Kerberos: Un serviciu de autentificare pentru sisteme Open Network

Proiect Final - Sisteme Distribuite

Prioteasa Liviu Florentin Rizescu Iulian Ștefan Țacu Ștefan Darius

Facultatea de Matematică și Informatică, Universitatea din București

Ianuarie 2026

Context și Problemă

- **Context:** Tranziția de la sisteme monolitice (acces *Serial/Local*) la sisteme distribuite (acces prin *Rețea*).

Problema: Mediul "Open Network" este nesigur

- **Sniffing:** Atacatorii pot intercepta pachetele de date care circulă pe cablu/Wi-Fi.
- **Spoofing:** Adresele IP nu sunt o dovadă suficientă a identității (pot fi falsificate).

Întrebare Cheie

Cum demonstrăm identitatea unui utilizator fără a trimite parola în clar prin rețea?

Soluția - Trusted Third Party (TTP)

Concept

Introducerea unei autorități centrale de încredere, denumită **KDC (Key Distribution Center)**.

Roluri și Arhitectură:

- **Elimină redundanța riscantă:** Nu mai este necesar ca fiecare server din rețea să stocheze parolele utilizatorilor.
- **Fortificare:** Centralizează administrarea securității într-un singur punct critic, care poate fi protejat intens.

Element Cheie

Utilizatorul se autentifică o singură dată (**Single Sign-On**) și primește "bilete" (*Tichete*) pe care le folosește ulterior, fără a mai introduce parola.

Algoritmul Serial

Algorithm 1 Autentificare Serială (Locală)

```
1:  procedure SerialAuth(UserID, InputPassword)
2:      record ← DB.lookup(primary_name = UserID)
3:      if record = NULL then return false          ▷ Utilizator inexistent
4:      end if
5:      DerivedKey ← KDF(InputPassword)
6:      if DerivedKey = record.key_bytes then
7:          CreateLocalSession(UserID) return true
8:      elsereturn false                            ▷ Parolă incorectă
9:      end if
10: end procedure
```

Algoritmul Distribuit (SPMD)

Tranziția la Sistem Distribuit

Spre deosebire de algoritmul serial, varianta **SPMD** (Single Program, Multiple Data) presupune noduri care rulează aceeași logică de protocol, dar operează pe date diferite.

Fluxul celor 3 faze (Mesaje în Rețea):

1. **AS Exchange:** Obținere TGT.
2. **TGS Exchange:** Obținere Tichet de Serviciu.
3. **AP Exchange:** Accesare Resursă.

Faza 1 (SPMD): Autentificarea Inițială (AS Exchange)

- **Scop:** Obținerea "Permisului Universal" (**TGT** - Ticket Granting Ticket).
- **Mecanism:**
  - Utilizator $\rightarrow$ AS: Trimite cererea (ID).
  - AS $\rightarrow$ Utilizator: Răspunde cu TGT-ul criptat (folosind cheia TGS).

