Authentication Protocols: Needham-Schroeder & Kerberos

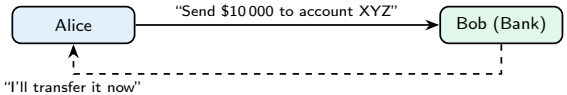
Luca Campa

University of Klagenfurt – Introduction to Cybersecurity

Outline

Why Authentication Protocols?

Example: Securing an e-banking application.



- How does Bob know the message originated from Alice?
- How does Bob know Alice *just* said it (freshness)?

Security objectives:

- **Entity authentication**: binding a message to its originator.
- **Key establishment**: agreeing on a shared session key.
- **Timeliness**: messages are "fresh", not replayed.
- **Secrecy**: generated keys remain confidential.

Solutions: IPSec, SSH, PGP, SSL/TLS, **Kerberos**, ...
We focus on the **underlying ideas**.

Security Protocol: Definition

Protocol

A **protocol** consists of a set of rules (conventions) that determine the exchange of messages between two or more *principals*.

In short: a **distributed algorithm** with emphasis on communication.

Security (Cryptographic) Protocol

A protocol that uses **cryptographic mechanisms** to achieve security objectives: entity/message authentication, key establishment, integrity, timeliness, fair exchange, non-repudiation, ...

Message Notation

Notation	Meaning
A, B, S	Principal names (Alice, Bob, Server)
K, K^{-1}	Key and its inverse
$\{M\}_K$	Encryption of M with key K
$\{M\}_{K_A}$	Encryption with A's **public** key
$\{M\}_{K_A^{-1}}$	Signing with A's **private** key
$\{M\}_{K_{AB}}$	Encryption with symmetric key shared by A, B
N_A, N_B	**Nonces**: fresh random values (challenge/response)
T, T_1	**Timestamps**: denote time, used for expiration
M_1, M_2 or $M_1 \| M_2$	Message concatenation

Example: $\{A, T_A, K_{AB}\}_{K_B}$ – encrypt the triple (A, T_A, K_{AB}) under B's key.

Freshness: Nonces vs. Timestamps

Nonces (challenge/response):

- Fresh random values generated per session.
- Verifier sends nonce as a *challenge*; prover must return it (or a function of it).
- **Pro**: no clock synchronisation needed.
- **Cons**: requires extra round-trip.

Timestamps:

- Encode current time in messages.
- Receiver checks that timestamp is "recent enough".
- **Pro**: can reduce messages (no challenge).
- **Cons**: requires synchronised clocks (difficult in practice).

Both mechanisms prevent **replay attacks**.

Assumptions and Security Goals

Assumptions (for honest principals):

- Principals know their own private keys and the public keys of others.
- Principals can generate/check nonces and timestamps, encrypt and decrypt with known keys.
- Honest principals correctly implement the protocol.

Security goals:

- **Authentication**: binding messages to their originator.
- **Timeliness**: messages are recent / fresh.
- **Key secrecy**: established session keys remain confidential.

Key constraint: goals must hold across *all possible interleavings* of concurrent protocol sessions.

Dolev-Yao Attacker Model

Standard (Dolev-Yao) Attacker

The attacker **controls the network** but **cannot break cryptography**.

The attacker is **active**:

1. Can **intercept and read** all messages.
2. Can **decompose** messages into parts.
3. **Cannot** decrypt without the correct inverse key (perfect cryptography assumption).
4. Can **build new messages** with available constructors.
5. Can **send messages at any time** to any principal.
6. Can be **one of the principals** running the protocol.

Strongest possible assumptions about the attacker $\Rightarrow$ correct protocols function in the widest range of environments.

Types of Attack

Replay attack Re-use parts of previous messages in a new session. *Countermeasure*: nonces or timestamps.

Man-in-the-middle Attacker M sits between A and B: $A \leftrightarrow M \leftrightarrow B$. Each party thinks they communicate with the other directly.

