Kerberos Deep Dive

Part 1 - Introduction

July 2025, Alex Joss

Content Overview

Part 1 - Kerberos Introduction

Part 2 - Kerberoasting

Part 3 - AS-REP Roasting

Part 4 - Unconstrained Delegation

Part 5 - Constrained Delegation

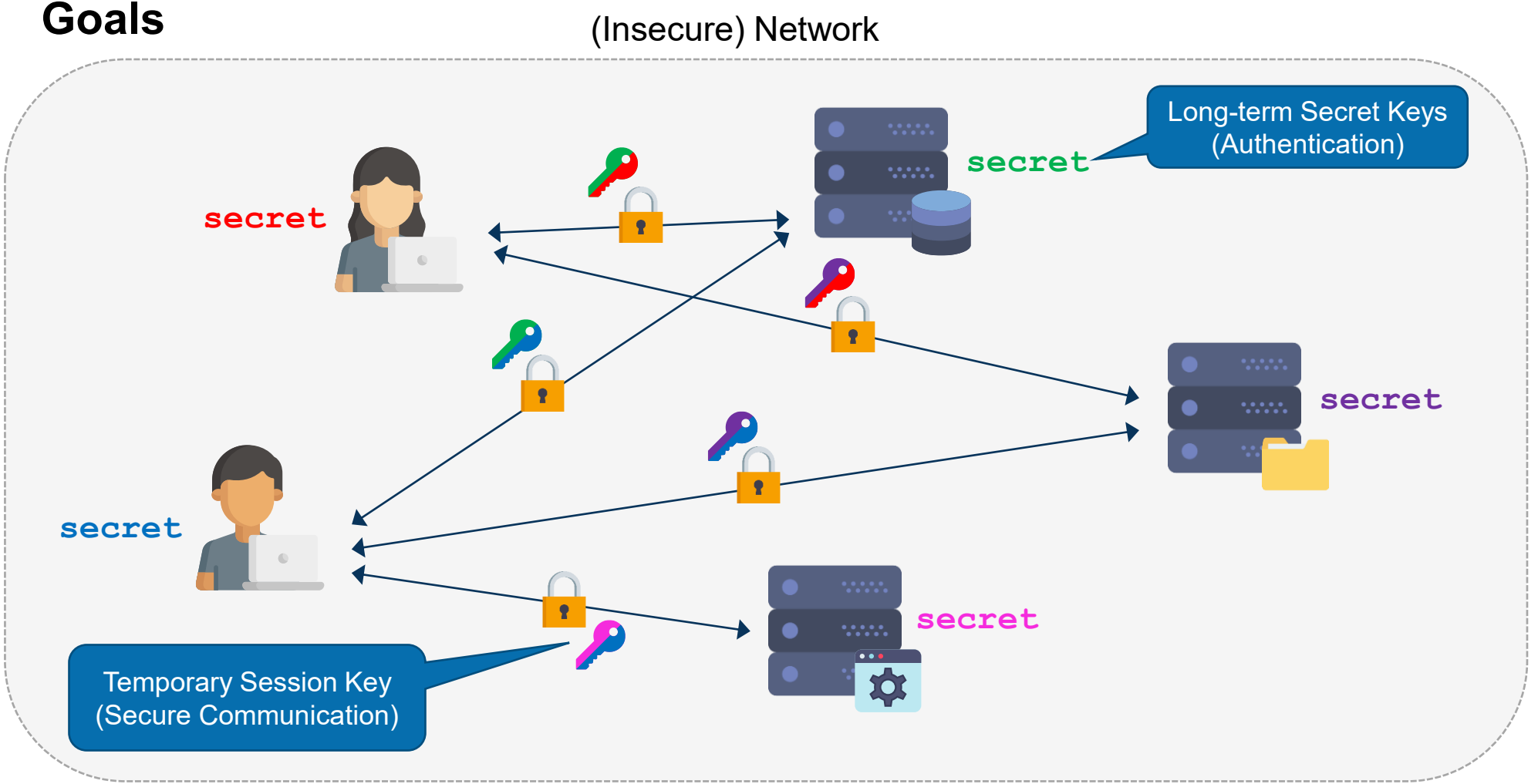
Part 6 - Resource-Based Constrained Delegation

What Is Kerberos?

