Key Management and Distribution

CSS322: Security and Cryptography

Sirindhorn International Institute of Technology
Thammasat University

Prepared by Steven Gordon on 29 December 2011
CSS322Y11S2L10, Steve/Courses/2011/S2/CSS322/Lectures/key.tex, r2070

Contents

Key Distribution and Management

- Symmetric key cryptography: fast implementations, good for encrypting large amounts of data; requires shared secret key
- Asymmetric (public) key cryptography: inefficient for large data, good for authentication; no need to share a secret
- How to share symmetric keys?
- How to distribute public keys?

Contents

Symmetric Key Distribution using Symmetric Encryption

- Objective: two entities share same secret key
- Principle: change keys frequently
- How to exchange a secret key?
  1. A physically delivers key to B
  2. Third party, C, can physically deliver key to A and B
  3. If A and B already have a key, can securely transmit new key to each other, encrypted with old key
  4. If A and B have secure connection with third party C, C can securely send keys to A and B
- Option 1 and 2: manual delivery; feasible if number of entites is small (link encryption)
- Option 3: requires initial distribution of key; discovery of initial key releases all subsequent keys
- Option 4: requires initial distribution of key with C; practical for large-scale systems (end-to-end encryption)

Link Encryption vs End-to-End Encryption

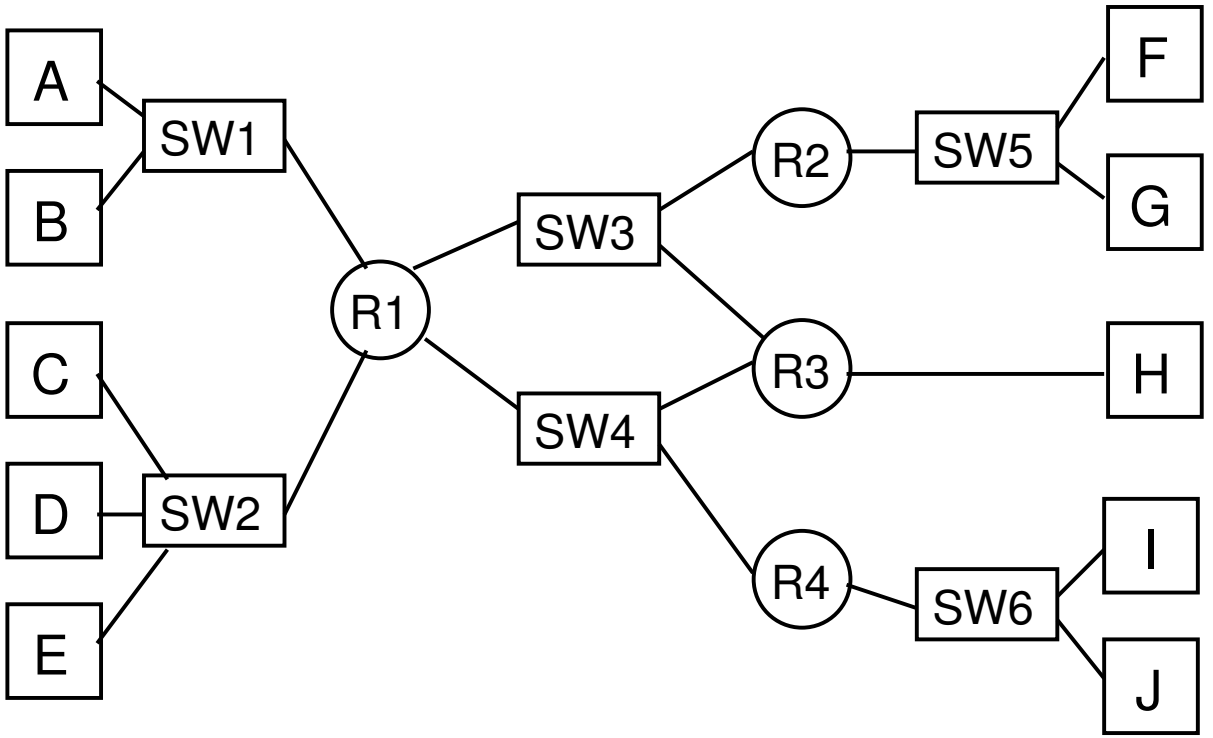
Link Encryption

- Encrypt data over individual links in network
- Each link end-point shares a secret key
- Decrypt/Encrypt at each device in path
- Requires all links/devices to support encryption

End-to-End Encryption

- Encrypt data at network end-points (e.g. hosts or applications)
- Each pair of hosts/applications share a secret key
- Does not rely on intermediate network devices

How Many Keys Need To Be Exchanged?