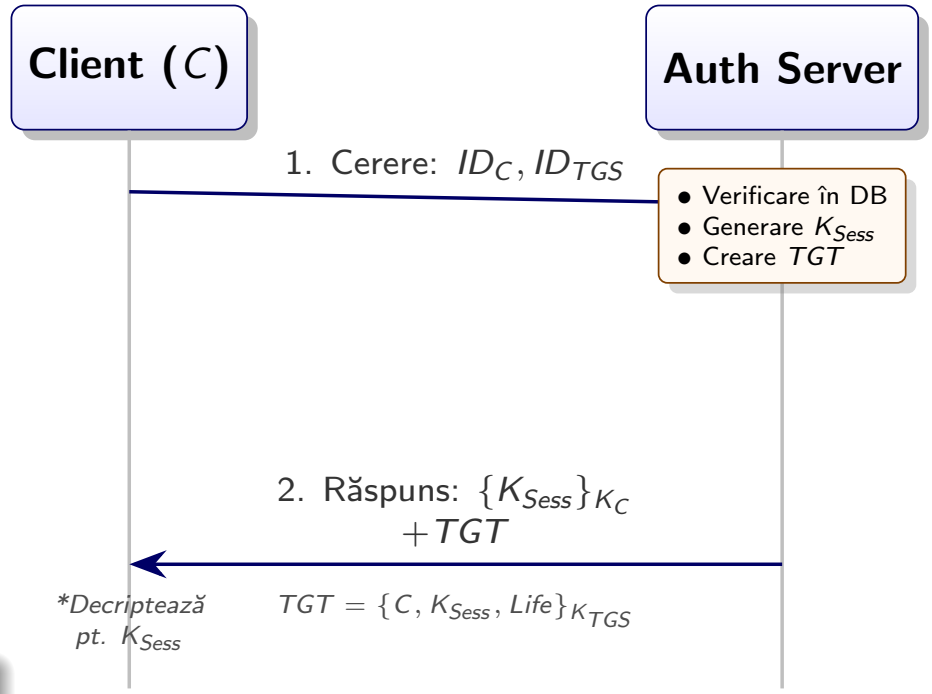

Figura: Fluxul AS Exchange

Detaliu de Securitate

Parola utilizatorului este folosită **doar local** pentru a decripta răspunsul. Ea nu este trimisă niciodată prin rețea.

Faza 2 (SPMD): Obținerea Accesului (TGS Exchange)

- **Scop:** Obținerea tichetului pentru un serviciu specific.
- **Mecanism:** Clientul prezintă TGT-ul + un Authenticator.

Securitate: Replay

Serverul TGS verifică dacă timestamp-ul este recent (< 5 min).
Pachetele vechi sunt respinse imediat.

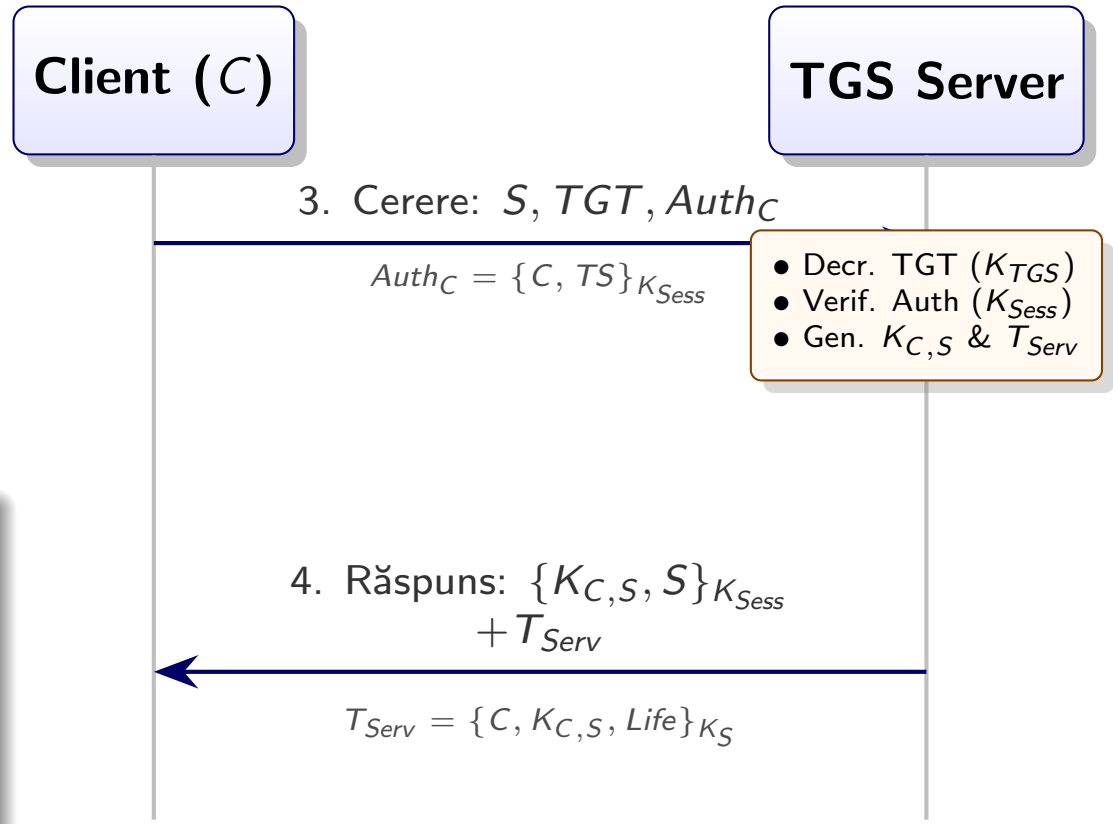

Figura: Fluxul TGS Exchange

Faza 3 (SPMD): Accesarea Serviciului (Client-Server)