Reflection attack Send transmitted information back to its originator. Exploits symmetric message structure.

Type-flaw attack Substitute a different type of message field. *Example*: use a name where a nonce is expected.

Needham-Schroeder Public Key (NSPK) Protocol: Specification

Goal: Mutual (entity) authentication using public-key cryptography.

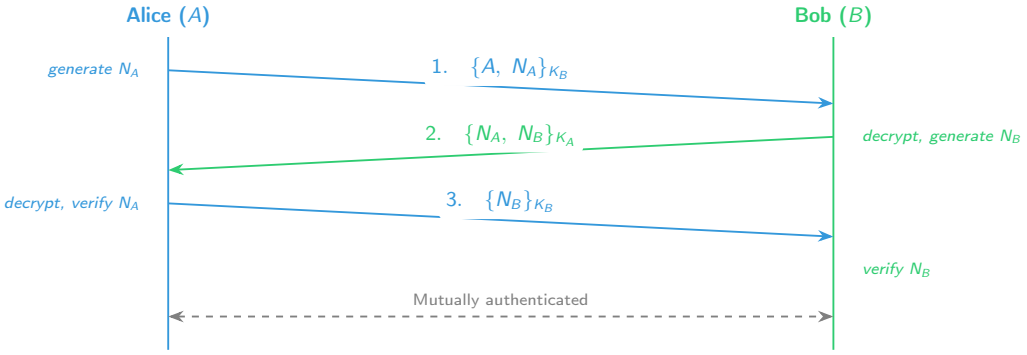
1. $A \rightarrow B : \{A, N_A\}_{K_B}$
2. $B \rightarrow A : \{N_A, N_B\}_{K_A}$
3. $A \rightarrow B : \{N_B\}_{K_B}$

Informal correctness argument:

1. "This is Alice and I have chosen nonce N_A."
2. "Here is your nonce N_A. Since I could decrypt it, I must be Bob. I also challenge you with N_B."
3. "You sent me N_B. Since only Alice can decrypt it and I returned it, I must be Alice."

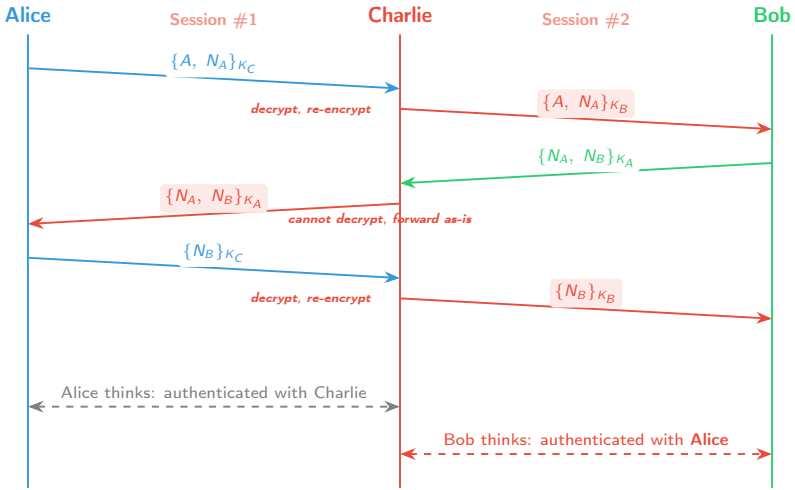
Protocol proposed by Needham & Schroeder (1978) and used for **decades**...

Needham-Schroeder Public Key (NSPK) Protocol: Message Exchange



Attack on NSPK: Lowe's Man-in-the-Middle (1995)

Key insight: Charlie is a legitimate principal who Alice initiates a session with. Charlie relays the challenge to Bob.



Why the Attack Works

The problem: Message 2 does not bind the responder's identity:

$$2.\ B \to A : \{N_A, N_B\}_{K_A}$$

Bob's name does **not** appear inside the encryption. Alice cannot tell whether the nonce challenge came from Bob or was relayed by Charlie.

