Security Protocols

Outline

- What is an security protocol

    key exchange protocols

    authentication protocols

- Learn how to design one and how to make mistakes through examples

- With both crypto technologies: SK and AK

- Formal specification and correctness of protocols

Security Protocols

A security protocol is a well defined and ordered sequence of msg with a start and an end plus context domain and assumptions.

A security protocol offers one or more security properties.

Security Protocols

Assumptions about:

- Information known by the parties before the protocol starts
- Threats (e.g., passive or attive attacks, etc.)
- Capabilities and competence of the participating parties
- Environment conditions
- etc…

Security protocols

- security protocols can involve more than 2 parties. Most of the case the extension is not straightforward

Notation

$A \rightarrow B$: msg   msg sent by A to B

K_{AB} symmetric key shared by A and B

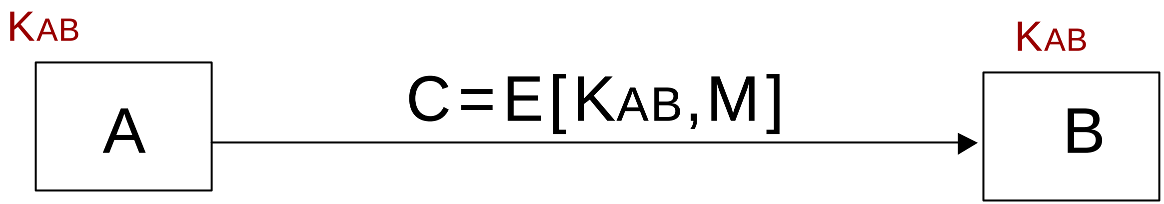
$E[K_{AB}, m]$ symmetric-key encryption of msg *m* using key K_{AB}

K_{A-} A's private (decryption) key

K_{A+} A's public (encryption) key

$[m]K_{B+}$ asymmetric key encryption of msg *m* using B's public key

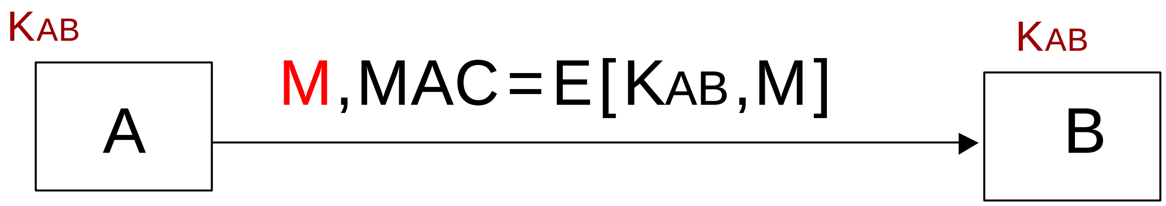
Confidentiality with SK



$$M=D[K_{AB},C]=D[K_{AB},E[K_{AB},M]]$$

Encryption/decryption

Integrity with SK



$$MAC \stackrel{?}{=} E[K_{AB}, M]$$

MAC=message authentication code

Confidentiality and Integrity with AK

K_{A+}, K_{A-}, K_{B+} $\qquad$ K_{B+}, K_{B-}, K_{A+}

$$A \xrightarrow{C=[M]K_{B+}} B$$

$$M=[C]K_{B-} = [[M]K_{B+}]K_{B-}$$

encryption

K_{A+}, K_{A-}, K_{B+} $\qquad$ K_{B+}, K_{B-}, K_{A+}

$$A \xrightarrow{M,[M]K_{A-}} B$$

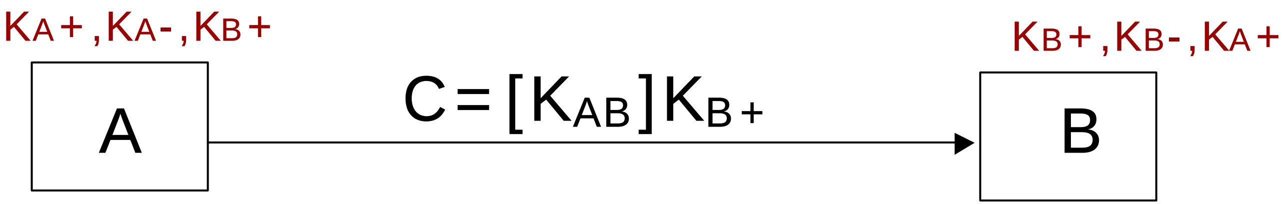
$$M' = [[M]K_{A-}]K_{A+} \stackrel{?}{=} M$$

