Lecture 17

Key Management Services

Nicola Laurenti November 25, 2020

Lecture 17— Contents

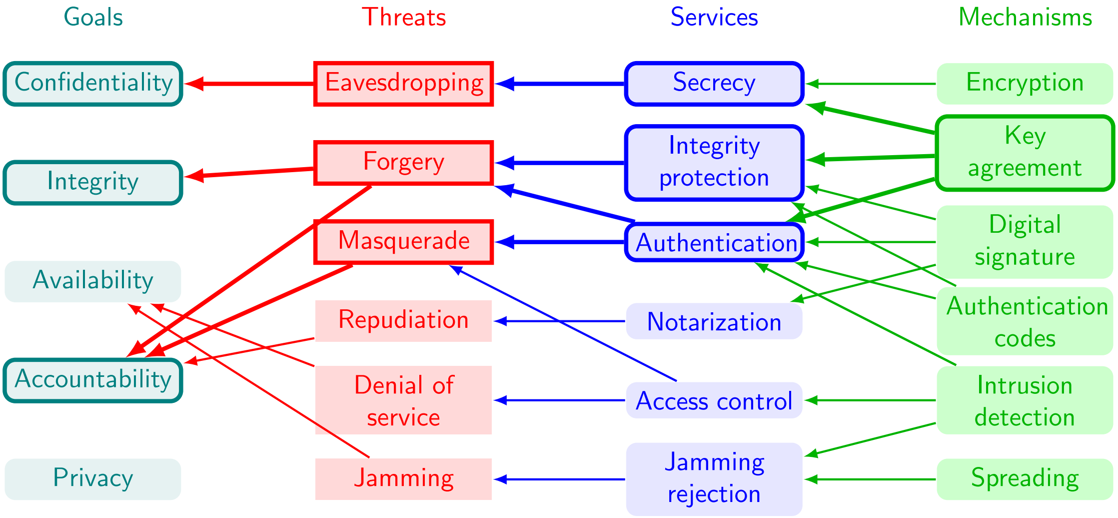
Key management services

Cryptographic key agreement

Arbitrated key distribution protocols

Public key infrastructure

Security goals, threats, services and mechanisms



Key management services

key generation

key distribution

key storage

key recovery

key renewal

revocation of captured keys

destruction of unused keys

Symmetric key distribution via asymmetric cryptography

Problem

A and B want to use a symmetric cryptographic mechanism
They need to share a symmetric key $k_\mathsf{s} \in \mathcal{K}_\mathsf{s}$

Resources

Assume they have asymmetric mechanisms in place for

- encryption, with private keys $k_{\mathsf{E,A}}, k_{\mathsf{E,B}} \in \mathcal{K}_\mathsf{E}$, public keys $k'_{\mathsf{E,A}}, k'_{\mathsf{E,B}} \in \mathcal{K}'_\mathsf{E}$
- digital signature, with private keys $k_{\mathsf{I,A}}, k_{\mathsf{I,B}} \in \mathcal{K}_\mathsf{E}$, public keys $k'_{\mathsf{I,A}}, k'_{\mathsf{I,B}} \in \mathcal{K}'_\mathsf{E}$

Symmetric key distribution via asymmetric cryptography

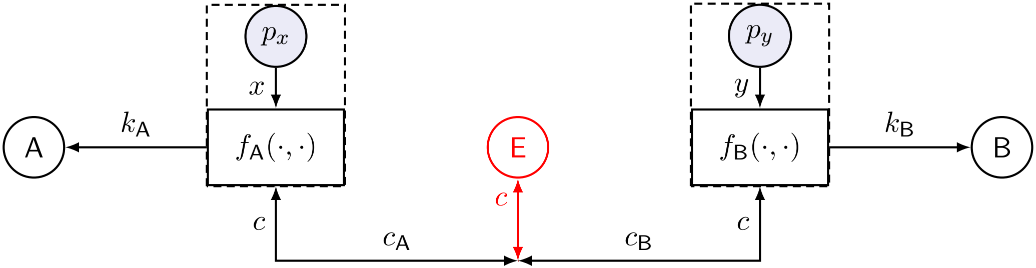
Solution

- A generates a random key $k_{\mathsf{s}} \sim \mathcal{U}(\mathcal{K}_{\mathsf{s}})$
- A encrypts (with the public key $k'_{\mathsf{E,B}}$) and signs (with his own private key $k_{\mathsf{I,A}}$) the generated key then sends it to B (e.g., by encrypt-then-sign, send $S(k_{\mathsf{I,A}}, E(k'_{\mathsf{S,B}}))$) encryption,
- B verifies (with the public key $k'_{\mathsf{I,A}}$) and decrypts (with his own private key $k_{\mathsf{E,B}}$) the received message, obtianing the symmetric key