- **Scop:** Accesul efectiv la resursa dorită.
- **Mecanism:** Clientul prezintă serverului:
  - **Tichetul de Serviciu** (T_{Serv}).
  - Un nou Authenticator pentru a dovedi identitatea.

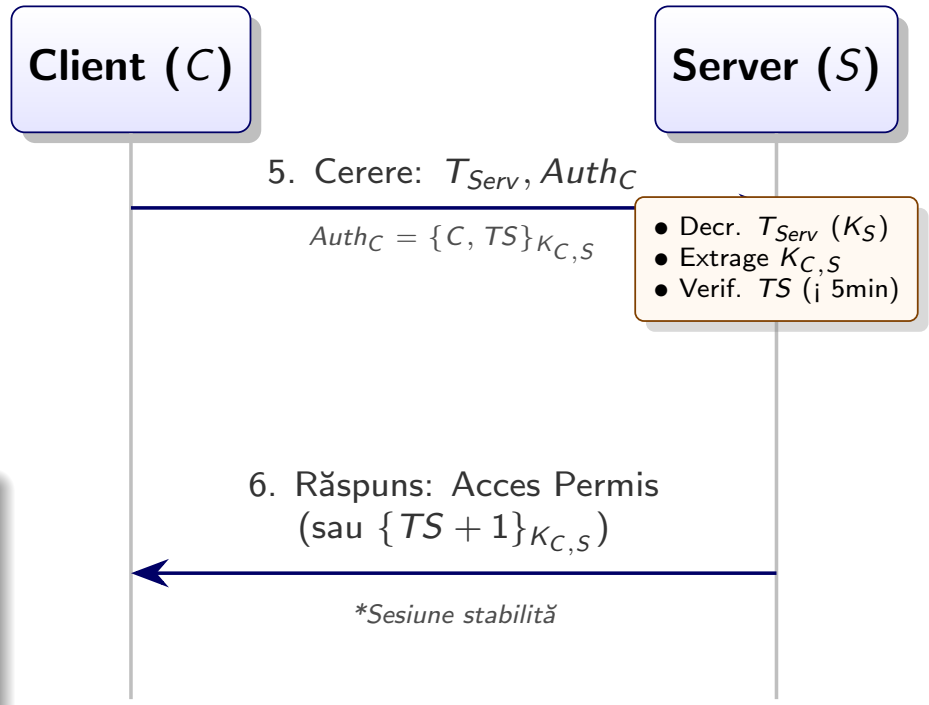

Figura: Fluxul Client-Server

Rezultat: Sistem Stateless

Serverul acceptă cererea și validează tichetul local, **fără a comunica cu KDC-ul**.
KDC-ul nu este interogat la fiecare accesare, asigurând scalabilitatea.

Replicare și Disponibilitate

- **Arhitectura:** Tip **Master-Slave**.
- **Master KDC:**
  - Deține copia autoritativă (Read-Write).
  - Singurul care acceptă scrieri (ex: schimbarea parolei).
- **Slave KDCs:**
  - Dețin copii Read-Only actualizate periodic.
  - Asigură **disponibilitatea** în rețea.