integrity

Combination of SK and AK

K_A+, K_B+ = public keys
K_A-, K_B- = private keys

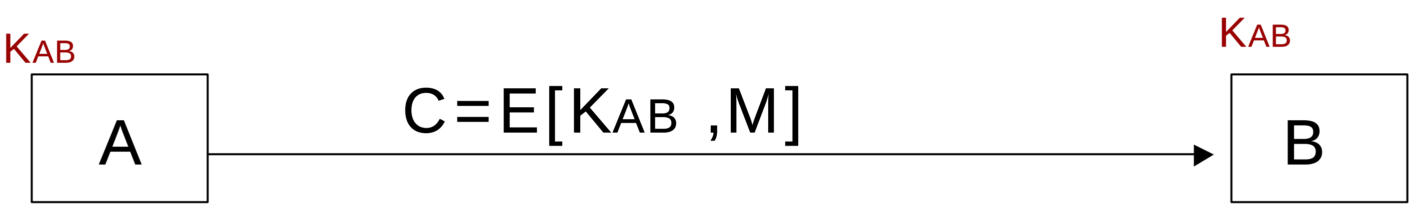
AK



$[C]K_{B-} = K_{AB}$ *shared key*

Shared Key Encryption

SK



$M = D[K_{AB}, C] = D[K_{AB}, E[K_{AB}, M]]$

Bulk message encryption

Key exchange protocols

- The goal of a key exchange protocol is for the participants to share a common cryptographic key at the end of the protocol run.

Diffie-Hellman

n large prime such that g is primitive mod n

$A \rightarrow B$: $g^x \bmod n$ $\quad x, g^{xy}$

$B \rightarrow A$: $g^y \bmod n$ $\quad y, g^{xy}$

Shared key $(g^x)^y = (g^y)^x = g^{xy} \bmod n$

$A \rightarrow B$: $[msg_1]g^{xy}$

$B \rightarrow A$: $[msg_2]g^{xy}$

Man-in-the-middle attack

A $\rightarrow g^x$

C(A) $\rightarrow g^z$ **B**

$g^y \leftarrow$ **B**

A $g^z \leftarrow$ **C(B)**

g^{xz} *g^{zy}*

A $\rightarrow$ [msg] g^{xz}

C $\rightarrow$ [msg] g^{zy} **B**

[msg] $g^{zy} \leftarrow$ **B**

A [msg] $g^{zx} \leftarrow$ **C**

Authenticated D-H

A and B shared a pre-arranged secret S

$A \rightarrow B$: $g^x \bmod n$ $\quad g^{xy}$

$B \rightarrow A$: $g^y \bmod n$ $\quad g^{xy}$

$A \rightarrow B$: $h(A,S,g^{xy})$

$B \rightarrow A$: $h(B,S,g^{xy})$

Authentication protocols

- authentication protocols aim to assure the two party about the electronic identity of the other one. At the end of the protocol run at least one of the party must be sure about the electronic identity of the other one.

Authentication protocols

- One-way athentication (login)
- Mutual authentication (SSL, AH)

- Many authentication protocol are also key exchange protocol $\rightarrow$ authenticated key exchange. Parties can establish a secure channel after the run of the protocol

Authentication

human being
(is he really who he claims to be?)

Entity authentication

machine
(is it really workstation X?)

Data authentication (is the message authentic?)

Authentication

- What I know → pwd (weak)
- What I have → key (strong)
- What I am → biometrics (strong)

Two factor authentication: the combination of two of them. Example debit bank card: pwd(pin)+key(card)

Authentication protocols

$A \rightarrow B$: I am A

$B \rightarrow A$: I am B

Fine sometimes but in general anybody can claim to be A or B

$C \rightarrow B$: I am A

$B \rightarrow C$: I am B

Login protocols

$A \rightarrow$ Host: password

Host computes h(password)

h(passwords) = stored value ?

h(passwords) instead of password in clear

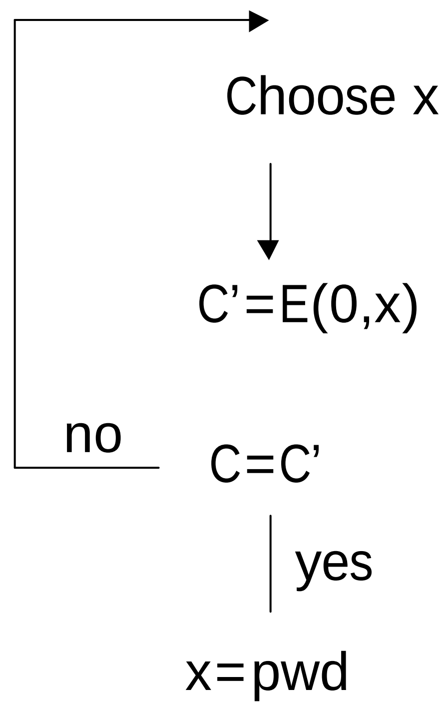
to prevent pwd exposure in case host is attacked