Problem

If $k_{\mathsf{E,B}}$ is later discovered by an attacker, k_{s} becomes exposed, i.e., there is no forward secrecy

Cryptographic key agreement [Diffie-Hellman, '76]



Requirements

uniformity $k_\mathsf{A} \sim \mathcal{U}(\mathcal{K})$

correctness $k_\mathsf{A} = k_\mathsf{B}$

secrecy k_A independent of $(c_\mathsf{A}, c_\mathsf{B})$

Algorithm

Choose a group $(\mathbb{G}, \circ)$ where the finite log problem is hard. Let $n = |\mathbb{G}|$ and $g \in \mathbb{G}$ primitive

$$x, y \text{ i.i.d.} \sim \mathcal{U}(\mathbb{Z}_n \setminus \{0\}) \quad , \quad f_\mathsf{A} : \begin{cases} c_\mathsf{A} = g \overset{x}{\circ} \\ k_\mathsf{A} = c_\mathsf{B} \overset{x}{\circ} \end{cases} \quad , \quad f_\mathsf{B} : \begin{cases} c_\mathsf{B} = g \overset{y}{\circ} \\ k_\mathsf{B} = c_\mathsf{A} \overset{y}{\circ} \end{cases}$$

Typical groups: multiplicative integers $(\mathbb{Z}_p, \times)$, elliptic curve $(\mathcal{E}, \circ)$

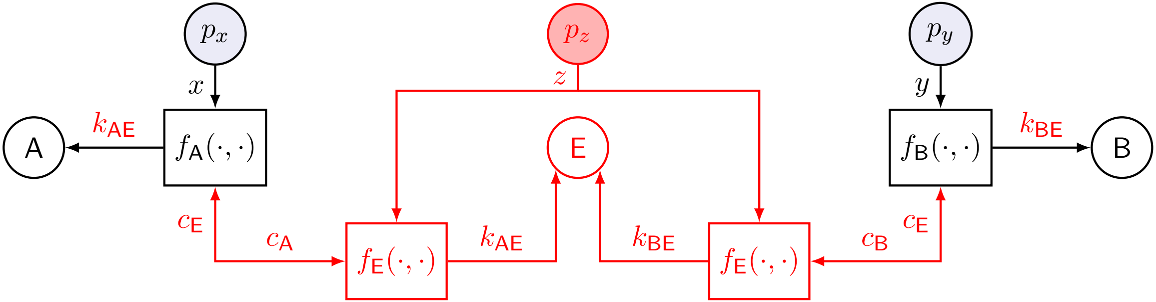
Result

correctness : $k_\mathsf{A} = \left(g \overset{y}{\circ}\right) \overset{x}{\circ} = g \overset{xy \bmod n}{\circ} = \left(g \overset{x}{\circ}\right) \overset{y}{\circ} = k_\mathsf{B}$

computational secrecy : infeasible to derive x, y, k from c

uniformity : $x, y \sim \mathcal{U}(\mathbb{Z}_n \setminus \{0\})$
$\Rightarrow xy \bmod n \sim \mathcal{U}(\mathbb{Z}_n \setminus \{0\})$
$\Rightarrow k \sim \mathcal{U}(\mathbb{G} \setminus \{0_\mathbb{G}\})$