Consequence: Bob authenticates "Alice", but Alice never intended to talk to Bob. Charlie impersonates Alice to Bob.

Timeline:

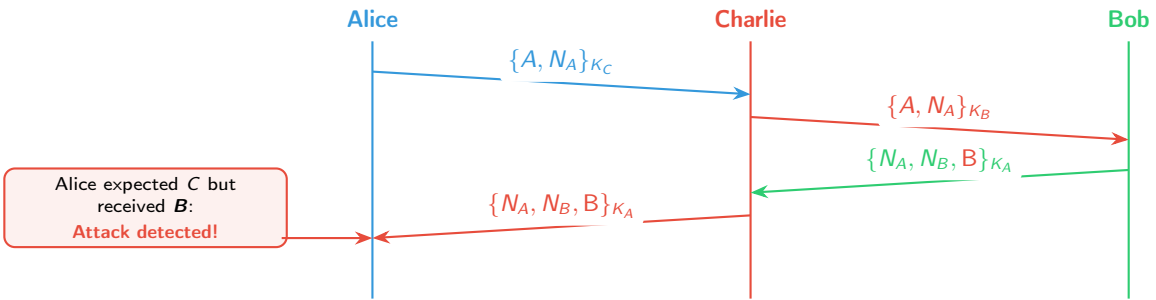
1. Alice $\to$ Charlie: $\{A, N_A\}_{K_C}$
2. Charlie $\to$ Bob: $\{A, N_A\}_{K_B}$
3. Bob $\to$ Charlie: $\{N_A, N_B\}_{K_A}$
4. Charlie $\to$ Alice: $\{N_A, N_B\}_{K_A}$
5. Alice $\to$ Charlie: $\{N_B\}_{K_C}$
6. Charlie $\to$ Bob: $\{N_B\}_{K_B}$

Discovered by Gavin Lowe in 1995, 17 years after the protocol was published.

Lowe's Fix

Fix: Include the responder's identity B inside Message 2:

1. $A \to B : \{A, N_A\}_{K_B}$
2. $B \to A : \{N_A, N_B, \boldsymbol{B}\}_{K_A}$
3. $A \to B : \{N_B\}_{K_B}$



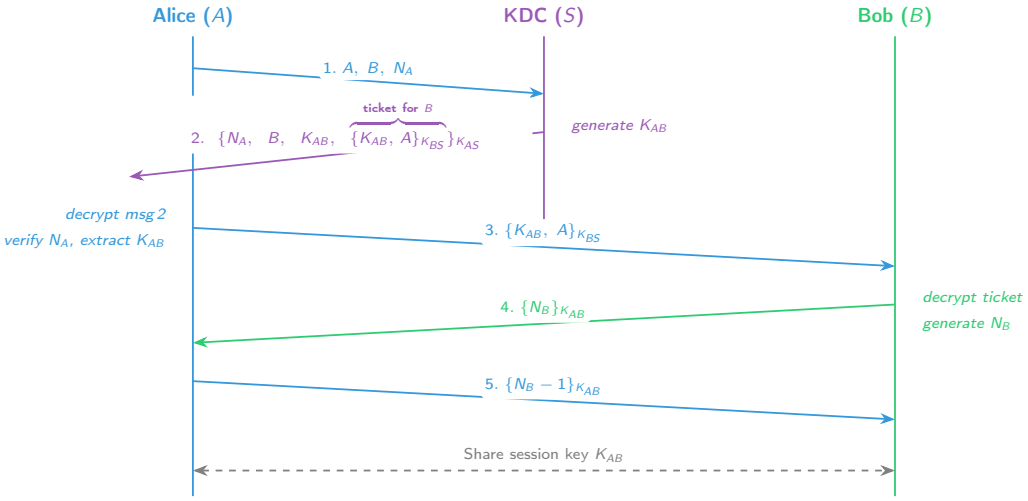
Needham-Schroeder Shared-Key (NSSK) Protocol: Specification

Goal: Authenticated key exchange via a trusted **Key Distribution Centre (KDC)**.
Uses **symmetric-key** cryptography only.