Kerberos

- Developed by MIT in 1988
- Named after the 3-headed dog Cerberus that guards the gates to the underworld
- Authentication protocol over untrusted networks
- Authentication is performed by a trusted third party (the KDC)
- Involved parties can prove their identities to one another (mutual authentication)
- Based on tickets that are issued to and exchanged between parties
- Uses secret-key (symmetric) cryptographic principles

Goals


(Insecure) Network
secret
secret
Long-term Secret Keys
(Authentication)
secret
secret
secret
Temporary Session Key
(Secure Communication)

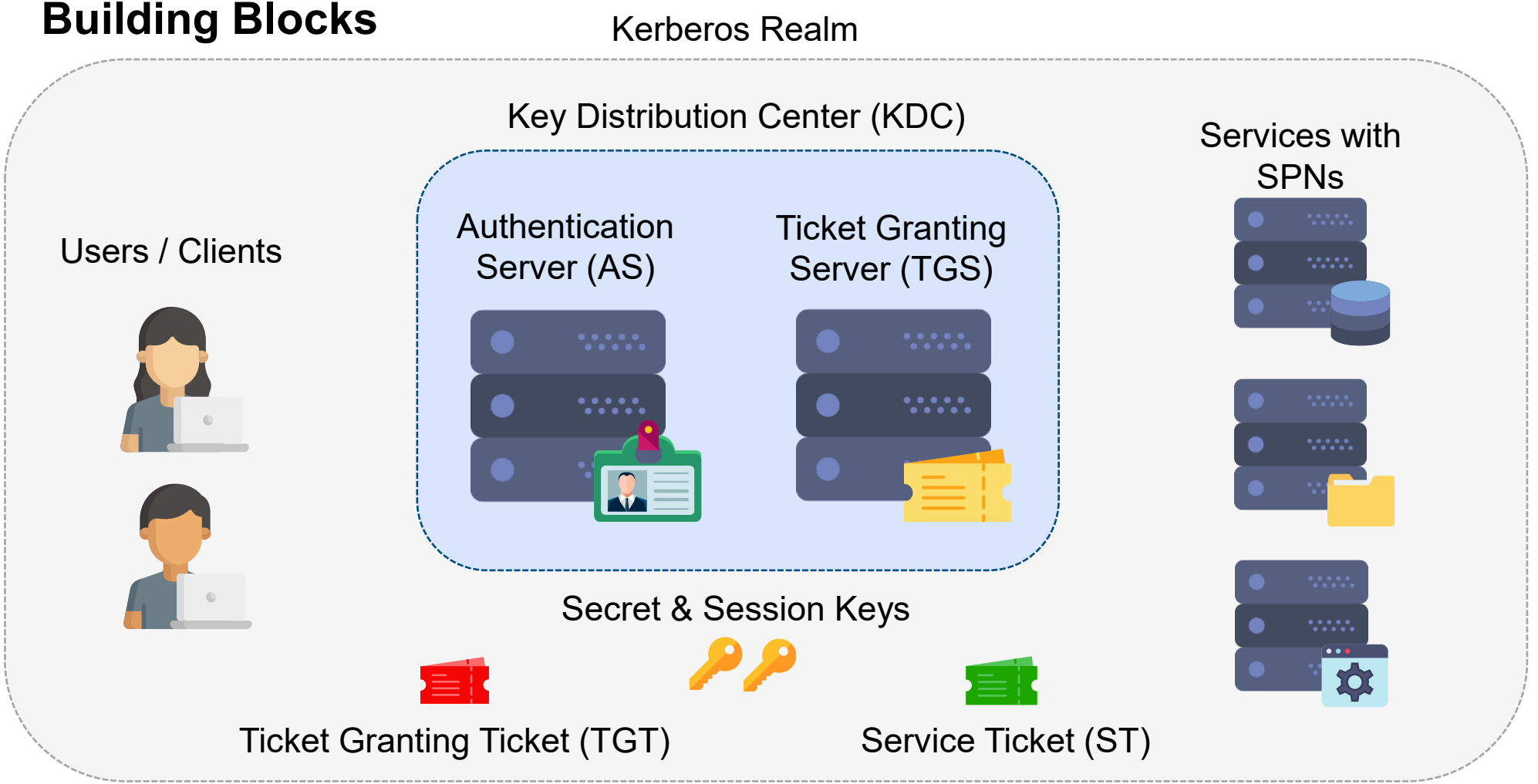
Kerberos in Windows

- Microsoft has its own implementation of the Kerberos Protocol, called MS-KILE*
- Available since Windows 2000
- Default authentication package
- Intended to replace NTLM authentication, however NTLM is still widely used
- Used for SSO purposes (e.g. users can access services with their AD credentials)
- Requires services to have a registered SPN (Service Principal Name)
- Only works with FQDN/NetBIOS names, not with IPs (unless the IP is configured in the SPN**)

* https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-kile/2a32282e-dd48-4ad9-a542-609804b02cc9

** https://docs.microsoft.com/en-us/windows-server/security/kerberos/configuring-kerberos-over-ip

Building Blocks



Details: KDC

- The KDC is the **trusted 3rd party** that mediates between all other parties
- Implicitly configured/installed on **domain controllers**
- Has access to the **Kerberos key material** of all users
- Consists of two logical core services:
  - **Authentication Server (AS)** 🪪
    → Authenticates clients and issues Ticket Granting Tickets (TGT)
  - **Ticket Granting Server (TGS)** 🎟️
    → Accepts authenticated clients and issues Service Tickets (ST) for specific services

