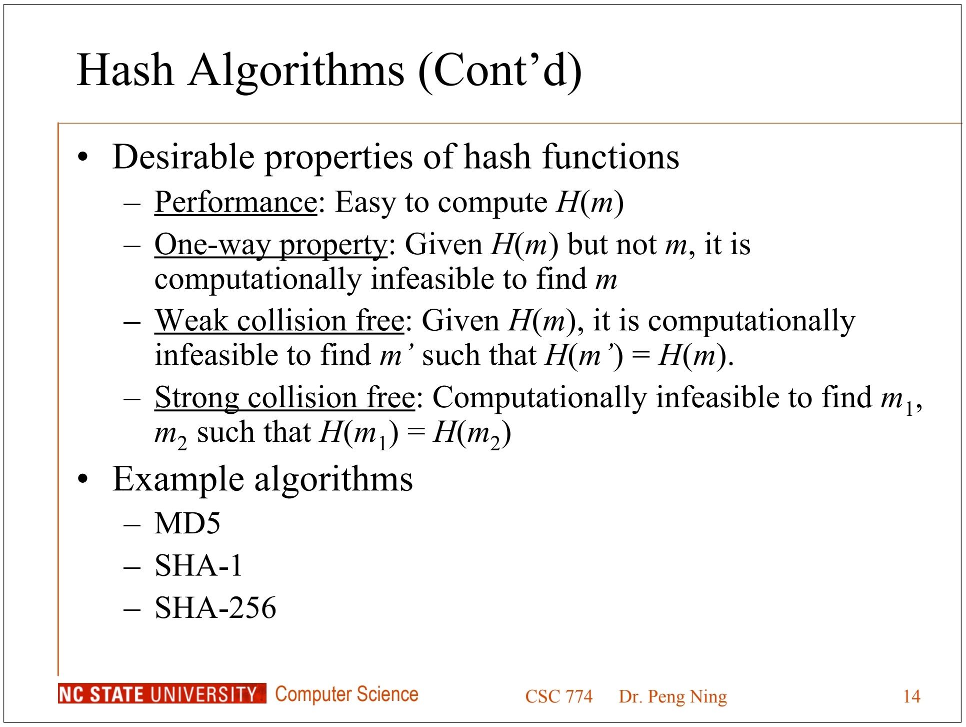
- Example algorithms
  - RSA
  - DSA
  - Diffie-Hellman



Hash Algorithms

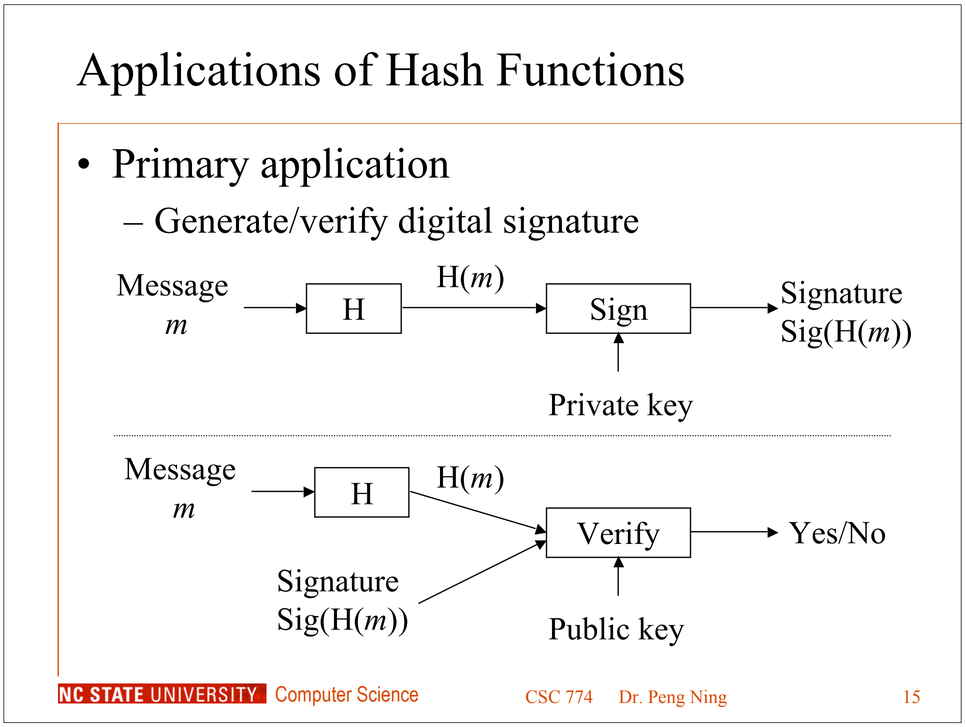
Message of arbitrary length → Hash H → A fixed-length short message