1. $A \to S$: $A,\ B,\ N_A$
2. $S \to A$: $\{N_A,\ B,\ K_{AB},\ \{K_{AB},\ A\}_{K_{BS}}\}_{K_{AS}}$
3. $A \to B$: $\{K_{AB},\ A\}_{K_{BS}}$
4. $B \to A$: $\{N_B\}_{K_{AB}}$
5. $A \to B$: $\{N_B - 1\}_{K_{AB}}$

- K_{AS}: long-term key shared between A and server S (KDC).
- K_{BS}: long-term key shared between B and server S.
- K_{AB}: **fresh session key** generated by S.
- The "ticket" $\{K_{AB}, A\}_{K_{BS}}$ is opaque to A.

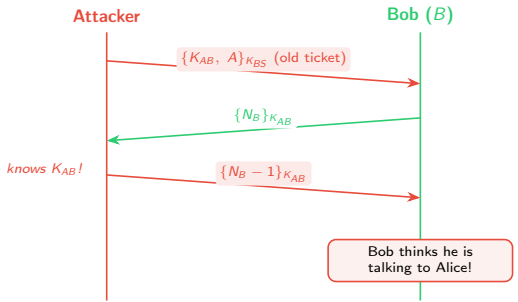
NSSK Protocol: Message Exchange



NSSK Protocol: Replay Attack Weakness

Problem: If a previous session key K_{AB} gets **compromised**, an attacker can:

1. Replay the old ticket $\{K_{AB}, A\}_{K_{BS}}$ to B.
2. B accepts it (cannot tell it is old).
3. Attacker knows K_{AB} and can complete the challenge–response (msgs 4–5).
4. B believes it is talking to A



Root cause: B has no way to check the *freshness* of the ticket (no timestamp), and B did not use a nonce *before* the ticket was created.

Fix: Include a **timestamp** in the ticket $\Rightarrow$ leads to **Kerberos**.

NSSK → Kerberos: Key Changes

Kerberos is loosely based on NSSK with two critical improvements:

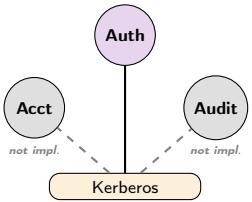
1. **Timestamps** instead of nonces to assure freshness of session keys.
2. **Removal of nested encryption** (simplified message structure).

NSSK	**Kerberos (simplified)**
1. A, B, N_A	1. A, B
2. $\{N_A, B, K_{AB}, \{K_{AB}, A\}_{K_{BS}}\}_{K_{AS}}$	2. $\{B, K_{AB}, T_1\}_{K_{AS}}$, $\{K_{AB}, A, T_1\}_{K_{BS}}$
3. $\{K_{AB}, A\}_{K_{BS}}$	3. $\{K_{AB}, A, T_1\}_{K_{BS}}$, $\{A, T_2\}_{K_{AB}}$
4. $\{N_B\}_{K_{AB}}$	4. $\{T_2 - 1\}_{K_{AB}}$
5. $\{N_B - 1\}_{K_{AB}}$	

- Kerberos uses **4 messages** (vs. 5 for NSSK).
- Timestamp T_1 in the ticket prevents replay of old tickets.

