# 727 Crypto Management

# Lecture 12

GIOVANNI RUSSELLO

*G.RUSSELLO@AUCKLAND.AC.NZ*

# Key Distribution with Asymmetric Encryption

Merkle's Scheme

A generates (PU_A, PR_A),  B generates (PU_B, PR_B), K_s

A -> B: PU_A || ID_A

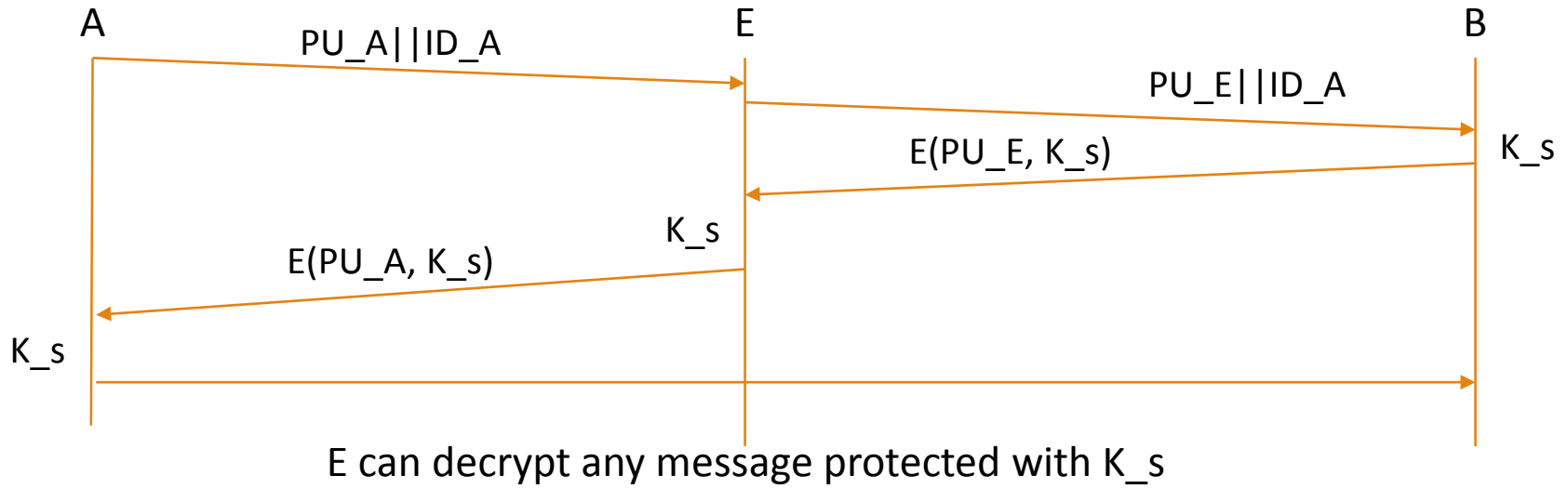
B: generates K_s

B -> A: E(PU_A, K_s)

A & B discard (PU_A, PR_A), (PU_B, PR_B)

After connection is closed discard K_s

# Key Distribution with Asymmetric Encryption - Issue



# Key Distribution with Confidentiality and Authentication

Assume that A and B already have securely exchanged PU_A, and PU_B

A -> B: E(PU_B, (N_1||ID_A))

B -> A: E(PU_A, (N_1||N_2))

A -> B: E(PU_B, N_2)

A -> B: E(PU_B, E(PR_A, K_s))

B uses PU_A to retrieve K_s

# Key Distribution with Confidentiality and Authentication

Assume that A and B already have securely exchanged PU_A, and PU_B

A -> B: E(PU_B, (N_1||ID_A)) <- N_1 is used for uniquely identify this transaction

B -> A: E(PU_A, (N_1||N_2))

A -> B: E(PU_B, N_2)

A -> B: E(PU_B, E(PR_A, K_s))

B uses PU_A to retrieve K_s

# Key Distribution with Confidentiality and Authentication

Assume that A and B already have securely exchanged PU_A, and PU_B

A -> B: E(PU_B, (N_1||ID_A))