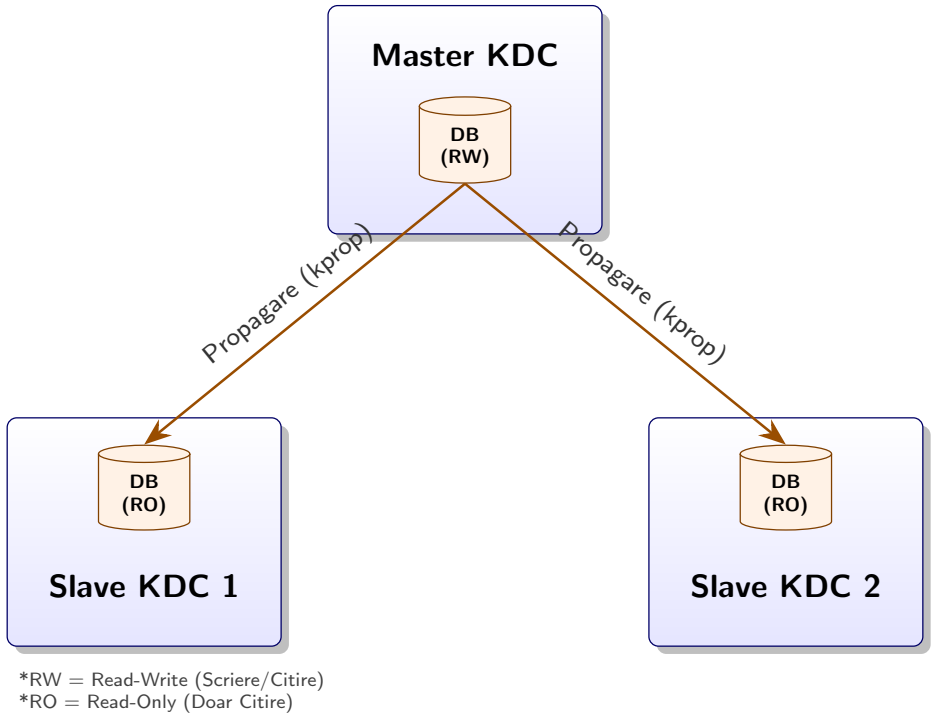

Figura: Arhitectura de Replicare

Analiză Teoretică: Corectitudine și Complexitate

Corectitudine și Dependențe

- **Sincronizarea Ceasurilor (Time Skew):** Deoarece protocolul se bazează pe *Timestamp-uri* pentru a preveni atacurile, este critică sincronizarea prin NTP. O deviație > 5 minute duce la respingerea tichetelor.
- **Integritatea KDC (TTP):** Securitatea este centralizată. Compromiterea cheii Master a KDC-ului permite crearea de *"Golden Tickets"* (impersonare totală).

Performanță și Eficiență

- **Complexitate de Rețea:** $\mathcal{O}(1)$. Protocolul necesită un număr constant de mesaje (**6 pachete**), indiferent de numărul de utilizatori.
- **Eficiență Computațională:** Utilizează exclusiv **criptare simetrică** (AES/DES), care este mult mai rapidă decât cea asimetrică (RSA), reducând încărcarea pe CPU.

Topologii Avansate (1/2): Cross-Realm Authentication

Problemă: Limitarea KDC-ului Unic

Într-o companie globală, gestionarea tuturor utilizatorilor într-o singură bază de date este imposibilă (latenta rețelei, gât de sticlă).

Soluție: Cross-Realm Authentication.

Fluxul de operare:

1. Client (Realm A) cere acces în Realm B.
2. KDC A emite un **"Remote TGT"**.
3. KDC B validează TGT-ul și emite tichetul.