Details: Service Principal Names

- An SPN is a unique identifier of a service instance within the domain

- Format: **<service class>**/**<host>**:**<port>**/**<service name>**
  - **<service class>** → Type of service (e.g. www, ldap, MSSQLSvc etc.)
  - **<host>** → Name of the computer the service is running on (FQDN or NetBIOS)
  - **<port>** (optional) → Port of service, to differentiate between multiple service on the same host
  - **<service name>** (optional) → For replicable services
- Example: **MSSQLSvc**/**ws1.winattacklab.local**:**1433**

- Used to associate a service instance with an actual account
- In a Windows domain, some SPNs are created automatically, others need to be registered manually

More information: https://adsecurity.org/?page_id=183

Details: Secret & Session Keys

- Kerberos uses symmetric cryptography to establish trust & secure communication

- Secret keys:
  - **Long-term** shared secrets between users, services and the KDC
  - Serve to authenticate individual parties
  - **Derived** from the respective account's **password** (either actual end user or SPN/service account)
  - Derivation via **`string-to-key`** function (different implementation based on encryption type)

- Session keys:
  - **Short-term** shared secrets between users & individual services
  - Serve to secure communication between two principals for one session
  - **Generated randomly on-the-fly** by the involved parties

Secret Key Example

- When a new domain user is created, their password is defined
- The domain controller then derives the respective secret keys and stores them
- The same happens when you log in with a domain user (on your client)
- Example:

```
NT Hash*: 8cf0345c0d74a3efeb598489493cf47b
AES256:   a8f2e85f4fff9b8399ed6e7855d8b757f6098b433b4c844a5cda458b689e0138
AES128:   0c7c30cbf5d727a436b4848d253ef0c4
DES:      ae326dc416c2bfd3
```

* Used for RC4 encryption

Details: Tickets

- Tickets carry **identity information** about a subject (e.g. username, groups etc.)
- They are intended for a **specific audience** (e.g. a receiving service)
- Certain parts of a ticket are **encrypted**, either with:
  - the recipients long-term secret (e.g. password hash) or
  - a temporary session key established between two parties

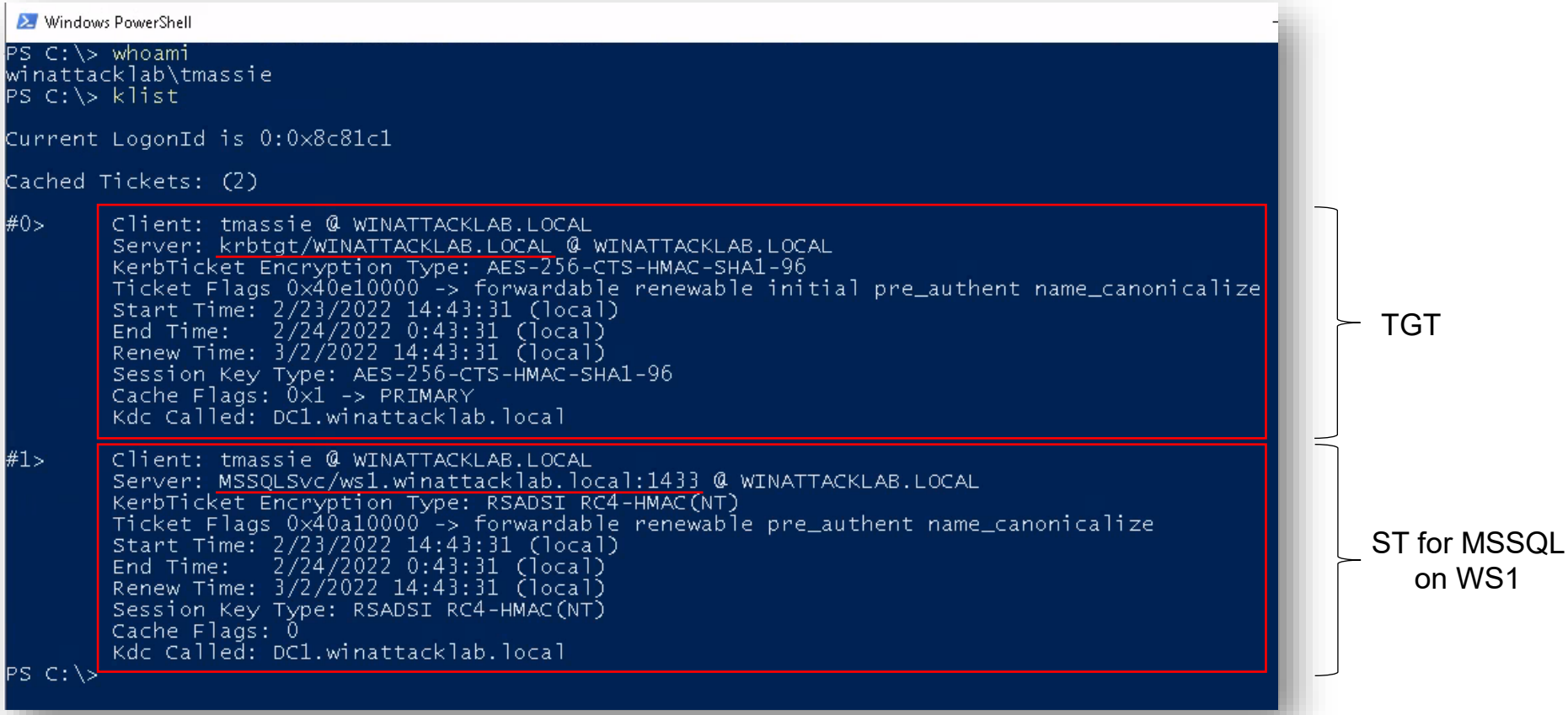
- Ticket types:
  - **Ticket Granting Ticket (TGT)**