- Also known as
  - Message digests
  - One-way transformations
  - One-way functions
  - Hash functions
- Length of $H(m)$ much shorter then length of m
- Usually fixed lengths: 128 or 160 bits



Hash Algorithms (Cont'd)

- Desirable properties of hash functions
  - Performance: Easy to compute $H(m)$
  - One-way property: Given $H(m)$ but not m, it is computationally infeasible to find m
  - Weak collision free: Given $H(m)$, it is computationally infeasible to find m' such that $H(m') = H(m)$.
  - Strong collision free: Computationally infeasible to find m_1, m_2 such that $H(m_1) = H(m_2)$
- Example algorithms
  - MD5
  - SHA-1
  - SHA-256



Applications of Hash Functions

- Primary application
  - Generate/verify digital signature

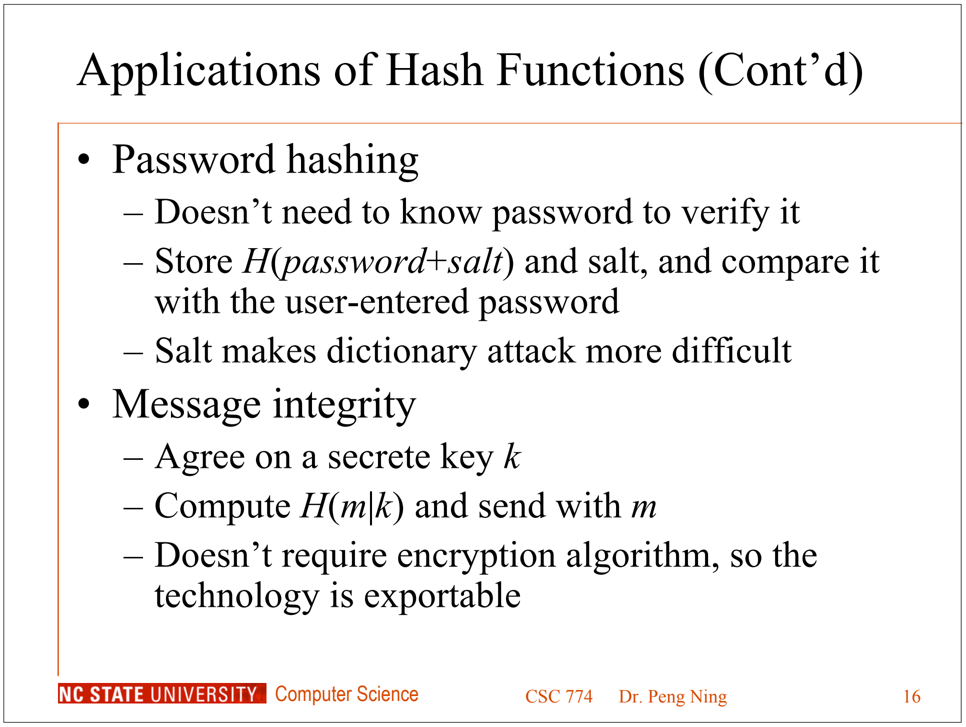
Message *m* → H → H(*m*) → Sign → Signature Sig(H(*m*))