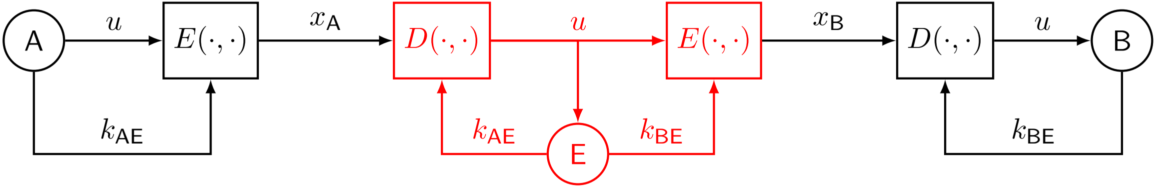
Man-in-the-middle attack



- E generates $z \in \mathbb{Z}_n \setminus \{0\}$ and computes $c_{\mathsf{E}} = g \overset{z}{\circ}$
- E intercepts $c_{\mathsf{A}}, c_{\mathsf{B}}$ and replaces each with c_{E}
- A receives c_{E}, computes $k_{\mathsf{AE}} = c_{\mathsf{E}} \overset{x}{\circ} = g \overset{xz \bmod n}{\circ}$
- B receives c_{E}, computes $k_{\mathsf{BE}} = c_{\mathsf{E}} \overset{y}{\circ} = g \overset{yz \bmod n}{\circ}$
- E can compute $k_{\mathsf{AE}} = c_{\mathsf{A}} \overset{z}{\circ} = g \overset{xz \bmod n}{\circ}$ as well as $k_{\mathsf{BE}} = c_{\mathsf{B}} \overset{z}{\circ} = g \overset{yz \bmod n}{\circ}$

Man-in-the-middle attack

If the attack succeeds, E can violate the symmetric protocol between A and B without them knowing



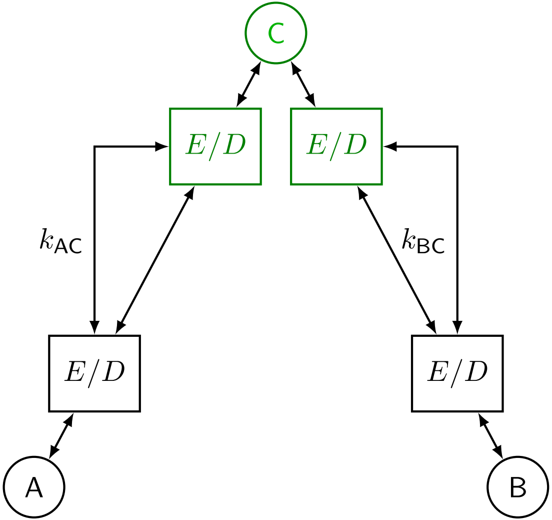
In order to avoid the man-in-the-middle attack, messages $c_{\mathsf{A}}, c_{\mathsf{B}}$ must be authenticated and integrity protected

Is this a "chicken and egg" problem? No, can use digital signature

Forward secrecy

Even if E later learns the authentication provate keys, he will no longer be able to retrieve k_{A}

Needham-Schroeder symmetric protocol



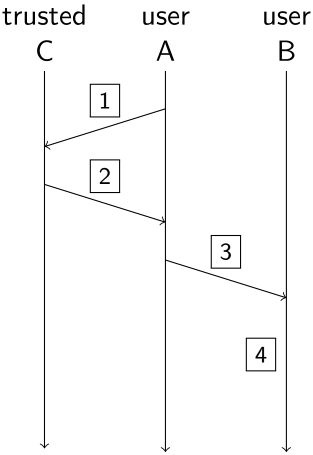
entities two parties A and B, a trusted third party C

tools a symmetric encryption mechanism $E(\cdot,\cdot)$ with keys shared between A and C, and between B and C; random generators at all entities

aim to securely distribute a key k_{AB} between A and B for a symmetric mechanism

Needham-Schroeder symmetric protocol (cont.)

phase I: key distribution



$\boxed{1}$ A : generates nonce n_A

A $\rightarrow$ C : $u_1 = (\mathrm{id}_\mathsf{A}, \mathrm{id}_\mathsf{B}, n_\mathsf{A})$

$\boxed{2}$ C : generates $k_\mathsf{AB} \sim \mathcal{U}(\mathcal{K})$

encrypts $x_2 = E_{k_\mathsf{BC}}([\mathrm{id}_\mathsf{A}, k_\mathsf{AB}])$

and $x_3 = E_{k_\mathsf{AC}}([n_\mathsf{A}, \mathrm{id}_\mathsf{B}, k_\mathsf{AB}, x_2])$