    → Confirms the user's identity towards the Ticket Granting Service of the KDC
  - **Service Ticket (ST)**

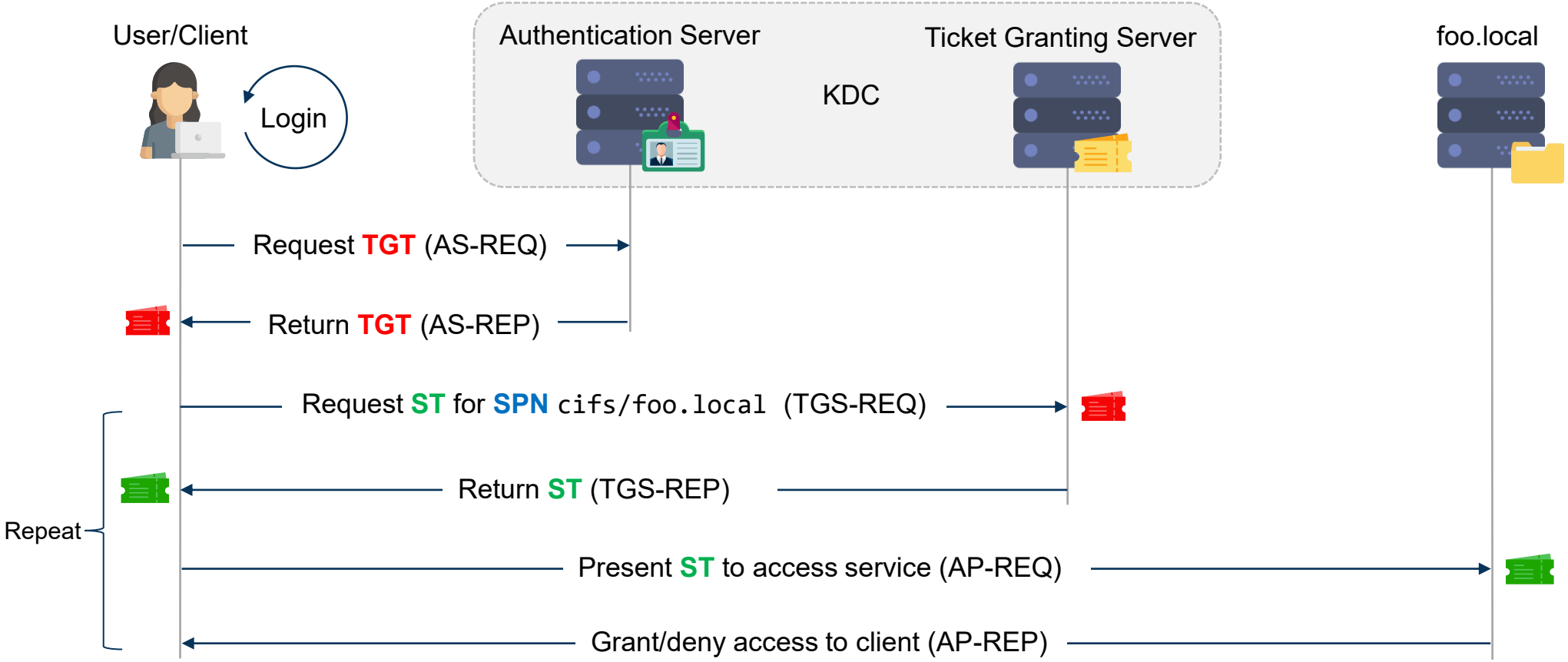
    → Confirms the user's identity towards a specific service (e.g. CIFS/fs1.foo.local)