B -> A: E(PU_A, (N_1||N_2)) <- N_1 assures A that the correspondent is B because only B could have decrypted the message

A -> B: E(PU_B, N_2)

A -> B: E(PU_B, E(PR_A, K_s))

B uses PU_A to retrieve K_s

# Key Distribution with Confidentiality and Authentication

Assume that A and B already have securely exchanged PU_A, and PU_B

A -> B: E(PU_B, (N_1||ID_A))

B -> A: E(PU_A, (N_1||N_2))

A -> B: E(PU_B, N_2) <- N_2 ensures B that A is the correspondent

A -> B: E(PU_B, E(PR_A, K_s))

B uses PU_A to retrieve K_s

# Key Distribution with Confidentiality and Authentication

Assume that A and B already have securely exchanged PU_A, and PU_B

A -> B: E(PU_B, (N_1||ID_A))

B -> A: E(PU_A, (N_1||N_2))

A -> B: E(PU_B, N_2)

A -> B: E(PU_B, E(PR_A, K_s)) <- (1) Only B can read this message; (2) Only A could have encrypted K_s

B uses PU_A to retrieve K_s

# Hybrid Scheme

IBM introduced a hybrid scheme for their mainframes where both symmetric and asymmetric systems are used

The scheme retains the use of KDC and use a public-key system for distributing the master keys

The main advantages are:

- Performance: given that master keys are not refreshed often the penalty of using asymmetric encryption is not that high
- Backwards compatibility: the hybrid scheme can be still used on existing symmetric only systems using a KDC with minimal changes

# Public-Key Distribution

Several schemes have been proposed

- Public announcement
- Publicly available directory
- Public-key authority
- Public-key certificates

# Public Announcement

Public-key main feature is that it is **public**

Users can announce and distribute their public-key to other users

Typical scenario is in PGP where users attach their public-keys to their messages

Main drawback

- Anyone can forge the announcement and impersonate a valid user
- Until the valid user A notifies the others users then the impostor can read all the messages sent from the other users intended to A

# Publicly Available Directory

A trusted authority or organisation maintains a table with (ID, PU) pairs

Participants would need to register with the authority either in person or through a secure authenticated communication

A participant has the right to change the public-key at any time

Users can query the directory for a public-key entry by specifying the user's id (it could be an email address)

Main issue here is to maintain the directory secure

- An adversary could subvert the authority and send out fake public-key impersonating the participants

# Public-Key Authority

This is the case of a directory with more strict security

Each participants knows the public-key of the authority

- The authority is responsible to keep its private-key secure

A -> PA: request||T_1

PA -> A: E(PR_PA, (PU_B||request||T_1))

A -> B: E(PU_B, (ID_A, N_1))

B ->PA: request||T_2

PA -> B: E(PR_PA, (PU_A||request||T_2))

B -> A: E(PU_A, (N_1||N_2))

A -> B: E(PU_B, N_2)

# Public-Key Authority

This is the case of a directory with more strict security

Each participants knows the public-key of the authority

- The authority is responsible to keep its private-key secure

A -> PA: request||T_1

PA -> A: E(PR_PA, (PU_B||request||T_1)) <- The use of PR_PA ensures A the PA generated the response

A -> B: E(PU_B, (ID_A, N_1))

B ->PA: request||T_2

PA -> B: E(PR_PA, (PU_A||request||T_2))

B -> A: E(PU_A, (N_1||N_2))

A -> B: E(PU_B, N_2)

# Public-Key Authority

This is the case of a directory with more strict security

Each participants knows the public-key of the authority

- The authority is responsible to keep its private-key secure

A -> PA: request||T_1

PA -> A: E(PR_PA, (PU_B||request||T_1)) <- The original request allows A to verify that his message was not altered by the PA

A -> B: E(PU_B, (ID_A, N_1))

B ->PA: request||T_2

PA -> B: E(PR_PA, (PU_A||request||T_2))

B -> A: E(PU_A, (N_1||N_2))

A -> B: E(PU_B, N_2)

