HA NOI UNIVERSITY OF SCIENCE AND TECHNOLOGY
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

Cryptography IV

Key Management

Overview

- Key exchange
  - Session vs. interchange keys
  - Classical, public key methods
  - Key generation
- Cryptographic key infrastructure
  - Certificates
- Key storage
  - Key escrow
  - Key revocation

Cryptographic protocols with applications

- Zero- Knowledge Protocols
- Subliminal channel
- Special signature schemes
- Electronic account-based payment systems
- Electronic anonymous cash systems
- Micropayment systems
- Secure multi-party computation
- Watermarking and DRM
- Diffie-Hellman key exchange
- Keberos
-

Notation

- $X \rightarrow Y : \{ Z \,||\, W \}\, k_{X,Y}$
  - X sends Y the message produced by concatenating Z and W enciphered by key $k_{X,Y}$, which is shared by users X and Y
- $A \rightarrow T : \{ Z \}\, k_A \,||\, \{ W \}\, k_{A,T}$
  - A sends T a message consisting of the concatenation of Z enciphered using k_A, A's key, and W enciphered using $k_{A,T}$, the key shared by A and T
- r_1, r_2 nonces (nonrepeating random numbers)

Session key - Interchange key

- Alice wants to send a message *m* to Bob
  - Assume public key encryption
  - Alice know Bob's public key Z_B
- Proposed protocol
  - Alice generates a random cryptographic key k_s and uses it to encipher *m*
    - To be used for this message *only*
    - Called a *session key*
  - She enciphers k_s with Bob's public key Z_B
    - Z_B enciphers all session keys Alice uses to communicate with Bob
    - Called an *interchange key*
  - Alice sends Bob: $\{ m \} k_s \,||\, \{ k_s \} Z_B$

Why session key?

- Limits amount of traffic enciphered with single key
  - Standard practice, to decrease the amount of traffic an attacker can obtain
- Prevents some attacks
  - Example: Alice will send Bob message that is either “BUY” or “SELL”. Eve computes possible ciphertexts { “BUY” } Z_B and { “SELL” } Z_B. Eve intercepts enciphered message, compares, and gets plaintext at once

Key Exchange Algorithms

- Goal: Alice, Bob to get shared key (wo/ interchange key)
  - Key cannot be sent in clear
    - Attacker can listen in
    - Key can be sent enciphered, or derived from exchanged data plus data not known to an eavesdropper
  - Alice, Bob may use a trusted third party
  - All cryptosystems, protocols publicly known
    - Only secret data is the keys, ancillary information known only to Alice and Bob needed to derive keys
    - Anything transmitted is assumed known to attacker

Classical Key Exchange

- Bootstrap problem: how do Alice, Bob begin?
  - Alice can't send it to Bob in the clear!
- Assume trusted third party, Cathy
  - Alice and Cathy share secret key k_{AC}
  - Bob and Cathy share secret key k_{BC}
  - Use this to exchange shared key k_s

Simple Protocol

$$\text{Alice} \xrightarrow{\{\text{ request for session key to Bob }\}\, k_{AC}} \text{Cathy}$$

$$\text{Alice} \xleftarrow{\{\, k_s \,\}\, k_{AC} \,\|\, \{\, k_s \,\}\, k_{BC}} \text{Cathy}$$

$$\text{Alice} \xrightarrow{\{\, k_s \,\}\, k_{BC}} \text{Bob}$$

Problems

- How does Bob know he is talking to Alice?
  - Replay attack: Eve records message from Alice to Bob, later replays it; Bob may think he's talking to Alice, but he isn't
  - Session key reuse: Eve replays message from Alice to Bob, so Bob re-uses session key
    - Eve may fortunately get a session key dropped by Alice
- Protocols must provide authentication and defense against replay