Kerberos: Background

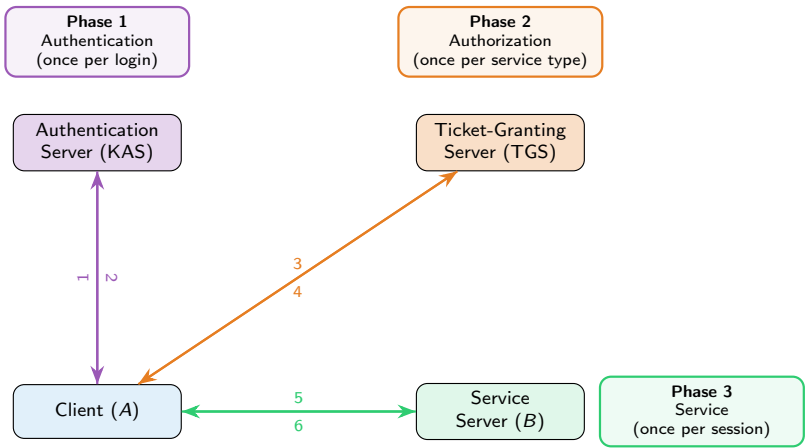
- Protocol for **authentication and access control** in client/server applications.
- Originally 3 components: authentication, accounting, and audit: only authentication was implemented.
- Developed as part of **Project Athena** (MIT, 1980s).
- Version V (1993, RFC 4120) is the current standard.
- **Widely deployed**: Microsoft Windows, Active Directory, Linux (SSSD / PAM), Hadoop, ...



Kerberos: Design Goals

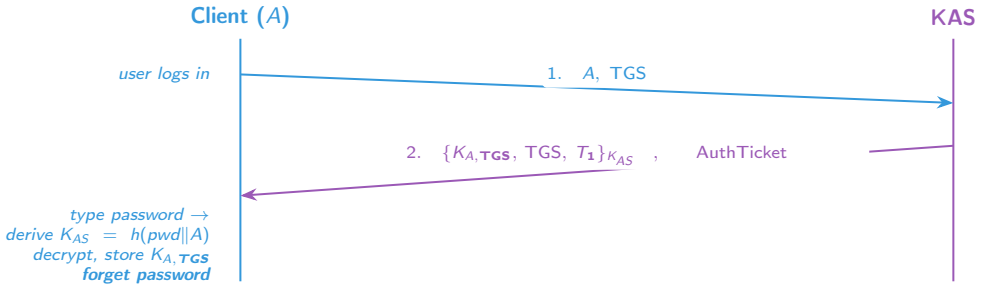
1. The user's password must **never travel** over the network.
2. The user's password must **never be stored** on the client machine: discarded immediately after use.
3. The password should **never be stored in plaintext**, even in the authentication server database.
4. **Single Sign-On (SSO)**: the user enters the password **once** per work session and can access all authorised services transparently.
5. Authentication information resides **only on the authentication server**:
   - Admin disables a user in one place.
   - Password change applies to all services at once.
   - No redundancy of credentials to safeguard.
6. **Mutual authentication**: servers also prove their identity to clients.
7. After authentication, client and server establish an **encrypted channel**.

Kerberos: Architecture Overview



Phase 1: Authentication (Login)

1. $A \rightarrow \text{KAS}: A, \text{TGS}$
2. $\text{KAS} \rightarrow A: \{K_{A,\text{TGS}}, \text{TGS}, T_1\}_{K_{AS}}, \underbrace{\{A, \text{TGS}, K_{A,\text{TGS}}, T_1\}_{K_{\mathbf{KAS,TGS}}}}_{\text{AuthTicket}}$

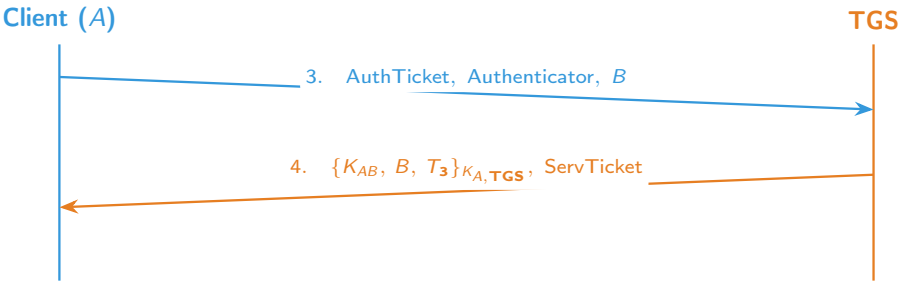


- $K_{A,\text{TGS}}$ has a lifetime of several hours.
- A is logged out when $K_{A,\text{TGS}}$ expires.