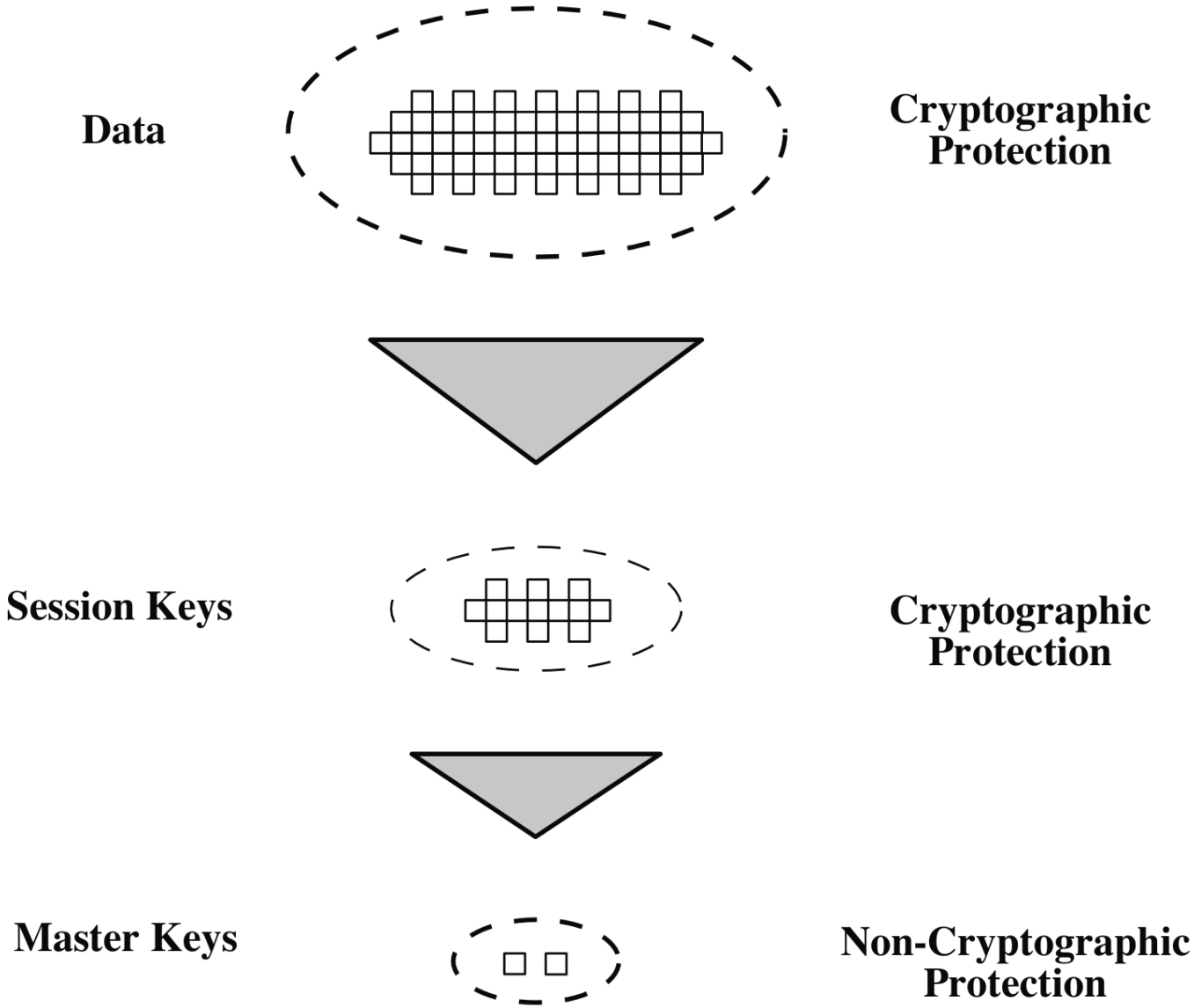


- Link-level encryption?
- End-to-end encryption between hosts?
- End-to-end encryption between applications?