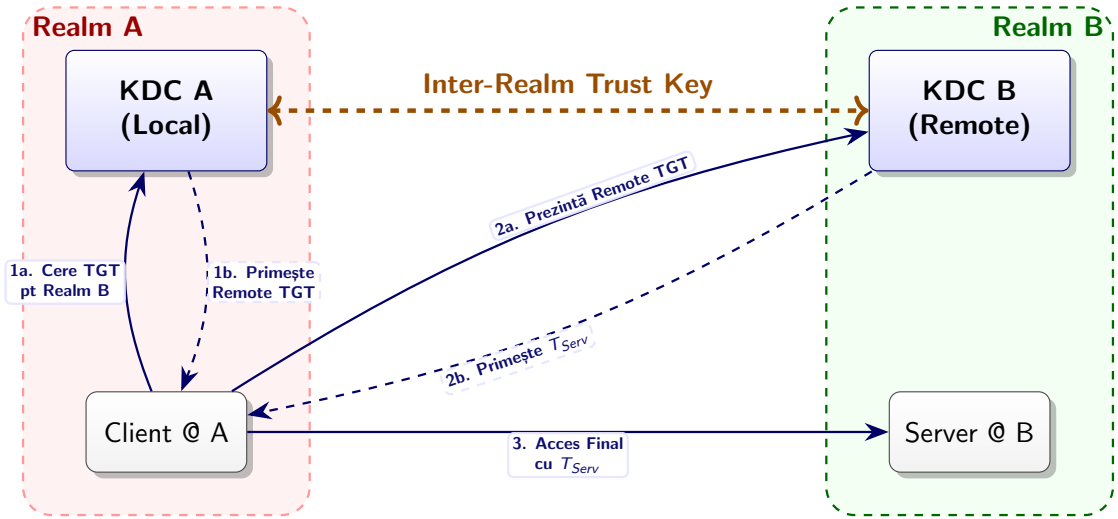

Figura: Flux Cross-Realm

Topologii Avansate (2/2): Delegare (Proxy)

Problemă: Accesul Indirect

Un serviciu intermediar (ex: *Print Server*) trebuie să acceseze o resursă protejată **în numele utilizatorului**, dar nu deține parola acestuia.

Soluție: Proxy Tickets.

Mecanism:

- Utilizatorul emite un tichet special care **deleagă** drepturile sale către Print Server.
- Tichetul conține restricții (adresă IP, timp de viață scurt).
- Serviciul intermediar îl prezintă mai departe ca dovadă.

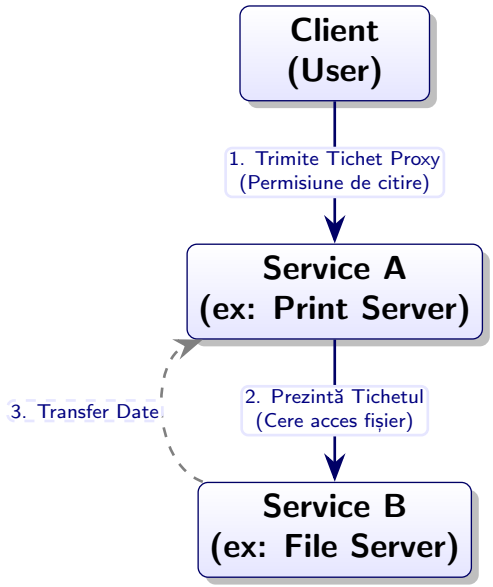

Figura: Topologie Proxy

Analiza Scalabilității (Variația Nodurilor)

Scalabilitate la Citire (Autentificare)

- **Creștere Liniară:** Adăugarea de noduri *Slave* multiplică direct capacitatea de procesare a cererilor de logare.
- **Performanță:** Sistemul poate susține mii de cereri simultane prin balansarea încărcării între Slave-uri.

Limitare la Scriere (Modificări)

- **Master Exclusiv:** Doar KDC-ul Master are drept de scriere în baza de date.
- **SPOF:** Master-ul rămâne un *Single Point of Failure* pentru operațiuni administrative (ex: schimbarea parolei).

Fenomen: Propagation Lag

Există o întârziere inerentă între momentul scrierii pe Master și actualizarea tuturor Slave-urilor.
Consecință: Un utilizator ar putea folosi vechea parolă pe un Slave neactualizat timp de câteva secunde/minute.

Demo Live și Concluzii

Arhitectura Implementării (Sistem Distribuit SPMD)

Sistem format din noduri care respectă protocolul, comunicând prin HTTP:

1. **KDC Server**	2. **Client App**	3. **API Server**
:8080	:3000	:9090
(SQLite DB)	*(Web Frontend)*	*(Resursă Protejată)*

Fluxul de Demonstrație (Step-by-Step):

1. **Login (AS Exchange):** UI → KDC. Obținere TGT (parola rămâne locală).
2. **Get Service Ticket (TGS Exchange):** Folosire TGT → Obținere tichet pentru :9090.
3. **Access Resource (AP Exchange):** Apel către API Server cu tichetul valid.

Insight Cheie

API Server NU contactează KDC!
Verificarea tichetului se face **offline** (folosind cheia secretă partajată) și Replay Cache.

Concluzie: Kerberos permite autentificare sigură pe rețele nesigure, fără a expune secretele.

Topologie și Workflow Complet