Phase 2: Authorization (Get Service Ticket)

3. $A \to \text{TGS}$: $\underbrace{\{A,\ \text{TGS},\ K_{A,\text{TGS}},\ T_1\}_{K_{\mathbf{KAS,TGS}}}}_{\text{AuthTicket}},\ \underbrace{\{A,\ T_2\}_{K_{A,\mathbf{TGS}}}}_{\text{Authenticator}},\ B$

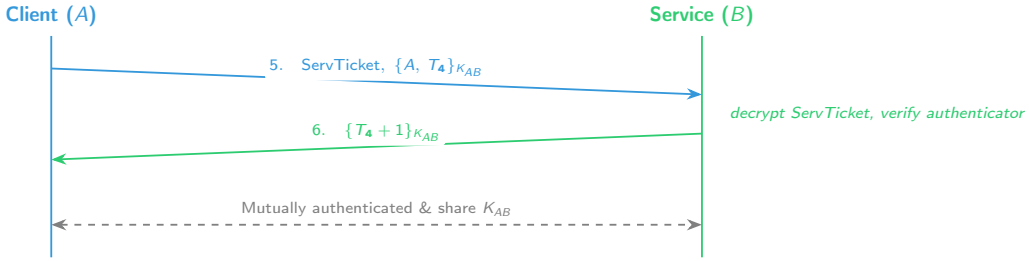
4. $\text{TGS} \to A$: $\{K_{AB},\ B,\ T_3\}_{K_{A,\mathbf{TGS}}},\ \underbrace{\{A,\ B,\ K_{AB},\ T_3\}_{K_{BS}}}_{\text{ServTicket}}$



- **Authenticator**: short-lived (seconds), prevents replay. Servers cache recent authenticators to detect immediate replays.
- TGS issues session key K_{AB} (lifetime: minutes) and ServTicket (opaque to A).

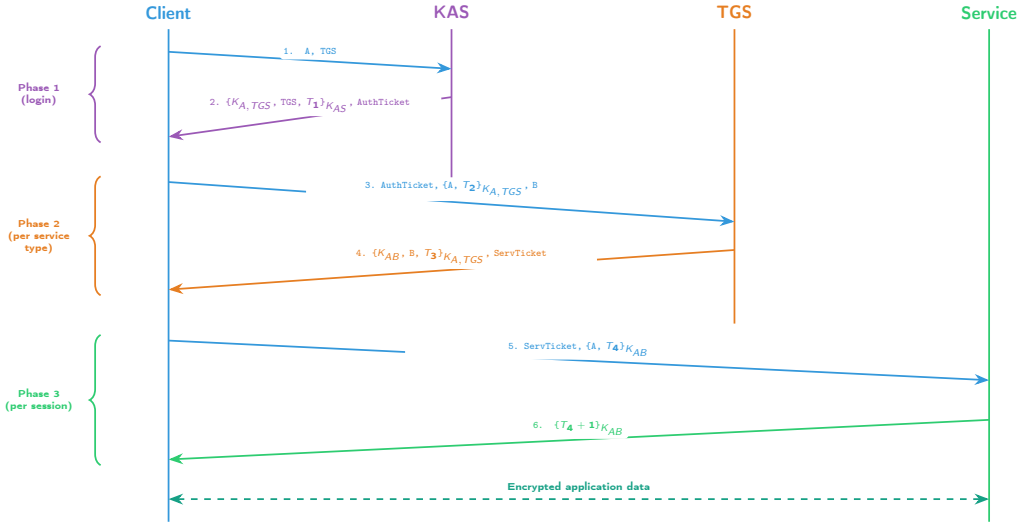
Phase 3: Service Access

5. $A \to B$: $\underbrace{\{A,\ B,\ K_{AB},\ T_3\}_{K_{BS}}}_{\text{ServTicket}},\ \underbrace{\{A,\ T_4\}_{K_{AB}}}_{\text{Authenticator}}$
6. $B \to A$: $\{T_4 + 1\}_{K_{AB}}$