# Public-Key Authority

This is the case of a directory with more strict security

Each participants knows the public-key of the authority

- The authority is responsible to keep its private-key secure

A -> PA: request||T_1

PA -> A: E(PR_PA, (PU_B||request||T_1)) <- T_1 allows A to determine this is not an old message

A -> B: E(PU_B, (ID_A, N_1))

B ->PA: request||T_2

PA -> B: E(PR_PA, (PU_A||request||T_2))

B -> A: E(PU_A, (N_1||N_2))

A -> B: E(PU_B, N_2)

# Public-Key Authority

This is the case of a directory with more strict security

Each participants knows the public-key of the authority

- The authority is responsible to keep its private-key secure

A -> PA: request||T_1

PA -> A: E(PR_PA, (PU_B||request||T_1))

A -> B: E(PU_B, (ID_A, N_1))

B ->PA: request||T_2

PA -> B: E(PR_PA, (PU_A||request||T_2))

B -> A: E(PU_A, (N_1||N_2)) <- N_1 ensures A that the message is coming from B

A -> B: E(PU_B, N_2)

# Public-Key Authority

This is the case of a directory with more strict security

Each participants knows the public-key of the authority

- The authority is responsible to keep its private-key secure

A -> PA: request||T_1

PA -> A: E(PR_PA, (PU_B||request||T_1))

A -> B: E(PU_B, (ID_A, N_1)) <- Only B could have decrypted this message and retrieve N_1

B ->PA: request||T_2

PA -> B: E(PR_PA, (PU_A||request||T_2))

B -> A: E(PU_A, (N_1||N_2))

A -> B: E(PU_B, N_2)

# Public-Key Authority

This is the case of a directory with more strict security

Each participants knows the public-key of the authority

- The authority is responsible to keep its private-key secure

A -> PA: request||T_1

PA -> A: E(PR_PA, (PU_B||request||T_1))

A -> B: E(PU_B, (ID_A, N_1))

B ->PA: request||T_2

PA -> B: E(PR_PA, (PU_A||request||T_2))

B -> A: E(PU_A, (N_1||N_2))

A -> B: E(PU_B, N_2) <- N_2 ensures B that A is the corresponding party

# Public-Key Authority

This is the case of a directory with more strict security

Each participants knows the public-key of the authority

- The authority is responsible to keep its private-key secure

A -> PA: request||T_1

PA -> A: E(PR_PA, (PU_B||request||T_1))

A -> B: E(PU_B, (ID_A, N_1))

B ->PA: request||T_2

PA -> B: E(PR_PA, (PU_A||request||T_2))

B -> A: E(PU_A, (N_1||N_2)) <- Only A could have decrypted this message

A -> B: E(PU_B, N_2)

# Public-Key Authority

This is the case of a directory with more strict security

Each participants knows the public-key of the authority

- The authority is responsible to keep its private-key secure

A -> PA: request||T_1

PA -> A: E(PR_PA, (PU_B||request||T_1))

A -> B: E(PU_B, (ID_A, N_1))

B ->PA: request||T_2

PA -> B: E(PR_PA, (PU_A||request||T_2))

B -> A: E(PU_A, (N_1||N_2))

A -> B: E(PU_B, N_2)

These 5 steps are executed infrequently because A and B can cache each other keys

# Public-Key Certificates

The PA still represents a bottleneck in the system and shares pretty much the same security issues as the public directory

As an alternative, Kohnfelder suggested in 1978 the idea of using digital certificates

- Allows participants to share public-key securely without involving a PA
- But with the same level of reliability

Simply put, a certificate consists of a (owner ID, the public-key)-pair signed with the private key of a trusted certificate authority (e.g., Comodo, Symantec, etc)

The user presents her public key to the CA in a secure way and then gets back the certificate that can be published

Anyone obtaining the certificate can retrieve the public key of a user and use the digital signature on the certificate to verify that is valid

# PK Certificate Requirements

1. Any user reading the certificates is able to determine the name and PK of the certificate owner
2. Any user can verify that the certificate originates from the CA and is not counterfeit
3. Only the CA can create and update the certificates
4. Any user can verify the currency of a certificate