Lampart's protocol

Objective : A identifies herself to B

Use a one way function H

Notation : $H^k(\omega) = \underbrace{H(H(\dots(\omega))}_{k \text{ times}}$

Initial parameters : t : max number of identifications ($t = 100, 1000$)

A chooses an initial password ω

A transfers $\omega_0 = H^t(\omega)$ to B

B initializes his counter for A to $i_A = 1$

Protocol actions for session : $i = 1, \dots, t$

A computes $\omega_i = H^{t-i}(\omega)$; transfers to B : (A, i, ω_i)

B checks that $i = i_A$ and $\omega_{i-1} = H(\omega_i)$

If both checks succeed, B accepts and sets $i_A \leftarrow i_A + 1$ and stores ω_i.

12.2 Challenge-Response Identification (C-R- Identif.)

Basic ideas:

1. A (the claimant) proves her identity to B (the verifier) by demonstrating knowledge of a secret known only to her without revealing the secret
2. The ~~request~~ response is requested by a time-varying challenge
3. The response from one execution does not provide information for a subsequent identification as subsequent challenges will differ.

12.2.1] C-R- Identif by symmetric key encryption
Techniques from ISO/IEC 9798-2 are described

Notation: E_K : symmetric encryption alg. with key K
t_A : time stamp generated by A
r_A, r_B : random numbers by A, B
$A \to B$: A transmits s.th to B
$C(\ ,\ )$: concatenation
$.^*$: optional elements are (may be added to ".")

- Unilateral authentication, time-stamp-based

$A \to B : E_K(t_A, B^*)$

Including the identifier B prevents an adversary from reusing the message immediately on A.

- Unilateral authentication, random numbers

$A \leftarrow B : r_B$ (1)
$A \to B : E_K(r_B, B^*)$ (2)

B decrypts (2) verifies r_B from (1). Inclusion of B avoids a reflection attack.

$O \leftarrow B : r_B$ (first protocol)

$O \rightarrow B : r_B$ (open a 2nd protocol)

$O \leftarrow B : E_k(r_B, A^*)$ (in the 2nd protocol: (2))

$O \rightarrow B : E_k(r_B, B^*)$ (in the first protocol: (2))

Avoided by including A^* or B^*.

Use k_1, k_2 or (as in the protocol) include names (A^* and B^*)

(and don't use same names)

A is not involved at all.

- Mutual authentication, random numbers

$A \leftarrow B : r_B$ (1)

$A \rightarrow B : E_k(r_A, r_B, B^*)$ (2)

$A \leftarrow B : E_k(r_B, r_A)$ (3)

B decrypts (2), verifies r_B from (1), obtains r_A, encrypts (r_B, r_A)

A decrypts (3), verifies r_B and r_A

r_A might be used as a shared secret key.

12.2.2 C-R-Identif. by public-key techniques

Principle: The claimant decrypts a challenge encrypted by her public key

Notation: h: Hash function, E_A Encryption under A's public key

$A \leftarrow B : h(r_B), B, E_A(r_B, B)$

$A \rightarrow B : r_B$

B chooses a random r_B, computes the witness $h(r_B)$ without revealing r_B, compute the challenge $E_A(r_B, B)$

A decrypts $E_A(r_B, B)$ to recover r', B', computes $h(r')$

If $h(r') = h(r_B)$ and $B' = B$ then A sends $r' = r_B$ to B.

12.2.3 | C-R-Identif. based on digital signatures

Principle: The claimant signs a challenge digitally

Notation: S_A : signature key A
$cert_A$: certificate which contains the authentic public signature key

Protocols are from ISO/IEC 9798-3

- Unilateral with timestamps

$A \to B : cert_A, t_A, B, S_A(t_A, B)$

B verifies that the timestamp is acceptable, the correct identifier B checks that the signature over (t_A, B) is correct.