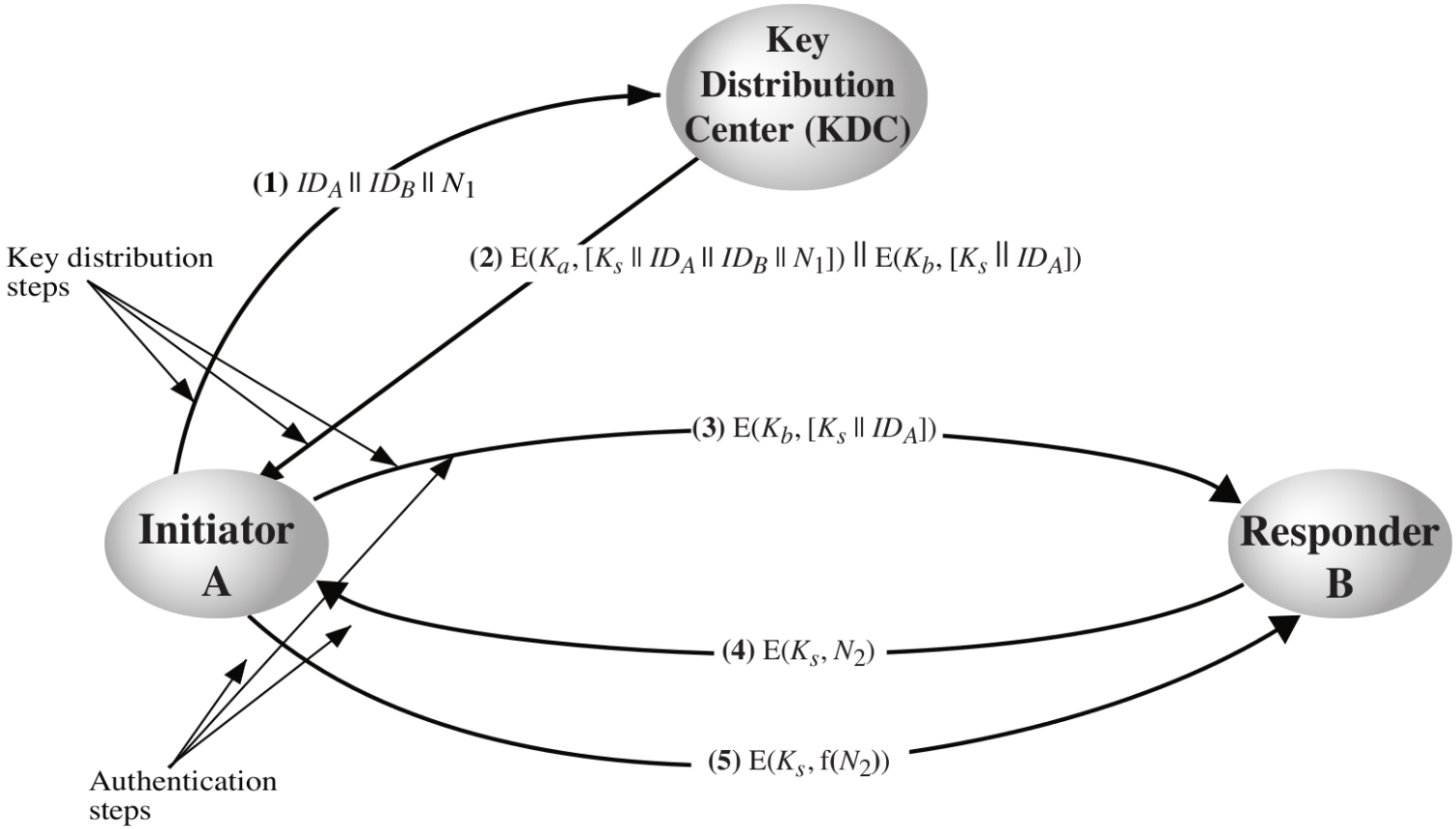
Using a Key Distribution Centre

- Key Distribution Centre (KDC) is trusted third party
- Hierarchy of keys used:
  - Data sent between end-systems encrypted with temporary session key
  - Session keys obtained from KDC over network; encrypted with master key
  - Master keys can be distributed using manual delivery

Use of a Key Hierarchy



Key Distribution Scenario



KDC Scenario Notation

- End-systems: A and B, identified by ID_A and ID_B
- Master keys: K_a, K_b
- Session key (between A and B): K_s
- Nonce values: N_1, N_2
  - E.g. timestamp, counter, random value
  - Must be different for each request
  - Must be difficult for attacker to guess

Practical Considerations

Hierarchical Key Control

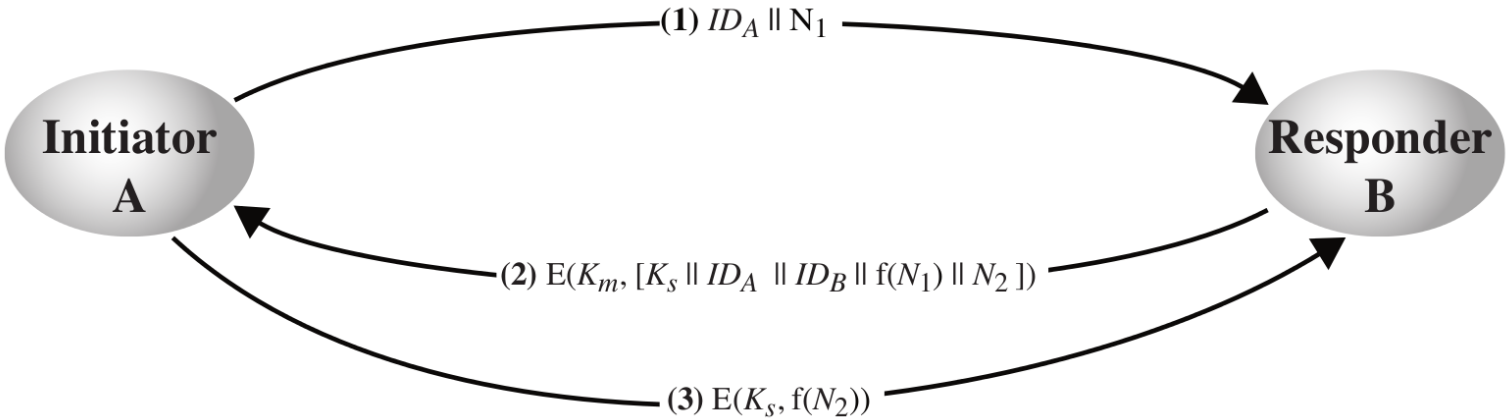
- Use multiple KDCs in a hierarchy
- E.g. KDC for each LAN (or building); central KDC to exchange keys between hosts in different LANs
- Reduces effort in key distribution; limits damage if local KDC is compromised

Session Key Lifetime

- Shorter lifetime is more secure; but increases overhead of exchanges
- Connection-oriented protocols (e.g. TCP): new session key for each connection
- Connection-less protocols (e.g. UDP/IP): change after fixed period or certain number of packets sent

Decentralised Key Distribution

- Alternative that doesn't rely on KDC
- Each end-system must manually exchange $n - 1$ master keys (K_m) with others