Private key → Sign

Message *m* → H → H(*m*) → Verify → Yes/No

Signature Sig(H(*m*)) → Verify

Public key → Verify



Applications of Hash Functions (Cont’d)

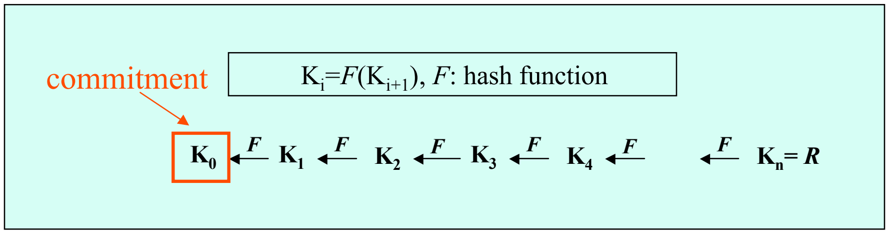
- Password hashing
  - Doesn’t need to know password to verify it
  - Store *H*(*password*+*salt*) and salt, and compare it with the user-entered password
  - Salt makes dictionary attack more difficult
- Message integrity
  - Agree on a secrete key *k*
  - Compute *H*(*m*|*k*) and send with *m*
  - Doesn’t require encryption algorithm, so the technology is exportable

Applications of Hash Functions (Cont'd)

- Authentication
  - Give $H(m)$ as an authentication token
  - Later release m

One-Way Hash Chain

- Used for many network security applications
  - Example: S/Key
- Good for authentication of the hash values



commitment

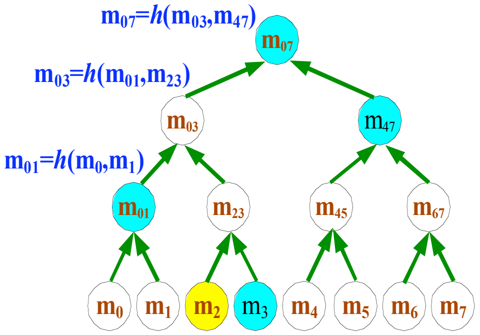
$K_i=F(K_{i+1})$, F: hash function

$$\begin{tikzcd}
\boxed{K_0} & K_1 \arrow[l, "F"'] & K_2 \arrow[l, "F"'] & K_3 \arrow[l, "F"'] & K_4 \arrow[l, "F"'] & {} \arrow[l, "F"'] & {} & K_n = R \arrow[l, "F"']
\end{tikzcd}$$

Merkle Hash Tree

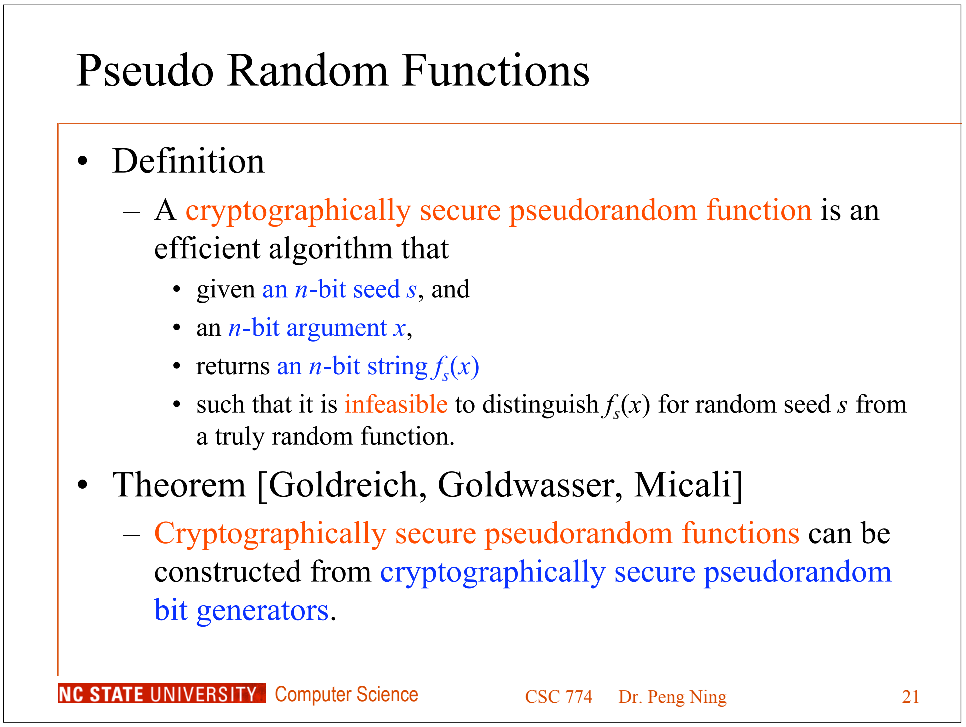
- A binary tree over data values
  - For authentication purpose
- The root is the commitment of the Merkle tree
  - Known to the verifier.
- Example
  - To authenticate m_2, send $(m_2, m_3, m_{01}, m_{47})$
  - Verify