Needham-Schroeder

$$\text{Alice} \xrightarrow{\text{Alice} \,\|\, \text{Bob} \,\|\, r_1} \text{Cathy}$$

$$\text{Alice} \xleftarrow{\{ \text{Alice} \,\|\, \text{Bob} \,\|\, r_1 \,\|\, k_s \,\|\, \{ \text{Alice} \,\|\, k_s \} k_{BC} \} k_{AC}} \text{Cathy}$$

$$\text{Alice} \xrightarrow{\{ \text{Alice} \,\|\, k_s \} k_{BC}} \text{Bob}$$

$$\text{Alice} \xleftarrow{\{ r_2 \} k_s} \text{Bob}$$

$$\text{Alice} \xrightarrow{\{ r_2 - 1 \} k_s} \text{Bob}$$

Argument: Alice talking to Bob

- Second message
  - Enciphered using key only she, Cathy knows
    - So Cathy enciphered it
  - Response to first message
    - As r_1 in it matches r_1 in first message
- Third message
  - Alice knows only Bob can read it
    - As only Bob can derive session key from message
  - Any messages enciphered with that key are from Bob

Argument: Bob talking to Alice

- Third message
  - Observe that: Enciphered using key only he and Cathy know
    - So Cathy enciphered it
  - Inside are the name Alice and the session key
    - Bob concludes that Cathy provided session key, saying Alice is the other party
- Fourth and Firth messages:
  - Use session key to determine if it is replay from Eve
    - If not, Alice will respond correctly in fifth message
    - If so, Eve can't decipher r_2 and so can't respond, or responds incorrectly

Denning-Sacco Problem

- Assumption: all keys are secret
- Question: suppose Eve can obtain session key. How does that affect protocol?
  - In what follows, by chance Eve knows k_s

$$\text{Eve} \xrightarrow{\{\text{ Alice} \,||\, k_s \}\, k_{BC}} \text{Bob}$$

$$\text{Eve} \xleftarrow{\{\, r_2 \,\}\, k_s} \text{Bob}$$

$$\text{Eve} \xrightarrow{\{\, r_2 - 1 \,\}\, k_s} \text{Bob}$$

Solution

- In protocol above, Eve impersonates Alice
- Problem: replay in third step
  - First in previous slide
- Solution: use time stamp *T* to detect replay

Needham-Schroeder with Denning-Sacco Modification

$$\text{Alice} \xrightarrow{\text{Alice} \,||\, \text{Bob} \,||\, r_1} \text{Cathy}$$

$$\text{Alice} \xleftarrow{\{ \text{Alice} \,||\, \text{Bob} \,||\, r_1 \,||\, k_s \,||\, \{ \text{Alice} \,||\, T \,||\, k_s \}\, k_B \}\, k_A} \text{Cathy}$$

$$\text{Alice} \xrightarrow{\{ \text{Alice} \,||\, T \,||\, k_s \}\, k_B} \text{Bob}$$

$$\text{Alice} \xleftarrow{\{ r_2 \}\, k_s} \text{Bob}$$

$$\text{Alice} \xrightarrow{\{ r_2 - 1 \}\, k_s} \text{Bob}$$

Still weakness, anyway

- If clocks not synchronized, may either reject valid messages or accept replays
  - Parties with either slow or fast clocks vulnerable to replay
  - Resetting clock does *not* eliminate vulnerability
- Use Otway-Rees protocol (Bishop’s text)

Kerberos

- Authentication system
  - Based on Needham-Schroeder with Denning-Sacco modification
  - Central server plays role of trusted third party (“Cathy”)
- Ticket
  - Issuer vouches for identity of requester of service
- Authenticator
  - Identifies sender