C $\rightarrow$ A : x_3

$\boxed{3}$ A : decrypts $[n_\mathsf{A}, \mathrm{id}_\mathsf{B}, k_\mathsf{AB}, x_2] = D_{k_\mathsf{AC}}(x_3)$

checks n_A and id_B

A $\rightarrow$ B : x_2

$\boxed{4}$ B : $[\mathrm{id}_\mathsf{A}, k_\mathsf{AB}] = D_{k_\mathsf{BC}}(x_2)$

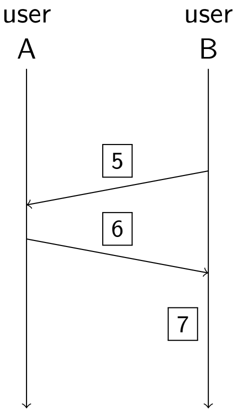
$u_1 - \{id_A \; id_B \; n_A\}$

k_{AB}

$x_2 = E(id_A, k_{AB}]$

Needham-Schroeder symmetric protocol (cont.)

phase II: key confirmation



$\boxed{5}$ B : generates random $r_{\mathsf{B}} \sim \mathcal{U}(\mathcal{R})$
encrypts $x_4 = E_{k_{\mathsf{AB}}}(r_{\mathsf{B}})$
B $\rightarrow$ A : x_4

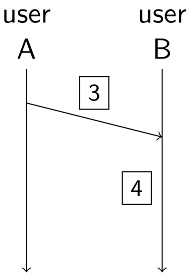
$\boxed{6}$ A : decrypts $r_{\mathsf{B}} = D_{k_{\mathsf{AB}}}(x_4)$
computes $r_{\mathsf{A}} = r_{\mathsf{B}} - 1$
encrypts $x_5 = E_{k_{\mathsf{AB}}}(r_{\mathsf{A}})$
A $\rightarrow$ B : x_5

$\boxed{7}$ B : decrypts $r_{\mathsf{A}} = D_{k_{\mathsf{AB}}}(x_5)$
checks if $r_{\mathsf{A}} = r_{\mathsf{B}} - 1$

The Dennning-Sacco attack

Assume that E has learnt an old $k_{\mathsf{AB}}^{\mathrm{old}}$ somehow and that he had recorded the corrsponding session between A and B

Then, E can replay message $\boxed{3}$ to B



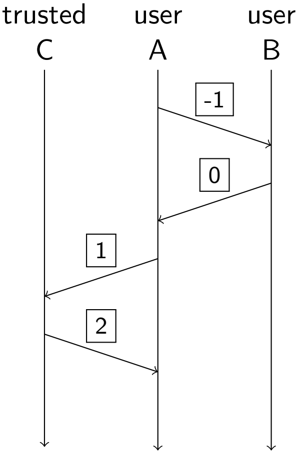
$$\boxed{3}\ \mathsf{E} \rightarrow \mathsf{B} : \ x_2^{\mathrm{old}}$$

$$\boxed{4}\ \mathsf{B} : \ [\mathrm{id}_{\mathsf{A}}, k_{\mathsf{AB}}^{\mathrm{old}}] = D_{k_{\mathsf{BC}}}(x_2^{\mathrm{old}})$$

B will use $k_{\mathsf{AB}}^{\mathrm{old}}$ as if it were a good new key shared with A

solution needs a nonce known to B, too

The Otway-Rees protocol (or Needham-Schroeder revised)



$\boxed{\text{-1}}$ A $\rightarrow$ B : id_A

$\boxed{0}$ B : generates nonce n_B
encrypts $x_1 = E_{k_\mathsf{BC}}([\mathrm{id}_\mathsf{A}, n_\mathsf{B}])$
B $\rightarrow$ A : x_1

$\boxed{1}$ A : generates nonce n_A
A $\rightarrow$ C : $u_1 = (\mathrm{id}_\mathsf{A}, \mathrm{id}_\mathsf{B}, n_\mathsf{A}, x_1)$

$\boxed{2}$ C : decrypts $(\mathrm{id}_\mathsf{A}, n_\mathsf{B}) = D_{k_\mathsf{BC}}(x_1)$
generates $k_\mathsf{AB} \sim \mathcal{U}(\mathcal{K})$
encrypts $x_2 = E_{k_\mathsf{BC}}([\mathrm{id}_\mathsf{A}, k_\mathsf{AB}])$
and $x_3 = E_{k_\mathsf{AC}}([n_\mathsf{A}, \mathrm{id}_\mathsf{B}, k_\mathsf{AB}, x_2])$
C $\rightarrow$ A : x_3

then continue as the original Needham-Schroeder...