- Unilateral with random numbers

$A \leftarrow B : r_B$

$A \to B : cert_A, r_A, B, S_A(r_A, r_B, B)$

B verifies its own identification, checks validity of A's signature over (r_A, r_B, B)

- Mutual authentication with random numbers

$A \leftarrow B : r_B$

$A \to B : cert_A, r_A, B, S_A(r_A, r_B, B)$

$A \leftarrow B : cert_B, A, S_B(r_B, r_A, A)$

B verifies as above. A knows r_A and r_B verifies the signature of B over (r_B, r_A, A)

12.3 Kerberos

Kerberos: Three headed dog guarding the underworld in Greek mythology.

Grew out of a larger „Athena" at MIT

Purpose: To provide strong levels of authentication and security in key exchange between servers and clients in a network.

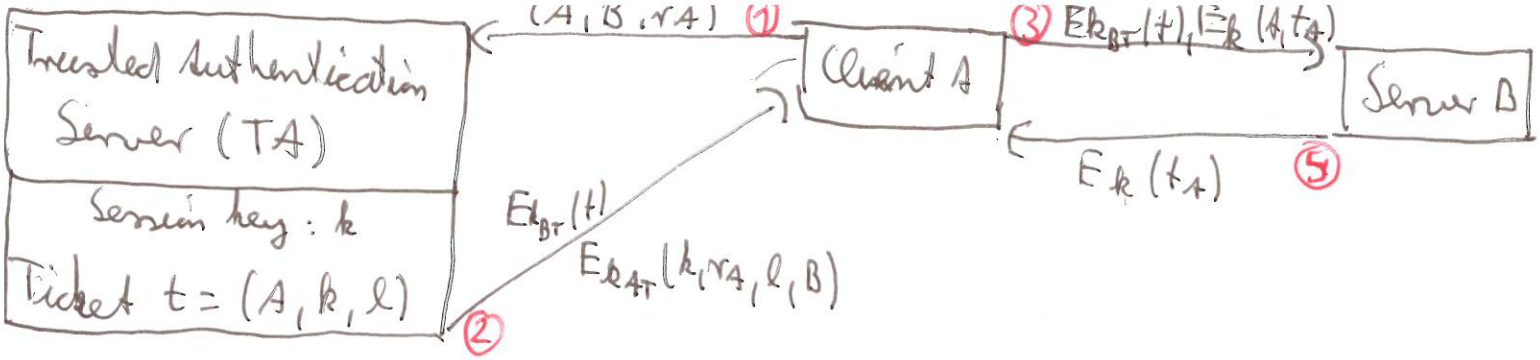
Use symmetric encryption and relies on a trusted authority (TA)

TA: Central server as trusted authority, Kerberos authentication server
It knows the secret key of each client and server.

Notation: E_k: Encryption with key k
r_A: random number by A
t_A: timestamp by A

Client A requests access to a server B.

Basic actions:



Protocol actions (simplified)

1. $A \rightarrow TA : (A, B, r_A)$
2. TA generates a session key k, validity period l, Ticket $t = (A, k, l)$
   $TA \rightarrow A : E_{k_{AT}}(k, r_A, l, B), E_{k_{BT}}(t)$
3. A recovers k, r_A, l, B verifies r_A, B, with t_A : current time
   $A \rightarrow B : E_{k_{BT}}(t), E_k(A, t_A)$
4. B recovers $t = (A, k, l)$ A, t_A and checks
   a) A from t matches A
   b) t_A is fresh
   c) t_A is in the validity period l

   If all checks pass, A's authentication is accepted.

Additionally, to authenticate B to A

5. $B \rightarrow A : E_k(t_A)$
6. A recovers t_A, checks if t_A is correct. If yes, B is authenticated

Session key k is used for encrypting comm. between A and B

Remarks : • r_A in 1 allows authentication of the TA to A.
• t_A in 3 prevents replay attacks of $E_k(t), E_{k_{BT}}(t)$
• Secure and synchronized clocks are needed
• The full version of Kerberos includes another server, the ticket granting server (TGS).