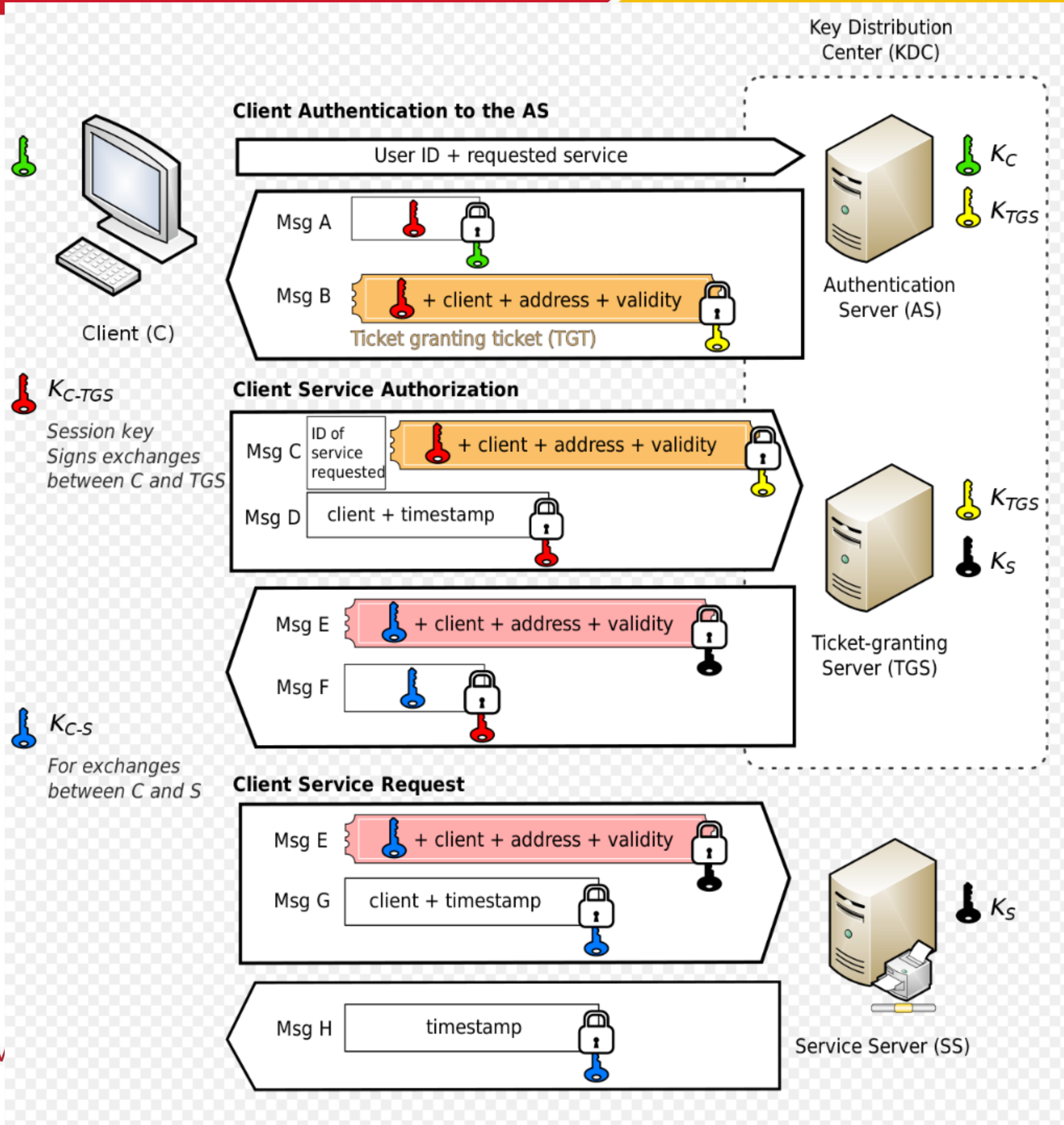
Idea

- Key Distribution Center (KDC) combines two severs:
  - Authentication Server, AS (Also, Kerberos server)
  - Ticket Granting Server, TGS
- User *u* authenticates to AS
    - Obtains ticket $T_{u,TGS}$ for ticket granting service (TGS)
- User *u* wants to use service *s*:
  - User sends authenticator A_u, ticket $T_{u,TGS}$ to TGS asking for ticket for service
  - TGS sends ticket $T_{u,s}$ to user
  - User sends A_u, $T_{u,s}$ to server as request to use *s*
- Details in Bishop's text (Wiki also OK)

Protocol

[From Wiki]

- Client Authentication to the AS
- Client Service Authorization
- Client Service Request



Public Key Key Exchange

- Here interchange keys known
  - e_A, e_B Alice and Bob's public keys known to all
  - d_A, d_B Alice and Bob's private keys known only to owner
- Simple protocol
  - k_s is desired session key