$m_{07}= h(h(m_{01}||h(m_2||m_3)||m_{47})$



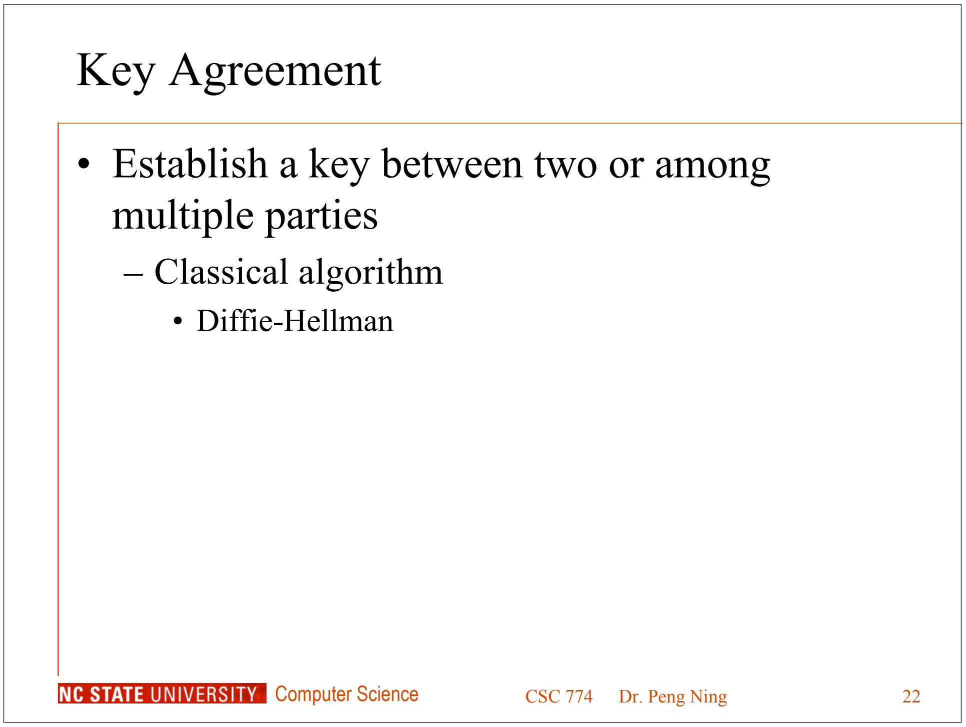
Pseudo Random Generator

- Definition
  - A cryptographically secure pseudorandom bit generator is an efficient algorithm that will expand a random *n*-bit seed to a longer sequence that is computationally indistinguishable from a truly random sequence.
- Theorem [Levin]
  - A one-way function can be used to construct a cryptographically secure pseudo-random bit generator.