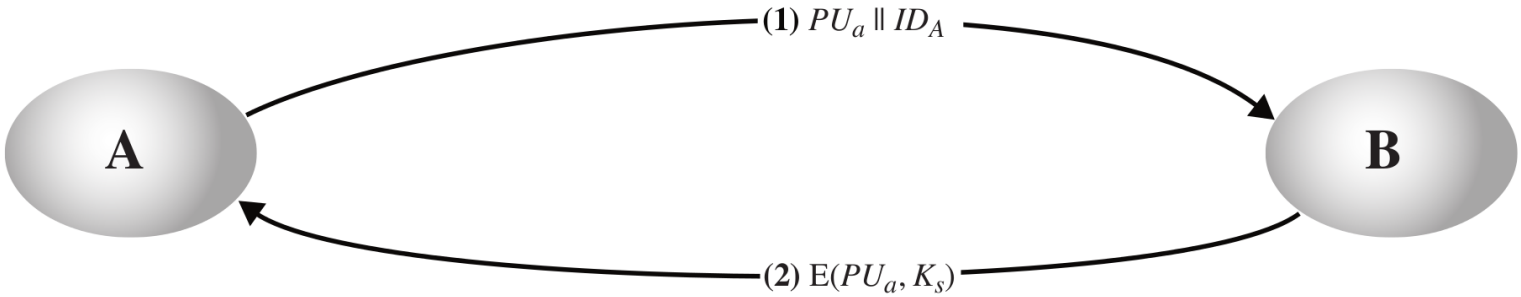
Contents

Symmetric Key Distribution using Asymmetric Encryption

- Asymmetric encryption generally too slow for encrypting large amount of data
- Common application of asymmetric encryption is exchanging secret keys
- Three examples:
  1. Simple Secret Key Distribution
  2. Secret Key Distribution with Confidentiality and Authentication
  3. Hybrid Scheme: Public-Key Distribution of KDC Master Keys

Simple Secret Key Distribution

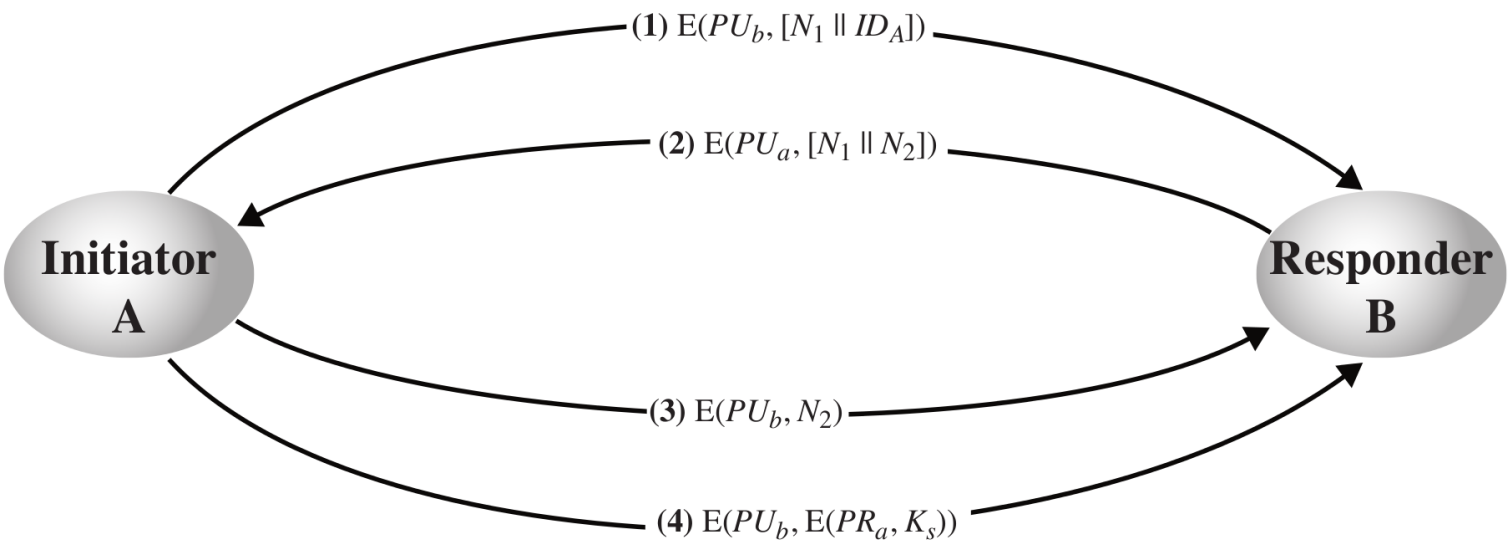
- Simple: no keys prior to or after communication
- Provides confidentiality for session key
- Subject to man-in-the-middle attack
- Only useful if attacker cannot modify/insert messages



Man-in-the-Middle Attack

Secret Key Distribution with Confidentiality and Authentication

- Provides both confidentiality and authentication in exchange of secret key



Hybrid Scheme: Public-Key Distribution of KDC Master Keys

- Use public-key distribution of secret keys when exchaning master keys between end-systems and KDC
- Efficient method of delivering master keys (rather than manual delivery)
- Useful for large networks, widely distributed set of users with single KDC

