Segurtasuna

Sistema Banatuak

Mikel Larrea, KAT Saila

Segurtasuna

- Sarrera
- Mehatxuak eta erasoak
- Segurtasun politikak
- Segurtasun mekanismoak
- Autentifikaziorako protokoloak
  - Adibideak: Needham-Schroeder, Kerberos
- Konputazio banatua segurua
  - *Secure Multiparty Computation*

Sarrera

- Segurtasun politikak:
  - baliabideak konpartitzerakoan ezarri nahi diren muga zehatzak definitzen dituzte
    - adibidea: nor sar daiteke gela baten
  - teknologiarekiko independenteak izaten dira
- Segurtasun mekanismoak:
  - segurtasun politikak nola inplementatzen diren definitzen dute
    - adibidea: sarraila + giltzak, edota zaindaria atean
  - teknologiari lotutako teknikak izaten dira

Mehatxuak eta erasoak

- Mehatxu motak:
  - informazioaren ihesa: baimendu gabeko informazioaren atzipena
  - baimendu gabeko informazioaren aldaketa
  - informazioaren suntsiketa/hondamena
- Erasoak:
  - mezuak entzutea
  - identitatea ordezkatzea (bezeroa edota zerbitzaria)
  - mezuak aldatzea
  - mezuak atzeratzea (beranduago bidaliz)
  - zerbitzua ukatzea (mezuz gainezkatuz)

Segurtasun politikak

- Konfidentzialtasuna (baimendutako irakurketak) eta integritatea (baimendutako aldaketak):
  - interfazeen babesa
  - sareen babesa
  - hardware elementuen babesa
- Autentifikazioa/egiaztatzea/kautotzea:
  - baimenak eta atzipen-eskubideak ahalik eta gehien murriztu
  - atzipen-gakoen bizitza (iraupena) eta eremua mugatu

Segurtasun mekanismoak

- Atzipenen-kontrola:
  - atzipen-kontrolerako zerrendak
- Zifratzea/enkriptatzea:
  - simetrikoa (adibidez, DES): gako pribatua
    - segurua, azkarra
    - gakoa banatzeko kanal segurua behar du
  - asimetrikoa (adibidez, RSA): bi gako, bat publikoa eta bestea pribatua
    - gako publikoa banatzeko ez du kanal segururik behar
- Autentifikaziorako protokoloak:
  - Needham-Schroeder
  - Kerberos

Needham-Schroeder

Header	*Message*	*Notes*
1. A->S:	A, B, N_A	A requests S to supply a key for communication with B.
2. S->A:	$\{N_A, B, K_{AB}, \{K_{AB}, A\}_{K_B}\}_{K_A}$	S returns a message encrypted in A's secret key, containing a newly generated key K_{AB} and a 'ticket' encrypted in B's secret key. The nonce N_A demonstrates that the message was sent in response to the preceding one. A believes that S sent the message because only S knows A's secret key.
3. A->B:	$\{K_{AB}, A\}_{K_B}$	A sends the 'ticket' to B.
4. B->A:	$\{N_B\}_{K_{AB}}$	B decrypts the ticket and uses the new key K_{AB} to encrypt another nonce N_B.
5. A->B:	$\{N_B - 1\}_{K_{AB}}$	A demonstrates to B that it was the sender of the previous message by returning an agreed transformation of N_B.

Kerberos

- Authentication protocol developed at MIT in the 1980s
- Source code available from MIT (www.mit.edu)
- Included in the OSF DCE, NFS, AFS-3, Microsoft Windows