Details: Tickets Inspection in Windows

You can inspect your current tickets with the `klist` utility:



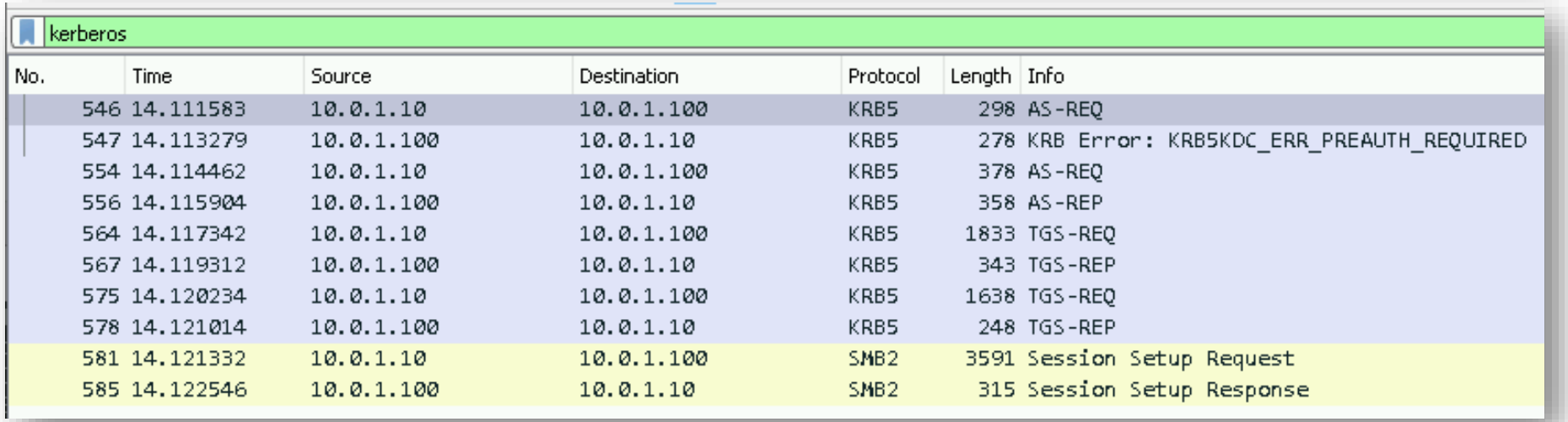
High Level Kerberos Authentication Flow



TGT: Ticket Granting Ticket **ST: Service Ticket** **SPN: Service Principal Name**