Pseudo Random Functions

- Definition
  - A cryptographically secure pseudorandom function is an efficient algorithm that
    - given an n-bit seed s, and
    - an n-bit argument x,
    - returns an n-bit string $f_s(x)$
    - such that it is infeasible to distinguish $f_s(x)$ for random seed s from a truly random function.
- Theorem [Goldreich, Goldwasser, Micali]
  - Cryptographically secure pseudorandom functions can be constructed from cryptographically secure pseudorandom bit generators.



Key Agreement

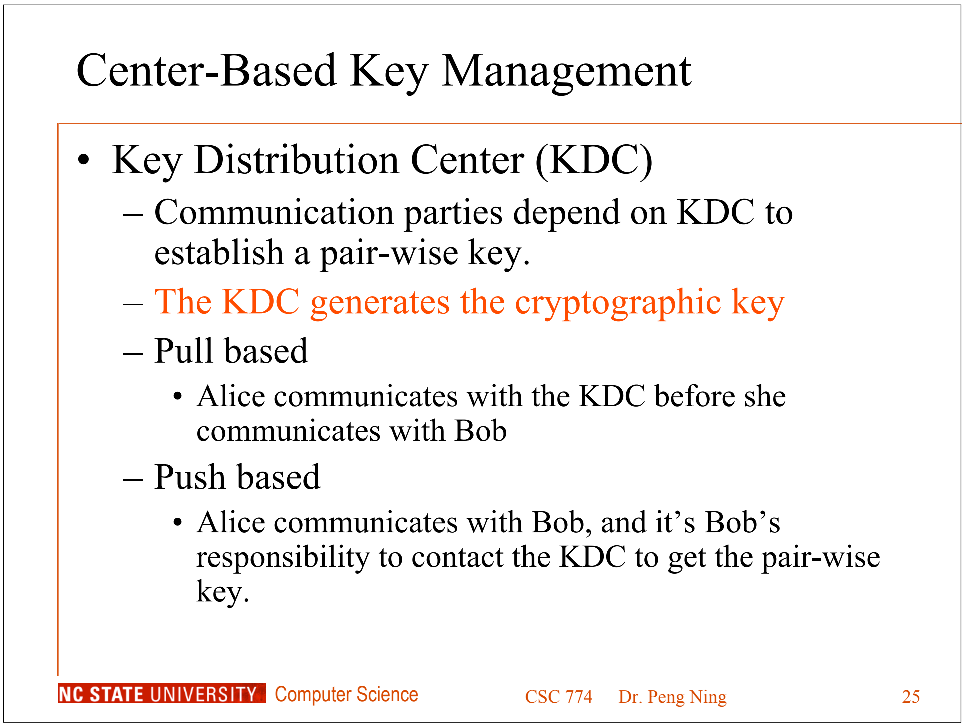
- Establish a key between two or among multiple parties
  - Classical algorithm
    - Diffie-Hellman

Key Exchange

- Key exchange
  - Between two parties
  - A special case of key agreement
  - Use public key cryptography
    - Examples: RSA, DH
  - Use symmetric key cryptography
    - Usually requires a pre-shared key