Kerberos - Architecture

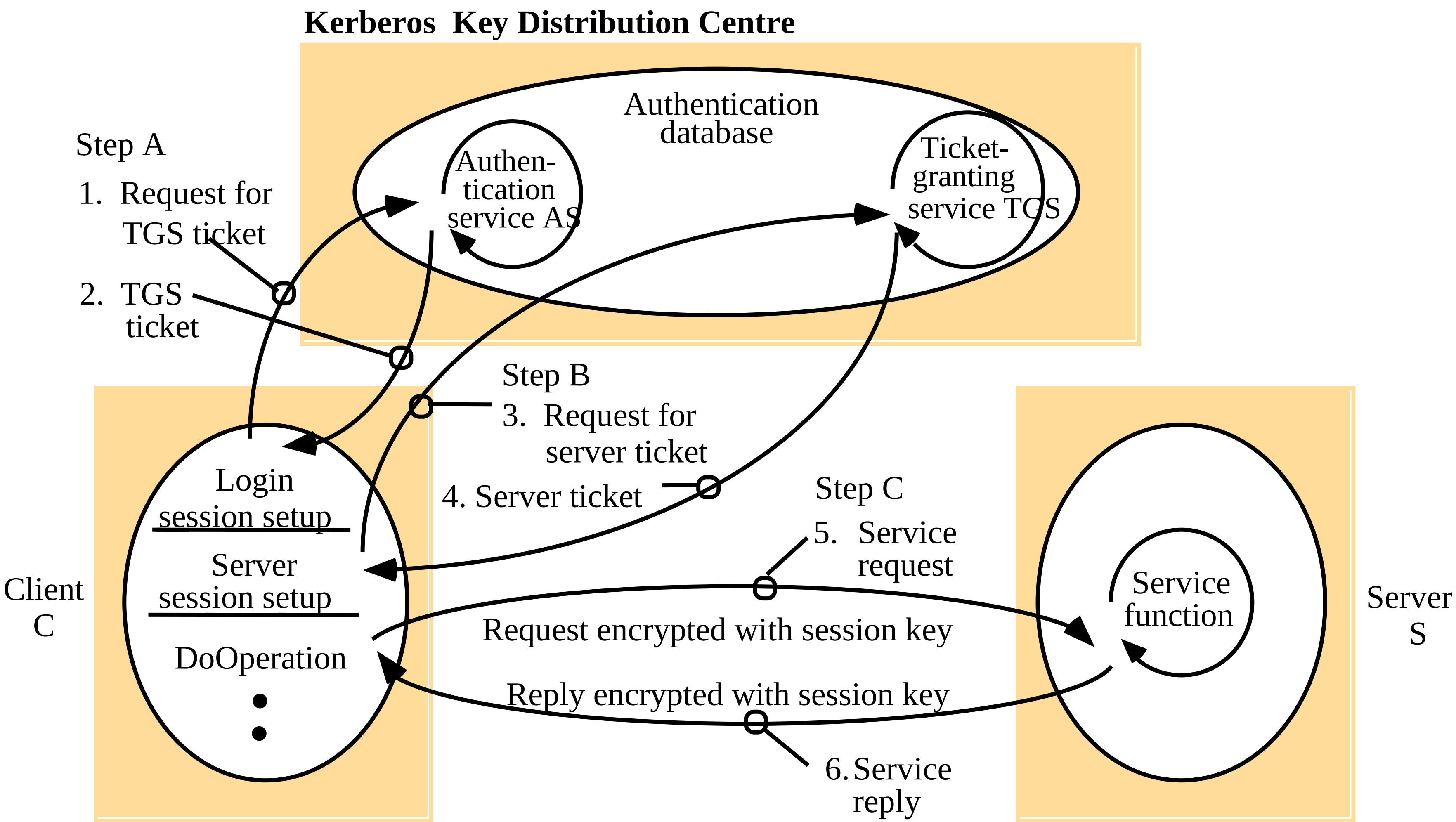
- System architecture:
  - **Kerberos Key Distribution Centre** (**KDC**). It is composed by the following two services:
    - **Authentication Service** (**AS**): authenticates clients on login, and extends tickets to access the Ticket-Granting Service
    - **Ticket-Granting Service** (**TGS**): extends tickets and session keys to clients for accessing particular services

- Kerberos deals with three kinds of security objects: tickets, authenticators, and session keys

Kerberos - Architecture

- **Ticket**: a token issued to a client by the Kerberos ticket-granting service for presentation to a particular server, verifying that the sender has recently been authenticated by Kerberos. Tickets include an expiry time and a session key
- **Authenticator**: a token constructed by a client and sent to a server to prove the identity of the user and the currency of the communication (single use)
- **Session key**: a secret key randomly generated by Kerberos and issued to a client for use when communicating with a particular server