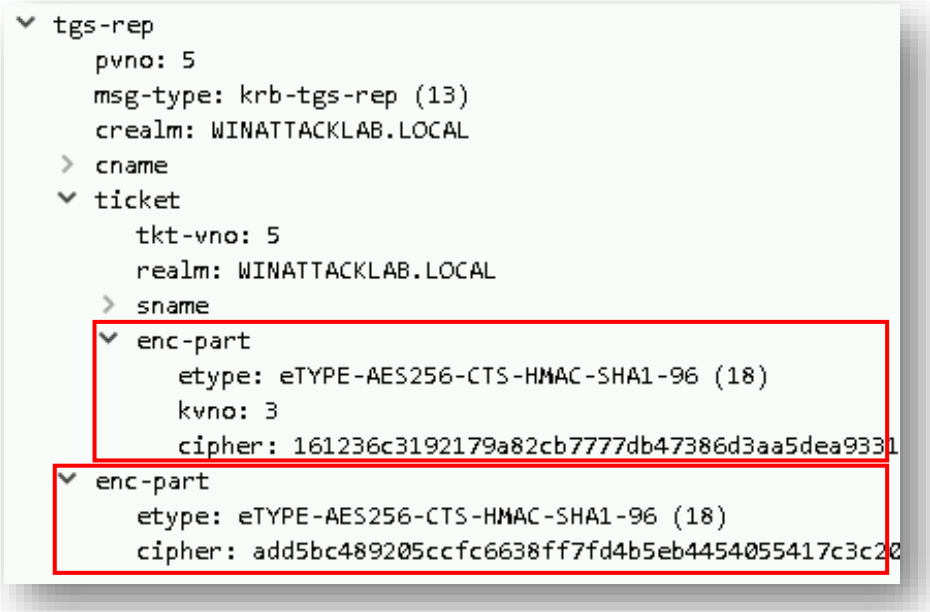
Kerberos & Wireshark

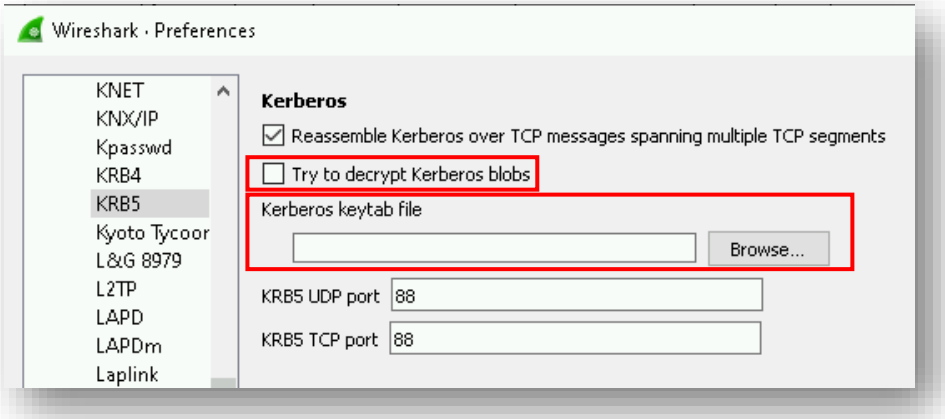
- Wireshark comes with a built-in dissector for Kerberos
- Kerberos messages are reconstructed automatically
- Filtering for `kerberos` shows actual Kerberos protocol traffic as well as usage of Kerberos authentication in other protocols (e.g. SMB)



Kerberos & Wireshark cont.

- Some parts of Kerberos messages are encrypted (with session or long-term keys)





- Wireshark can decrypt those parts, if the correct keytab file is provided in the configuration
- Edit > Preferences > Protocols > KRB5 > Kerberos keytab file

Creating Keytab files

- Use `ktutil` (on Unix) or `ktpass` (on Windows) to create keytab files for specific users
- Requires the password of the respective accounts and some other details
- The keytab file then contains the long-term key of said user(s)
- You can also create a keytab file for all accounts in a domain
- Tools to create domain-wide keytab files:
  - Forest-Trust-Tools by Dirk-Jan Mollema: https://github.com/dirkjanm/forest-trust-tools
  - Samba: https://wiki.samba.org/index.php/Keytab_Extraction

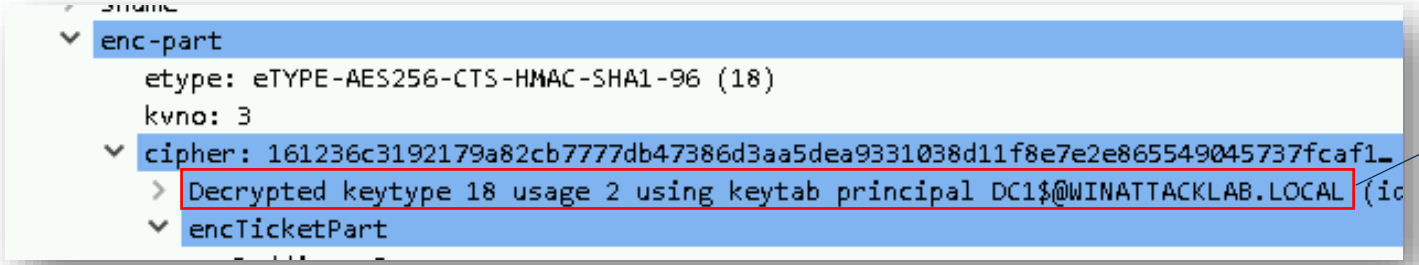
→ **Important:** Creating a domain-wide keytab of your productive environment is **not a good idea!**

Domain-Wide Keytab File

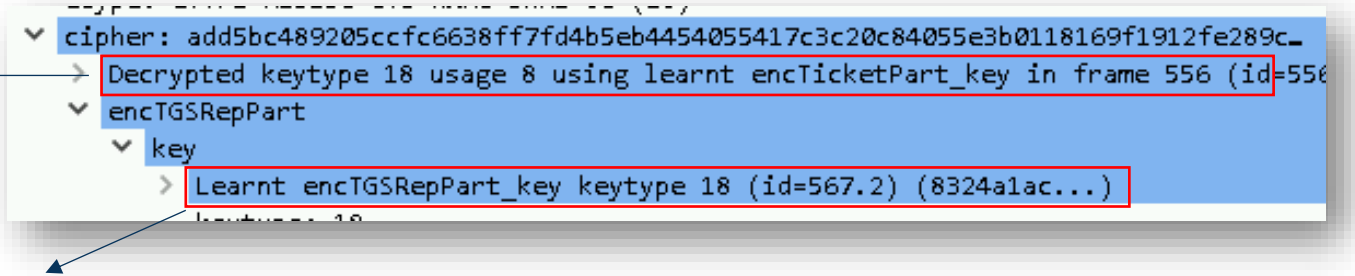
```
# samba-tool drs clone-dc-database WINATTACKLAB.LOCAL --targetdir=out --include-
secrets --server=10.0.1.100 -U winattacklab.local/ffast
Password for [WINATTACKLAB.LOCAL\ffast]:
Provision OK for domain DN DC=winattacklab,DC=local
Starting replication
...

# samba-tool domain exportkeytab domain.keytab --configfile=out/etc/smb.conf
Export complete keytab to domain.keytab

# klist -kte domain.keytab
Keytab name: FILE: domain.keytab
KVNO Timestamp           Principal
---- ------------------- ---------------------------------
2    05/17/2022 13:16:54 aalfort@WINATTACKLAB.LOCAL (aes256-cts-hmac-sha1-96)
8    05/17/2022 13:16:54 krbtgt@WINATTACKLAB.LOCAL (aes128-cts-hmac-sha1-96)
3    05/17/2022 13:16:54 DC1$@WINATTACKLAB.LOCAL (DEPRECATED:arcfour-hmac)
...
```