Contents

Distribution of Public Keys

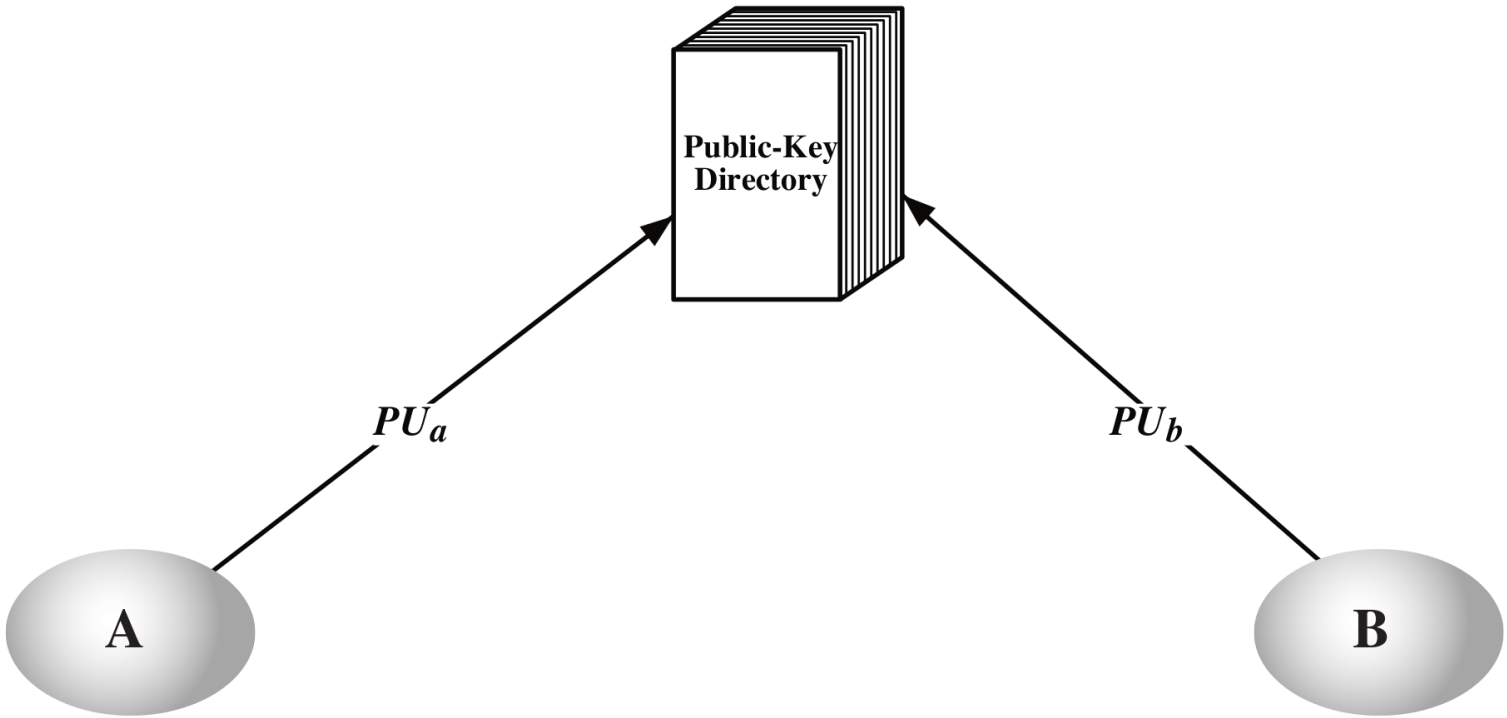
- By design, public keys are made public
- Issue: how to ensure public key of A actually belongs to A (and not someone pretending to be A)
- Four approaches for distributing public keys
  1. Public announcement
  2. Publicly available directory
  3. Public-key authority
  4. Public-key certificates

Public Announcements

- Make public key available in open forum: newspaper, email signature, website, conference, ...
- Problem: anyone can announce a key pretending to be another user