Kerberos

Kerberos is basically a double symmetric Needham-Schroeder key distribution protocol (not Otway-Rees) where:

phase I A is the client
B is the ticket granting server
C is the authentication server

phase II A is the client
B is the target service server
C is the ticket granting server

and the Dennings-Sacco attack is limited by the use of a validity time

Needham-Schroeder asymmetric protocol

entities two parties A and B, a trusted third party C

tools asymmetric encryption mechanisms (E', D) for A and B, and digital signature mechansim for C

assumptions both A and B know k'_C, C knows both k'_A and k'_B

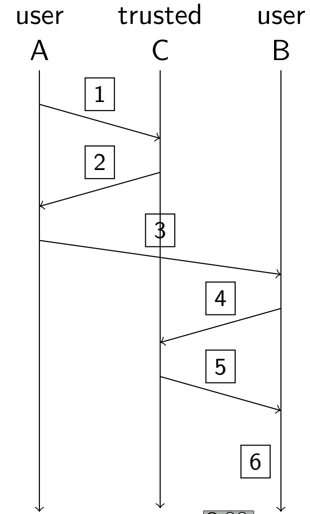
aim for A and B to securely learn each other's public key for use with asymmetric mechanisms

PUBLIC

K = privata
K' = pub

Needham-Schroeder asymmetric protocol (cont.)

phase I: key distribution



È un check su public key dei others

$\boxed{1}$ $\mathsf{A} \to \mathsf{C}: \quad u_1 = (\mathrm{id}_\mathsf{A}, \mathrm{id}_\mathsf{B})$

$\boxed{2}$ $\mathsf{C}:$ signs $x_2 = S_{k_\mathsf{C}}([\mathrm{id}_\mathsf{B}, k'_\mathsf{B}])$

C firma con la sua chiave privata

$\mathsf{C} \to \mathsf{A}: \quad x_2$

$\boxed{3}$ $\mathsf{A}:$ verifies $[\mathrm{id}_\mathsf{B}, k'_\mathsf{B}, \hat{b}] = V'_{k'_\mathsf{C}}(x_2)$

verifica se id e public key corrispondono

generates nonce $n_\mathsf{A} \sim \mathcal{U}(\mathcal{R})$

encrypts $x_3 = E'_{k'_\mathsf{B}}([n_\mathsf{A}, \mathrm{id}_\mathsf{A}])$

$\mathsf{A} \to \mathsf{B}: \quad x_3$

$\boxed{4}$ $\mathsf{B}: \quad [n_\mathsf{A}, \mathrm{id}_\mathsf{A}] = D_{k_\mathsf{B}}(x_3)$

$\mathsf{B} \to \mathsf{C}: \quad u_4 = (\mathrm{id}_\mathsf{A}, \mathrm{id}_\mathsf{B})$

$\boxed{5}$ $\mathsf{C}:$ signs $x_5 = S_{k_\mathsf{C}}([\mathrm{id}_\mathsf{A}, k'_\mathsf{A}])$

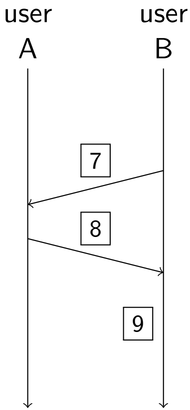
$\mathsf{C} \to \mathsf{B}: \quad x_5$

$\boxed{6}$ $\mathsf{B}:$ verifies $[\mathrm{id}_\mathsf{A}, k'_\mathsf{A}, \hat{b}] = V'_{k'_\mathsf{C}}(x_5)$

Needham-Schroeder asymmetric protocol (cont.)