$$\text{Alice} \xrightarrow{\{ k_s \}\, e_B} \text{Bob}$$

Problem and Solution

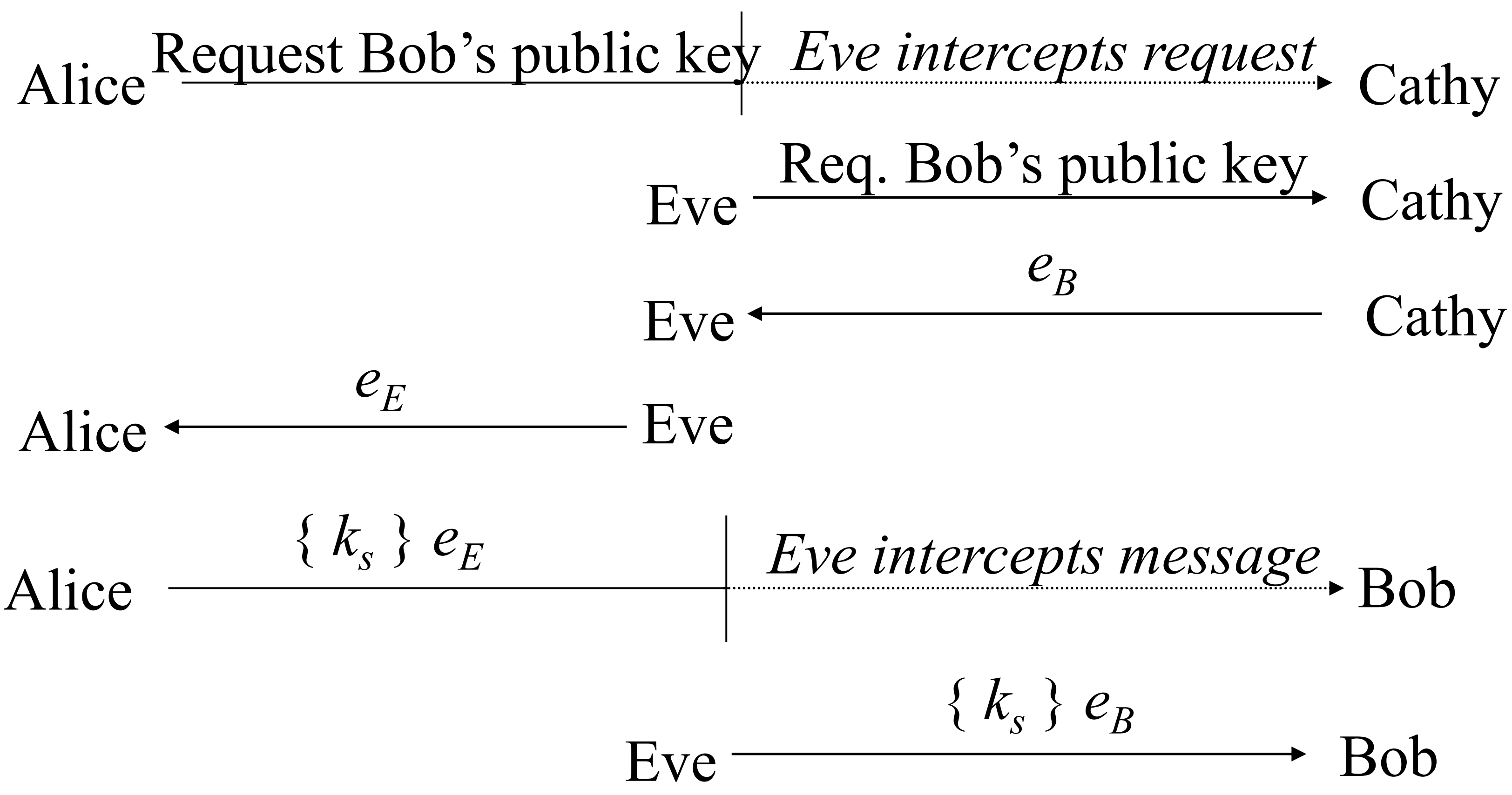
- Vulnerable to forgery or replay
  - Because e_B known to anyone, Bob has no assurance that Alice sent message
- Simple fix uses Alice's private key
  - k_s is desired session key
  - **Quiz:** Can Eve impersonate Alice and succeed to share a k_s with Bob?

$$\text{Alice} \xrightarrow{\{\,\{\,k_s\,\}\,d_A\,\}\,e_B} \text{Bob}$$

Notes

- Can include message enciphered with k_s
- Assumes Bob has Alice's public key, and *vice versa*
  - If not, each must get it from public server
  - If keys not bound to identity of owner, attacker Eve can launch a *man-in-the-middle* attack (next slide; Cathy is public server providing public keys)
    - Solution to this (binding identity to keys) discussed later as public key infrastructure (PKI)

Man-in-the-Middle Attack



Key Generation

- Goal: generate keys that are difficult to guess
- Problem statement: given a set of *K* potential keys, choose one randomly
  - Equivalent to selecting a random number between 0 and *K*–1 inclusive
- Why is this hard: generating random numbers
  - Actually, numbers are usually *pseudo-random*, that is, generated by an algorithm

What is "Random"?