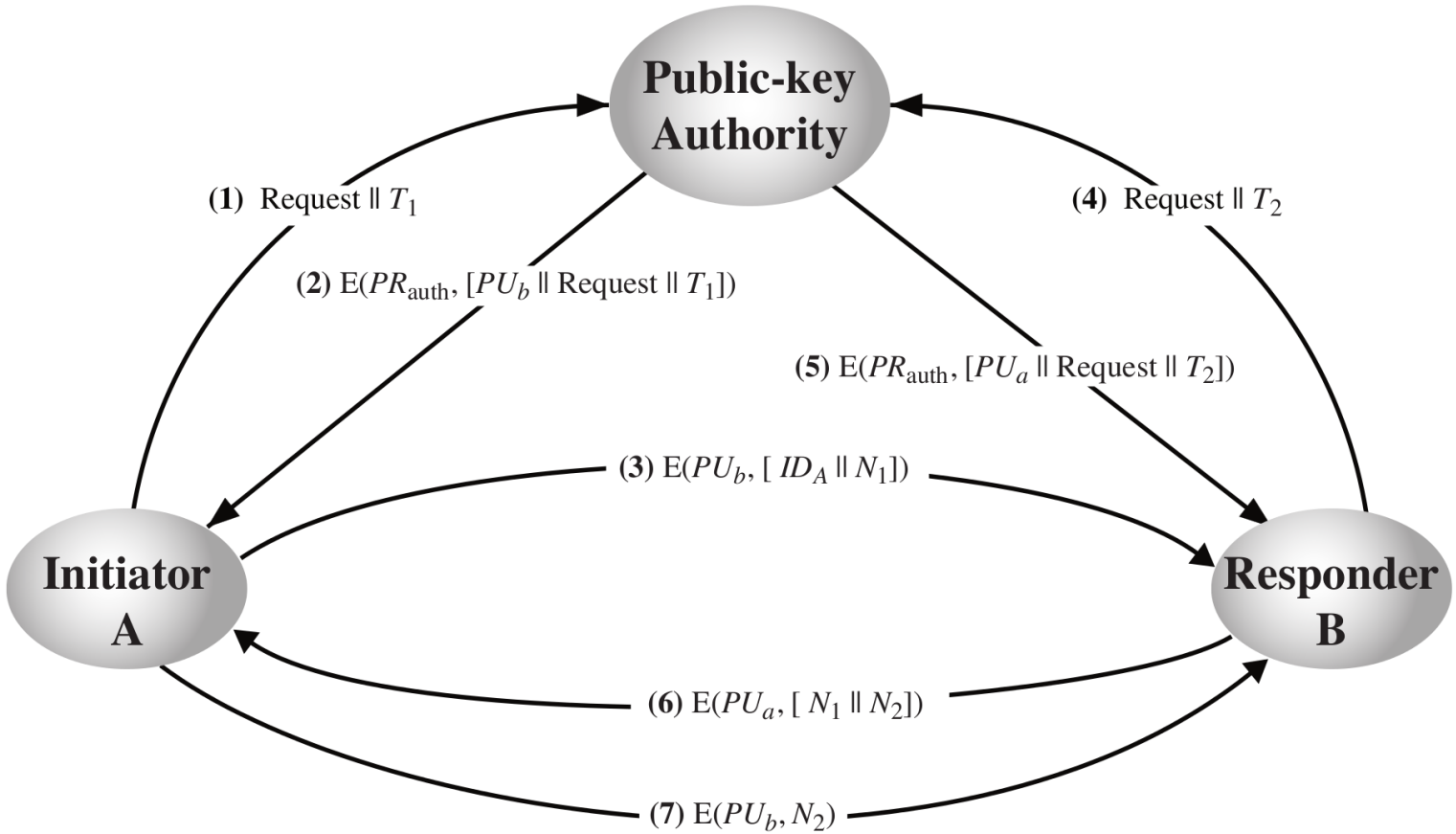
Publicly Available Directory

- All users publish keys in central directory
- Users must provide identification when publishing key
- Users can access directory electronically
- Weakness: directory must be secure



Public-Key Authority

- Specific instance of using publicly available directory
- Assume each user has already security published public-key at authority; each user knows authorities public key