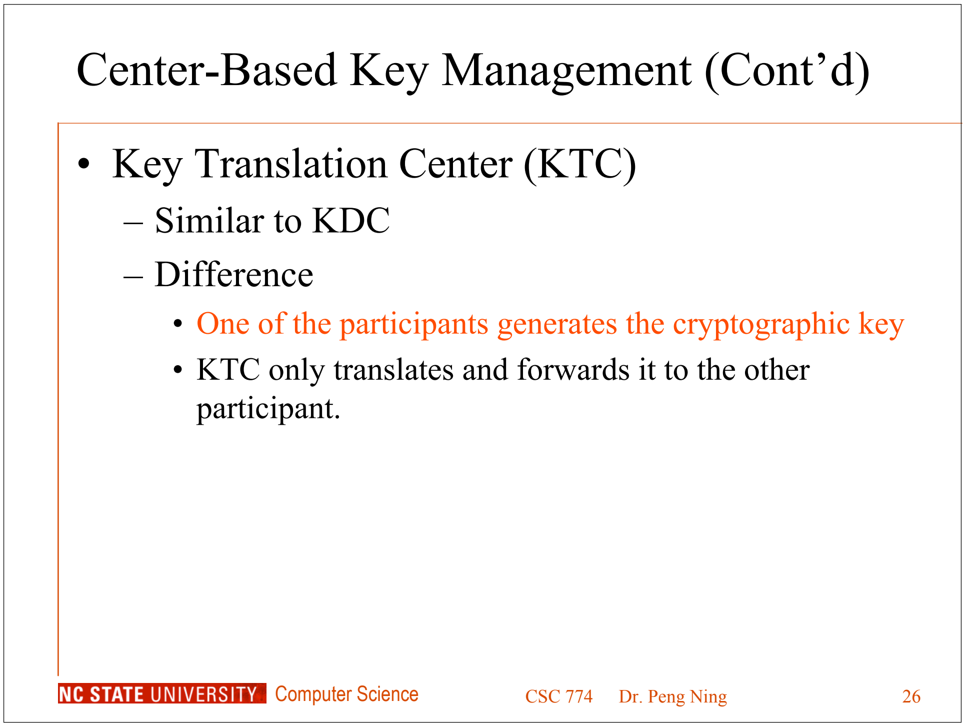
Key Distribution

- Involves a (trusted) third party to help establish keys.
- Based on
  - Symmetric key cryptography, or
  - Public key cryptography



Center-Based Key Management

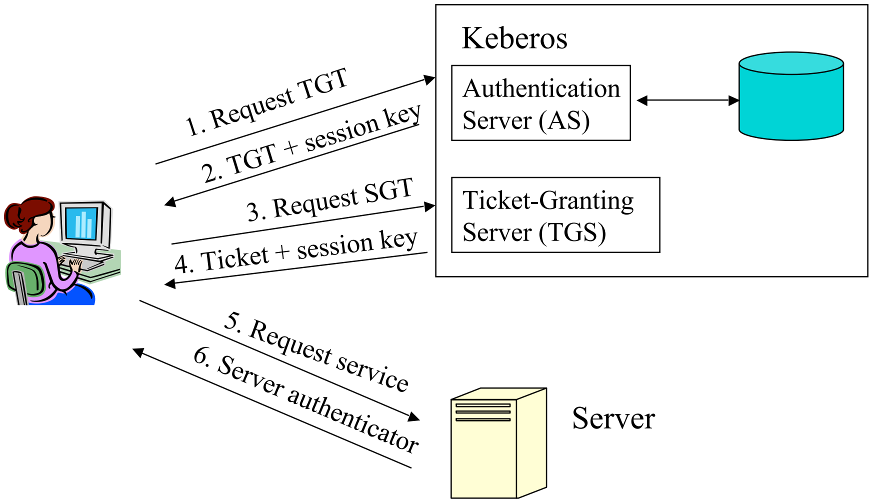
- Key Distribution Center (KDC)
  - Communication parties depend on KDC to establish a pair-wise key.
  - The KDC generates the cryptographic key
  - Pull based
    - Alice communicates with the KDC before she communicates with Bob
  - Push based
    - Alice communicates with Bob, and it's Bob's responsibility to contact the KDC to get the pair-wise key.



Center-Based Key Management (Cont'd)

- Key Translation Center (KTC)
  - Similar to KDC
  - Difference
    - One of the participants generates the cryptographic key
    - KTC only translates and forwards it to the other participant.

An Example of KDC: Kerberos



Keberos

Authentication Server (AS)

Ticket-Granting Server (TGS)

1. Request TGT

2. TGT + session key

3. Request SGT

4. Ticket + session key

5. Request service

6. Server authenticator

Server

When Public Key Cryptography is Used

- Need to authenticate public keys
- Public key certificate
  - Bind an identity and a public key together
  - Verify the authenticity of a party’s public key

Attacks

- Replay attacks
- Man-in-the-middle attacks
- Resource clogging attacks
- Denial of service attacks
- Meet-in-the-middle attacks
- Dictionary attacks
- Others specific to protocols