- *Sequence of cryptographically random numbers*: a sequence of numbers n_1, n_2, … such that for any integer $k > 0$, an observer cannot predict n_k even if all of n_1, …, n_{k-1} are known
  - Best: physical source of randomness
    - Random pulses
    - Electromagnetic phenomena
    - Characteristics of computing environment such as disk latency
    - Ambient background noise

What is “Pseudorandom”?

- *Sequence of cryptographically pseudorandom numbers*: sequence of numbers intended to simulate a sequence of cryptographically random numbers but generated by an algorithm
  - Very difficult to do this well
    - Linear congruential generators [$n_k = (an_{k-1} + b) \bmod n$] broken
    - Polynomial congruential generators [$n_k = (a_j n_{k-1}^j + \ldots + a_1 n_{k-1}\ a_0) \bmod n$] broken too
    - Here, “broken” means next number in sequence can be determined

Best Pseudorandom Numbers

- *Strong mixing function*: function of 2 or more inputs with each bit of output depending on some nonlinear function of all input bits
  - Examples: DES, MD5, SHA-1
  - Use on UNIX-based systems:

```
(date; ps gaux) | md5
```

  where “ps gaux” lists all information about all processes on system

Cryptographic Key Infrastructure

- Goal: bind identity to key
- Classical: not possible as all keys are shared
  - Use protocols to agree on a shared key (see earlier)
- Public key: bind identity to public key
  - Crucial as people will use key to communicate with principal whose identity is bound to key
  - Erroneous binding means no secrecy between principals
  - Assume principal identified by an acceptable name

Certificates

- Create token (message) containing
  - Identity of principal (here, Alice)
  - Corresponding public key
  - Timestamp (when issued)
  - Other information (perhaps identity of signer)

  signed by trusted authority (here, Cathy)

$$C_A = \{ e_A \,||\, \text{Alice} \,||\, T \} \, d_C$$

Use

- Bob gets Alice's certificate
  - If he knows Cathy's public key, he can decipher the certificate
    - When was certificate issued?
    - Is the principal Alice?
  - Now Bob has Alice's public key
- Problem: Bob needs Cathy's public key to validate certificate
  - Problem pushed "up" a level
  - Two approaches: Merkle's tree, signature chains

Certificate Signature Chains

- Create certificate
  - Generate hash of certificate
  - Encipher hash with issuer's private key
- Validate
  - Obtain issuer's public key
  - Decipher enciphered hash
  - Recompute hash from certificate and compare
- Problem: getting issuer's public key

X.509 Chains

- Some certificate components in X.509v3:
  - Version
  - Serial number
  - Signature algorithm identifier: hash algorithm
  - Issuer’s name; uniquely identifies issuer
  - Interval of validity
  - Subject’s name; uniquely identifies subject
  - Subject’s public key
  - Signature: enciphered hash

X.509 Certificate Validation

- Obtain issuer's public key
  - The one for the particular signature algorithm
- Decipher signature
  - Gives hash of certificate
- Recompute hash from certificate and compare
  - If they differ, there's a problem
- Check interval of validity
  - This confirms that certificate is current

Issuers

- *Certification Authority (CA)*: entity that issues certificates
  - Multiple issuers pose validation problem
  - Alice's CA is Cathy; Bob's CA is Don; how can Alice validate Bob's certificate?
  - Have Cathy and Don cross-certify
    - Each issues certificate for the other

Validation and Cross-Certifying

- Certificates:
  - Cathy<<Alice>>
  - Dan<<Bob>
  - Cathy<<Dan>>
  - Dan<<Cathy>>
- Alice validates Bob’s certificate
  - Alice obtains Cathy<<Dan>>
  - Alice uses (known) public key of Cathy to validate Cathy<<Dan>>
  - Alice uses Cathy<<Dan>> to validate Dan<<Bob>>

ĐẠI HỌC BÁCH KHOA

25 YEARS ANNIVERSARY SOICT

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

Thank you for your attentions!