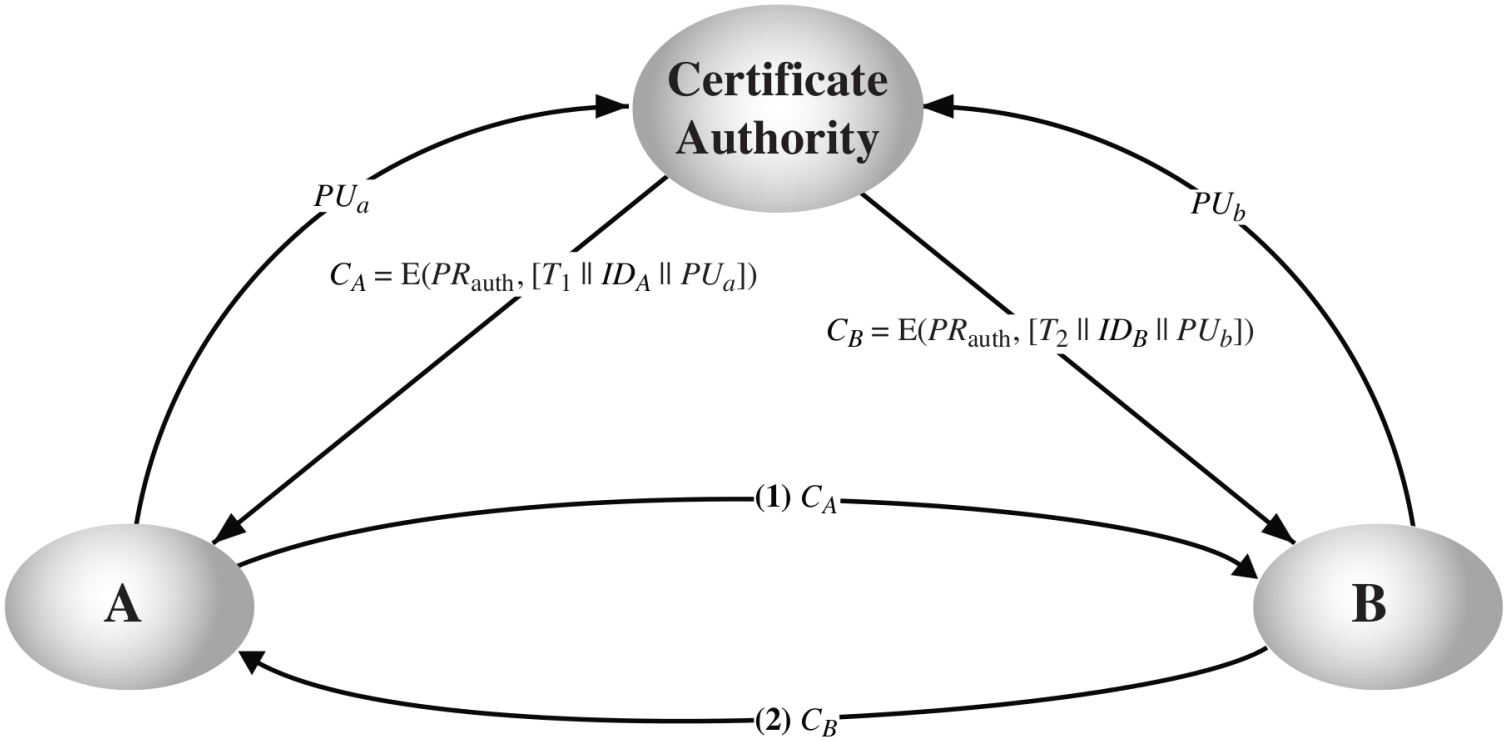


Public-Key Authority

- First 5 messages are for key exchange; last 2 are authentication of users
- Although 7 messages, public keys obtained from authority can be cached
- Problem: authority can be bottleneck
- Alternative: public-key certificates

Public-Key Certificates

- Assume public keys sent to CA can be authenticated by CA; each user has certificate of CA



Public Key Certificates

- A certificate is the ID and public-key of a user signed by CA

$$C_A = \mathrm{E}(PR_{auth}, [T||ID_A||PU_a])$$

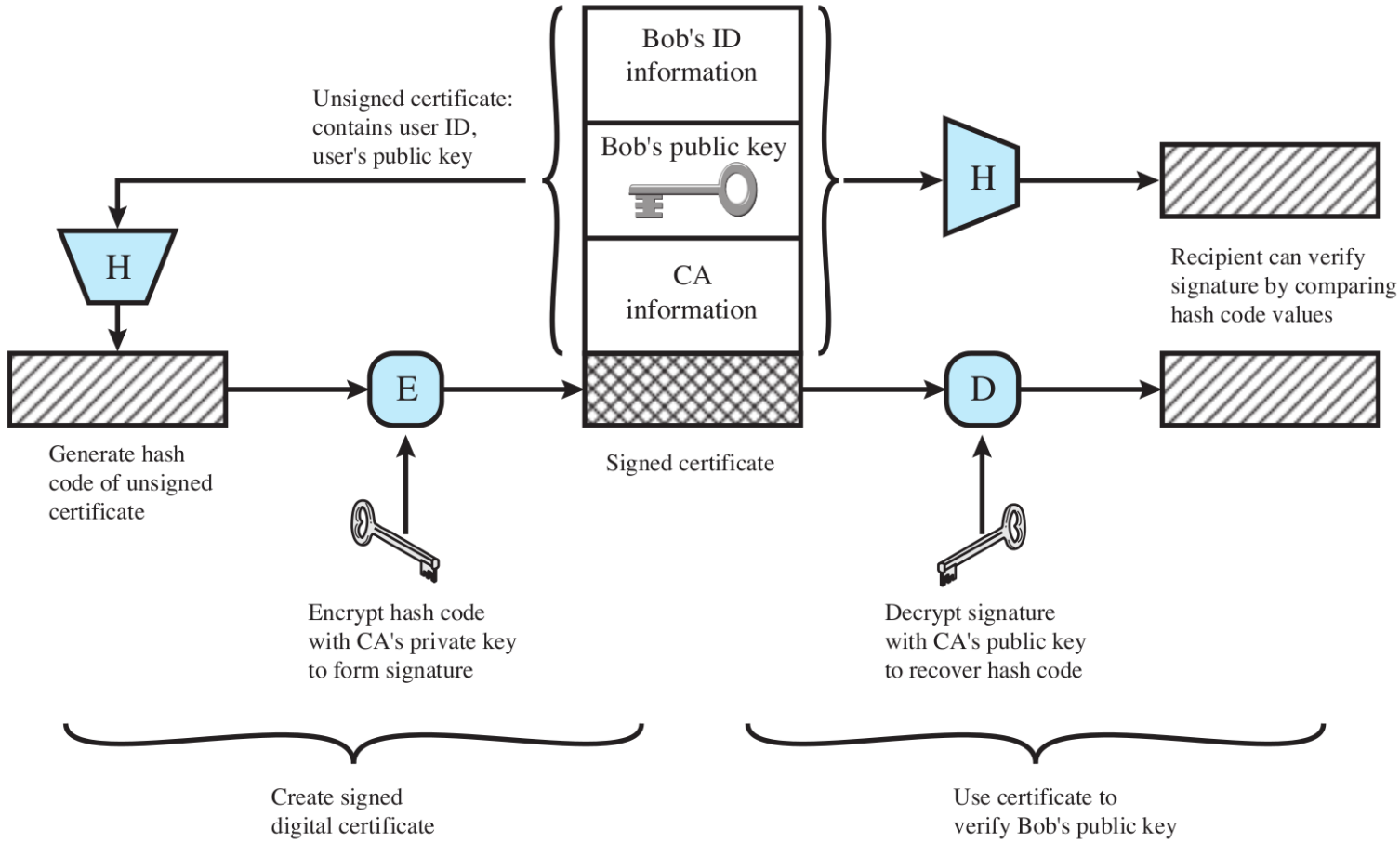
- Timestamp T validates currency of certificate (expiration date)
- Common format for certificates is X.509 standard (by ITU)
  - S/MIME (secure email)
  - IP security (network layer security)
  - SSL/TLS (transport layer security)
  - SET (e-commerce)