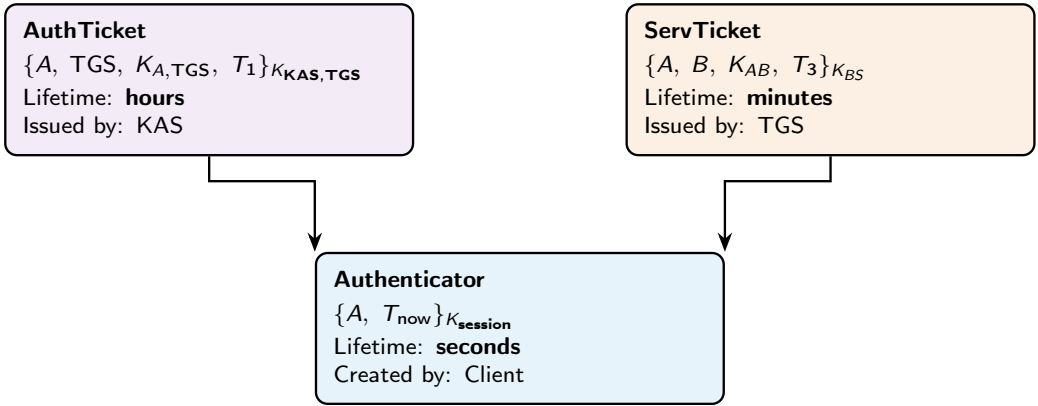


- Message 6 provides **mutual authentication**: B proves it could decrypt the ticket (and thus knows K_{BS}).
- K_{AB} is then used to encrypt subsequent application traffic.

Kerberos: Complete Message Flow



Kerberos: Role of Tickets and Authenticators



*Each request pairs a **ticket** (opaque to client) with a fresh **authenticator** to prevent replay.*

Kerberos: Security Analysis

Strengths:

- Password never leaves the client (derived key used once, then discarded).
- Tickets have bounded lifetimes via timestamps.
- Authenticators prevent ticket replay.
- Single Sign-On reduces password exposure.
- Mutual authentication (message 6) prevents server impersonation.

Limitations:

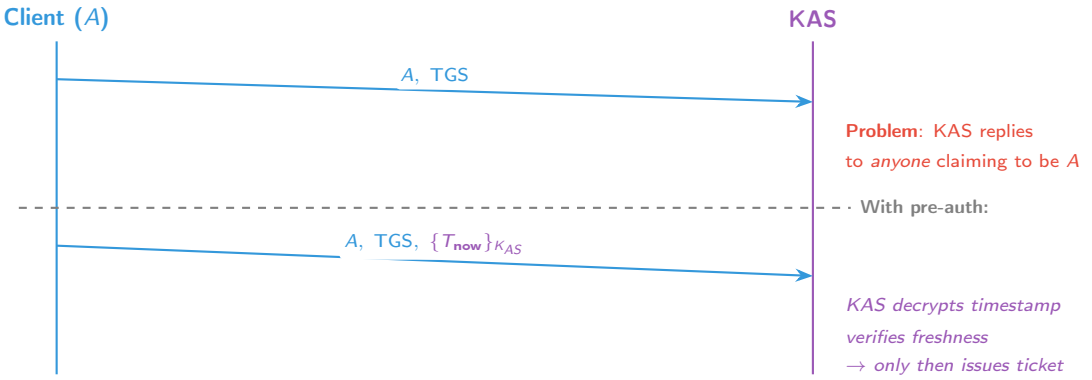
- Requires **synchronised clocks** (typically < 5 min tolerance) — Network Time Protocol (NTP) must itself be secured.
- **Single point of failure**
- Password-based: vulnerable to **offline dictionary attacks** on message 2 (mitigated by pre-authentication in Kerberos V);
- Ticket lifetime trade-off: too long $\rightarrow$ replay window; too short $\rightarrow$ user inconvenience.