Verifica

phase II: handshake



$\boxed{7}$ B : generates nonce $r_{\mathsf{B}} \sim \mathcal{U}(\mathcal{R})$
encrypts $x_6 = E'_{k'_{\mathsf{A}}}([r_{\mathsf{A}}, r_{\mathsf{B}}])$
B $\rightarrow$ A : x_6

$\boxed{8}$ A : decrypts $[r_{\mathsf{A}}, r_{\mathsf{B}}] = D_{k_{\mathsf{A}}}(x_4)$
checks n_{A}
encrypts $x_7 = E'_{k'_{\mathsf{B}}}(r_{\mathsf{B}})$
A $\rightarrow$ B : x_7

$\boxed{9}$ B : decrypts $r_{\mathsf{B}} = D_{k_{\mathsf{B}}}(x_7)$
checks r_{B}

The Lowe attack and countermeasure

This attack allows an internal E to

- use an authentic exchange with A
- to impersonate A in an exchange with B
- by re-encrypting message $\boxed{3}$ to B, and using A as an oracle for $\boxed{7}$, $\boxed{8}$

The solution is to include id_{B} in message $\boxed{7}$ and $\boxed{8}$

Certificates

A certificate for A, issued by a certification authority C is a message, signed by C with its private key, and bearing

$$x = S(k_{\mathsf{C}}, u) \quad , \qquad \text{with } u = (\mathrm{id}_{\mathsf{C}}, \mathrm{id}_{\mathsf{A}}, k'_{\mathsf{A}})$$

Thus any entity B that knows the public key k'_{C} (and trusts C) can learn and verify the public key k'_{A} from x: the certficate binds k'_{A} to id_{A}

A can itself send x around to the entities it is interested in connecting with, or publish it on a public bulletin

Typically, u also includes other data, such as validity dates, protocol specifications, cryptographic parameters, etc.

Public key hierarchy

What if B does not know or trust C?

Then it needs another authority C' to provide a certificate for C, which in turn will provide a certficate for A, etc.

A set of cetification authorities structured into layers where higher authorities provide certificates for lower authorities is called a public key hierarchy

Public Key Infrastructure (PKI)

A PKI defines protocols, policies and mechanisms needed to guarantee the authentication of public keys:

- certificate formats
- relationships between certification authorities and users
- policies for issuing and revocation of certificates

An example of PKI standard is defined in ITU-T X.509

Summary

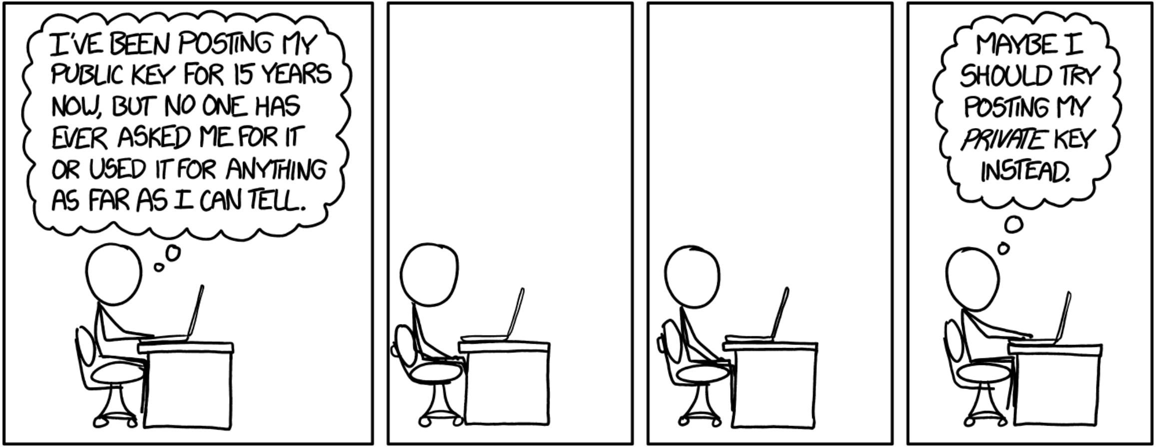
In this lecture we have:

- introduced key management services
- presented a general model for key agreement and described the Diffie-Hellman protocol
- described the Needham-Schroeder protocol for key distribution and its variants
- introduced public key infrastructures, made of certificates and hierarchies

Assignment

- class notes
- textbook, §11.1, §12.2, §11.3, §8.6

End of lecture



Public Key, reproduced from xkcd URL: xkcd.com/1553