Decryption in Wireshark

- Once the domain-wide keytab file is configured in Wireshark, all messages will be decrypted
- Also works for previously recorded messages, unless the password have changed in-between



Was encrypted with long-term key of `DC1$` account

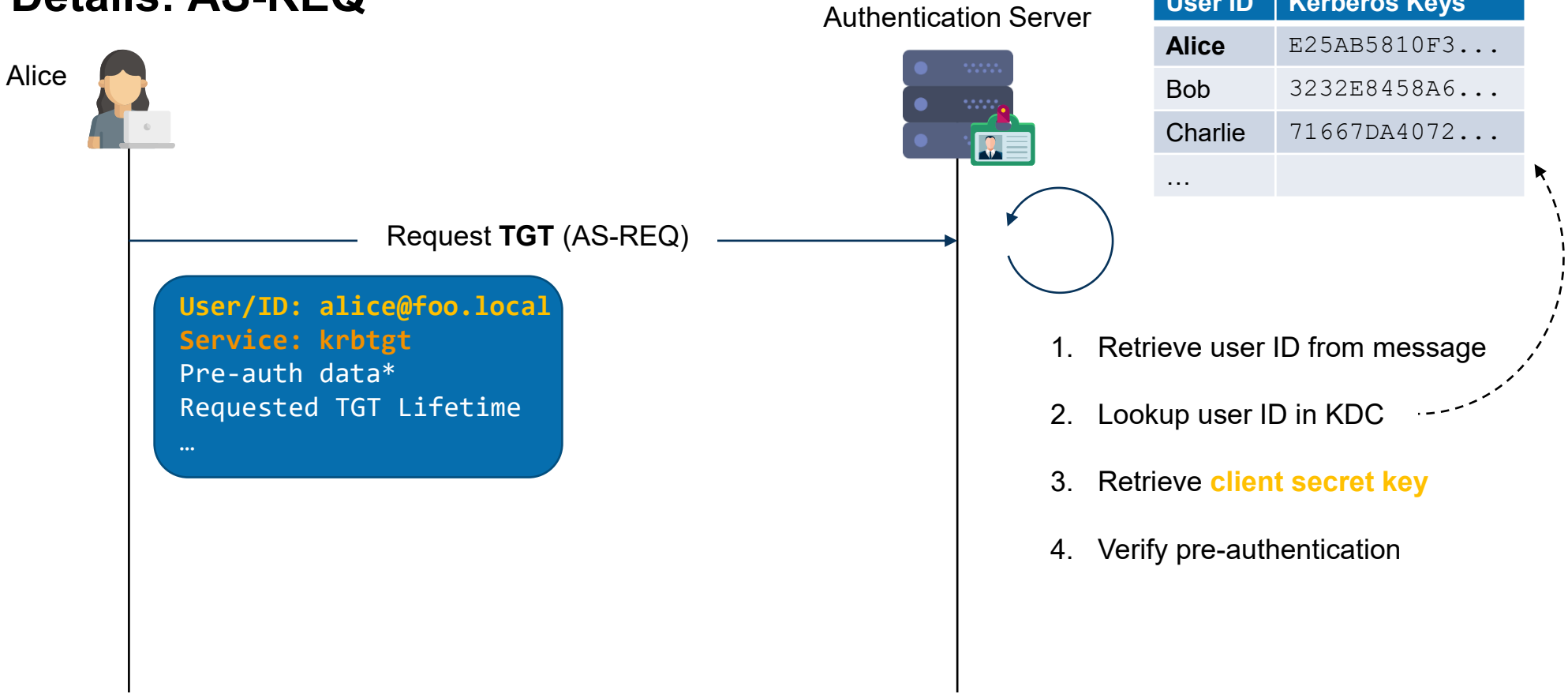


Was encrypted with session key, which was exchanged in a previous message (frame 556 in Wireshark)