Password-based login protocols

- On-line attacks
- Off-line attacks

k known and fixed, E known

pwd $\longrightarrow$ C=E(0,pwd)



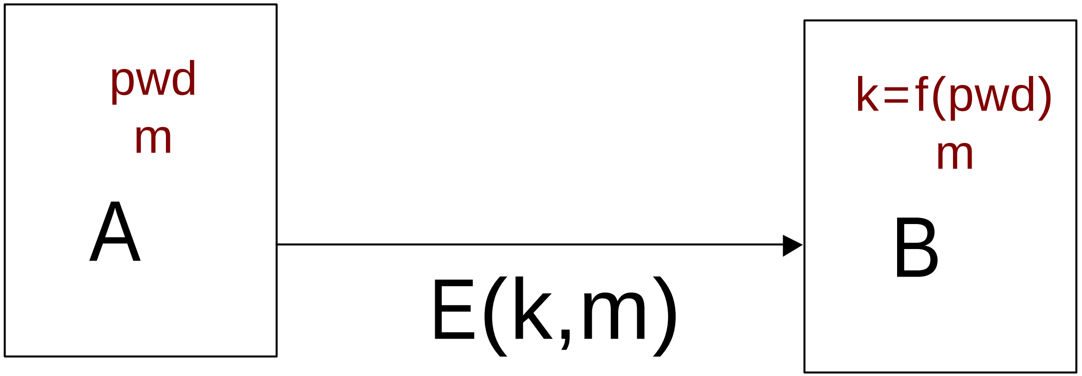
- Encryption vs Hash

Solutions

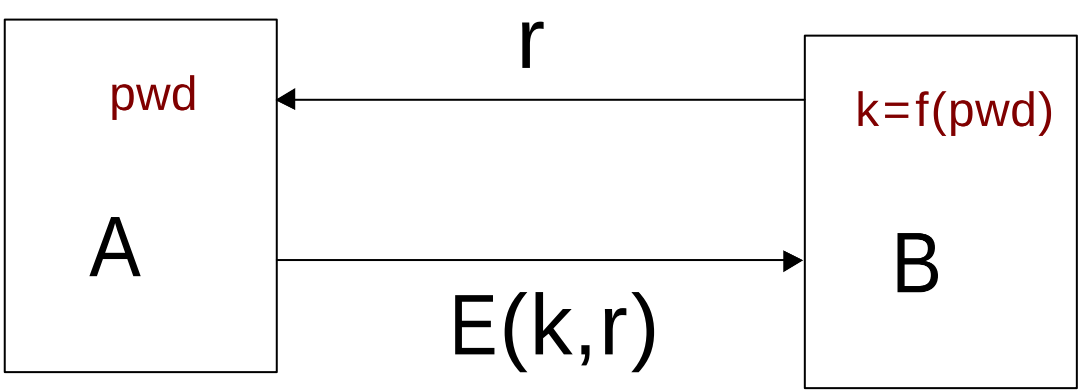
- On-line attacks
  - Limit the number of possible attempts $\rightarrow$ denial of service potential problem
  - Slow input procedures

- Off-line attacks
  - Dictionary attack prevented by the use of salt to increase entropy
  - Prudent to add also server id

Remote login



Attacker can simply reuse old E(k,m) to impersonate A



Same as before but now online attack only. Reading B's database allows impersonation of A

Remote login: Lamport's hash

r random

Host computes

$h(r), h(h(r))=h^2(r),\ldots, h^i(r)$

and gives these values to Alice. They are one-time password

Host stores $y=h^{i+1}(r)$

Lamport's hash (S/Key)

$h(r)$, $h(h(r))=h^2(r),\ldots, h^i(r)$

A $\rightarrow$ Host: $h^i(r)$

Host computes $h(h^i(r))$

if $h(h^i(r))=y$ success

Host replace y with $h^i(r)$

robust against eavesdropping and break host's security

- One way authentication only
- Still possible to hijack current session