Kerberos - Protocol



Kerberos - Protocol

- Notation
  - $\{M\}_k$: message M encrypted with key K
  - K_C: secret key of client C
  - *ticket*(C, S) = (C, S, t_1, t_2, K_{CS})
    - t_1: begin of validity period for the ticket
    - t_2: end of validity period for the ticket
    - K_{CS}: session key between C and S (randomly generated)
  - *authent*(C) = (C, t)
    - t: timestamp
  - n: number to identify every message
  - A: name of AS
  - T: name of TGS
  - K_T: secret key of TGS

Kerberos - Protocol

Step A: Login session setup

Getting a session key and a ticket for TGS

(1) $\mathbf{C} \xrightarrow{C,\ T,\ n} \mathbf{A}$

(2) $\mathbf{C} \xleftarrow{\{K_{CT},\ n\}_{K_C},\ \{\mathit{ticket}(C,\ T)\}_{K_T}} \mathbf{A}$

Client C is able to decrypt the part of message (2) which is encrypted with its secret key K_C

Kerberos - Protocol

Step B: Server session setup

Getting a session key and a ticket for S

(3) **C** $\xrightarrow{\{\textit{authent}(C)\}_{K_{CT}},\ \{\textit{ticket}(C, T)\}_{K_T},\ S,\ n}$ **T**

(4) **C** $\xleftarrow{\{K_{CS},\ n\}_{K_{CT}},\ \{\textit{ticket}(C, S)\}_{K_S}}$ **T**

Kerberos - Protocol

Step C: Accessing the service

Accessing the server S: request/reply

(5) **C** $\xrightarrow{\{authent(C)\}_{K_{CS}},\ \{ticket(C,\ S)\}_{K_S},\ \text{request},\ n}$ **S**

(6) **C** $\xleftarrow{\{n\}_{K_{CS}},\ \text{reply}}$ **S**

If needed, request and reply could be encrypted with K_{CS}. Including the value n in message (6) allows C to ensure the authenticity of S

Secure Multiparty Computation

- The problem:
  - Let's calculate how much we weight…
  - …without knowing each other's weight ☺
- Approaches:
  - Centralized: by means of a *Trusted Third Party* (TTP) entity
  - Distributed: by means of a *Secure Multiparty Computation* (SMC) algorithm

Secure Multiparty Computation

Model

- *n* participants: $P_1, \ldots, P_n$
- Each participant P_i has a private input x_i
- Goal:
  - Compute a function $f(x_1, \ldots, x_n) = (y_1, \ldots, y_n)$, such that P_i obtains y_i but no other information, in particular no x_i
- Even if some participants behave maliciously:
  - Initially, byzantine failure model
  - Using hardware security modules, e.g., smartcards, omission failure model (easier to handle)

Secure Multiparty Computation

Example

- Three participants, P1, P2, P3. Each one has a private value Ai
- Function to compute:
  - f(A1, A2, A3) = A1+A2+A3
- Strategy:
  - Each participant chooses 3 values between 0 and 1000: two of them are randomly chosen, and the third one is such that the sum of the three values modulo 1000 is equal to Ai
    - For example, if Ai=54 ➔ 300, 550, 204

Secure Multiparty Computation

Example

P1	P2	P3
300	700	320
550	180	500
204	197	239
1054	**1077**	**1059**

Secure Multiparty Computation

Example

P1	P2	P3
300	700	320
180	500	550
239	204	197

Every participant distributes two of its three values among the other participants, and keeps the third value

Secure Multiparty Computation

Example

P1	P2	P3
300	700	320
180	500	550
239	204	197
719	**404**	**67**

Every participant sums the two values received plus the value kept, modulo 1000

Secure Multiparty Computation

Example

P1	P2	P3
719	719	719
404	404	404
67	67	67
190	**190**	**190**

Finally, every participant broadcasts the result obtained previously, and repeats the addition for the received three values (modulo 1000)