Contents

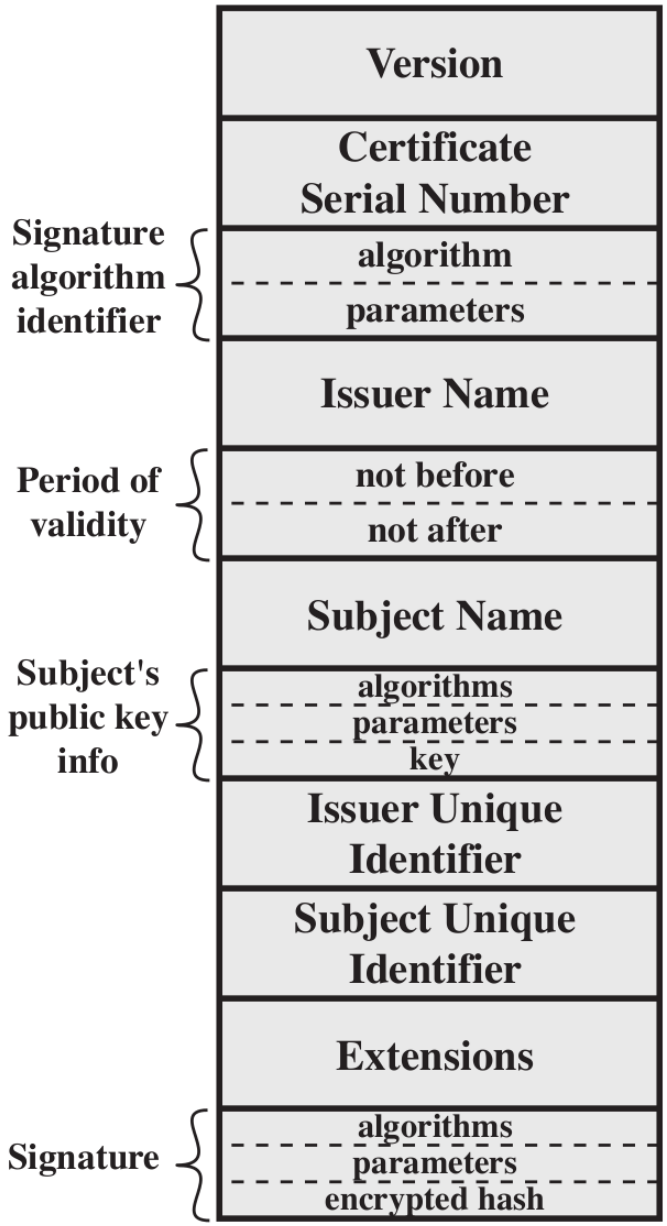
X.509 Certificates

- Each user has a certificate, although it is created by the Certificate Authority (CA)
- Certificates are stored in a public directory
- Certificate format includes:
  - Version of X.509 certificate
  - Signature algorithm
  - CA's name and unique identifier
  - Period of validity
  - User's name and unique identifier
  - User's public key information
  - Signature

Public-Key Certificate Use

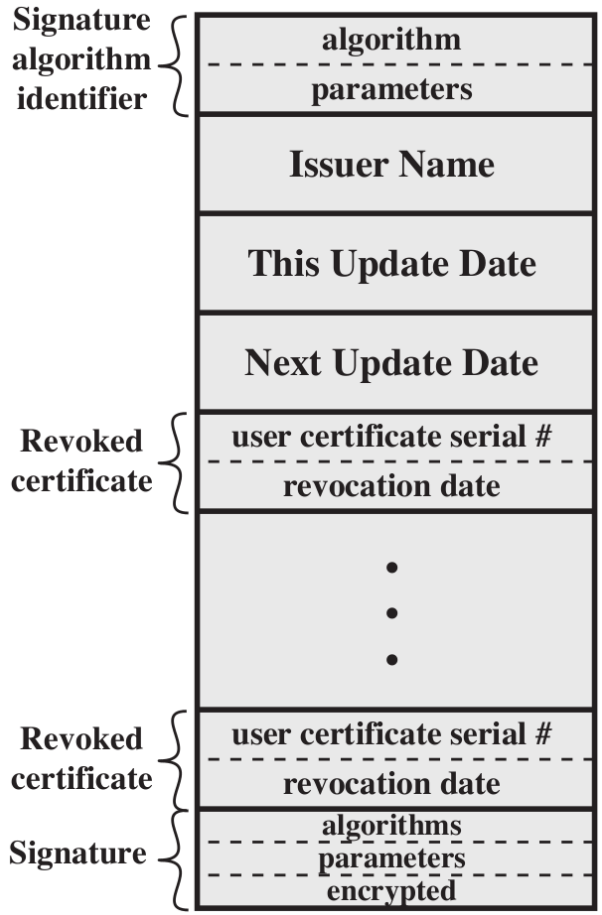


X.509 Formats



Certificate Revocation List

- Certificates may be revoked before expiry
- CA signs a CRL, which is stored in public directory



Multiple Certificate Authorities

- Multiple CA's can be arranged in hierarchy
- Notation: $Y << X >>$ certificate of X issued by CA Y