Kerberos Version IV vs. Version V

	Kerberos IV	**Kerberos V (RFC 4120)**
Encryption	DES only	AES, 3DES, ...
Network protocol	IPv4 only	IPv4 and IPv6
Ticket lifetime	8-bit, $\sim$21 hours max	Start/end timestamps
Inter-domain auth	Limited	Full transitive trust
Pre-authentication	None	Encrypted timestamp
Authorization data	None	Extensible field in tickets

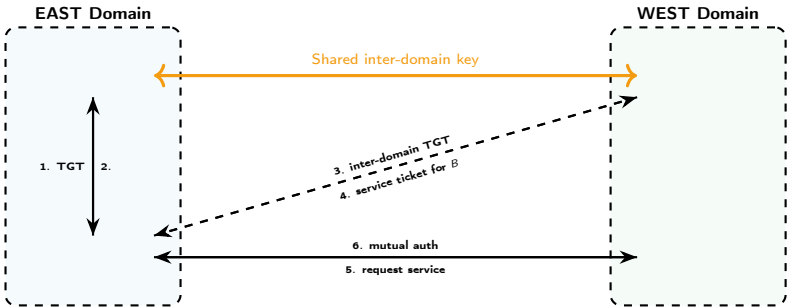
- Kerberos V pre-authentication: client must prove knowledge of K_{AS} **before** KAS issues a ticket, preventing offline dictionary attacks on message 2.
- **Inter-domain**: separate Kerberos servers can share trust via inter-server TGTs (Ticker Granting Tickets).

Kerberos V Pre-Authentication



- Attacker cannot request tickets for arbitrary principals without knowing K_{AS}.
- Prevents **offline dictionary attacks**.

Inter-domain(server) Authentication



- Client first obtains a **Inter-domain TGT** from its local KDC.
- Presents it to the remote KDC to obtain a service ticket.
- Enables enterprise-wide SSO across administrative boundaries.

Kerberos in Practice

Microsoft Active Directory:

- Kerberos V is the default authentication protocol since Windows 2000.
- Domain Controllers = KDC.
- Group Policy / LDAP for authorization.
- Supports constrained delegation, protocol transition.

Linux / UNIX:

- MIT Kerberos, Heimdal implementations.
- SSSD / PAM integration.
- NFS, SSH, LDAP can use Kerberos via GSSAPI.

Modern attacks to be aware of:

- **Golden Ticket**: attacker obtains the TGS long-term key $\rightarrow$ can forge *any* TGT.
- **Silver Ticket**: attacker obtains a service's key $\rightarrow$ forges service tickets.
- **Kerberoasting**: request service tickets for service accounts, then crack the (weak) service password offline.
- **AS-REP Roasting**: target accounts without pre-authentication.

Mitigations: strong/random service passwords, enforce pre-authentication, monitor anomalous ticket requests.

Summary

Needham-Schroeder Protocols:

- **NSPK**: 3-message mutual authentication using public keys.
- Vulnerable to **man-in-the-middle** (Lowe, 1995).
- **Fix**: include responder identity in message 2.
- **NSSK**: 5-message authenticated key exchange via KDC.
- Vulnerable to **replay** of old tickets.
- **Fix**: timestamps $\rightarrow$ Kerberos.

Kerberos:

- Built on NSSK with timestamps and authenticators.
- 3 phases: authentication, authorization, service.
- **Single Sign-On**: password entered once.
- Tickets + authenticators prevent replay.
- Kerberos V: pluggable crypto, pre-authentication, inter-domain trust.
- Dominant protocol in enterprise environments (Active Directory, Linux/UNIX).

Lesson: "Small recipes, but nontrivial to design" (David Basin - ETH Zurich) – formal verification matters!