Automatically extracted and stored a new session-key contained in this message

Protocol Details

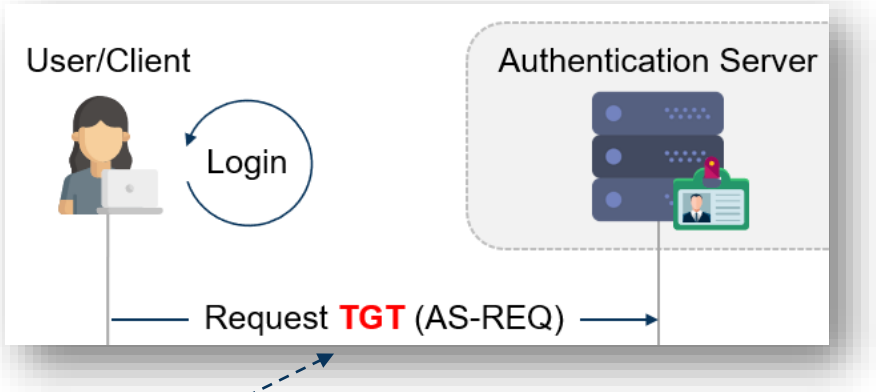
Details: AS-REQ



User ID	Kerberos Keys
Alice	E25AB5810F3...
Bob	3232E8458A6...
Charlie	71667DA4072...
…	

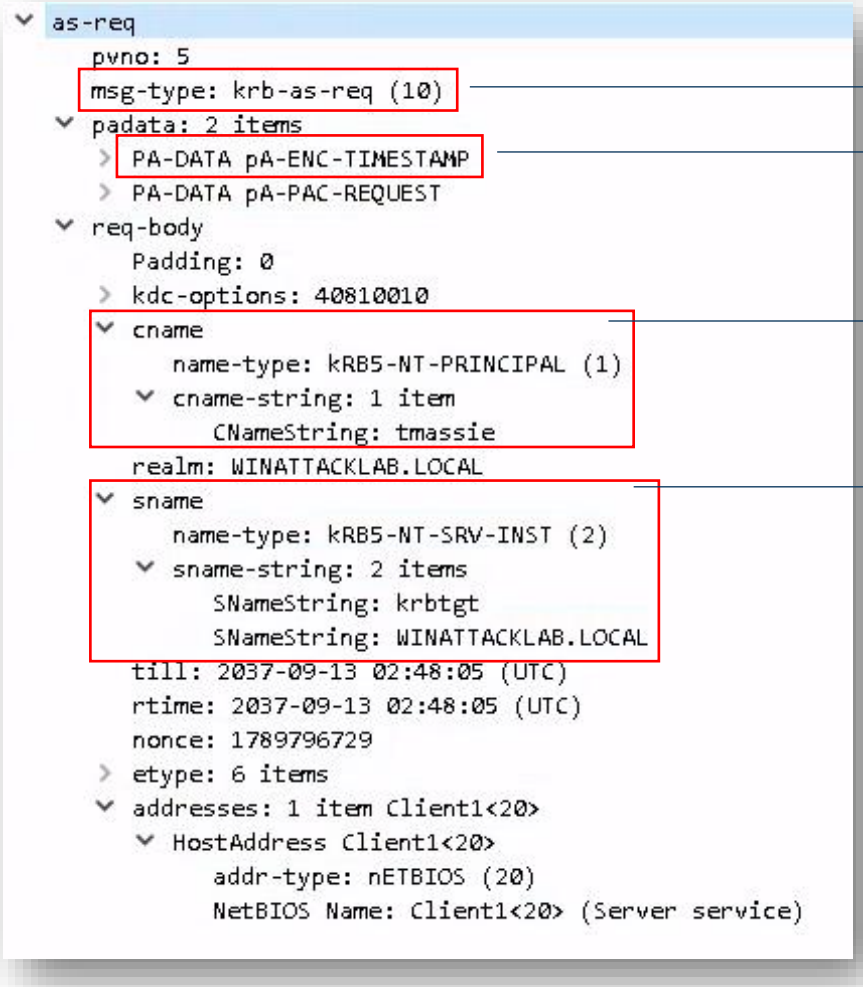
* Current timestamp, encrypted with the user's password (see next slide for details)

Kerberos Pre-Authentication



- To request a TGT, users must be authenticated
- This is called **Kerberos Pre-Authentication**
- Process:
  - Client encrypts a timestamp with the user's secret key
  - The encrypted timestamp is added to the first request (AS-REQ)
  - The KDC can decrypt and verify the timestamp
- This confirms that:
  - The user has provided the correct password
  - The message is not a replay attack
- Pre-Authentication is enabled by default, but can be **disabled** manually (for all/specific users)
- **Disabling pre-authentication is dangerous and leads to a vulnerability called ASREP-Roasting!**

Details: AS-REQ in Wireshark



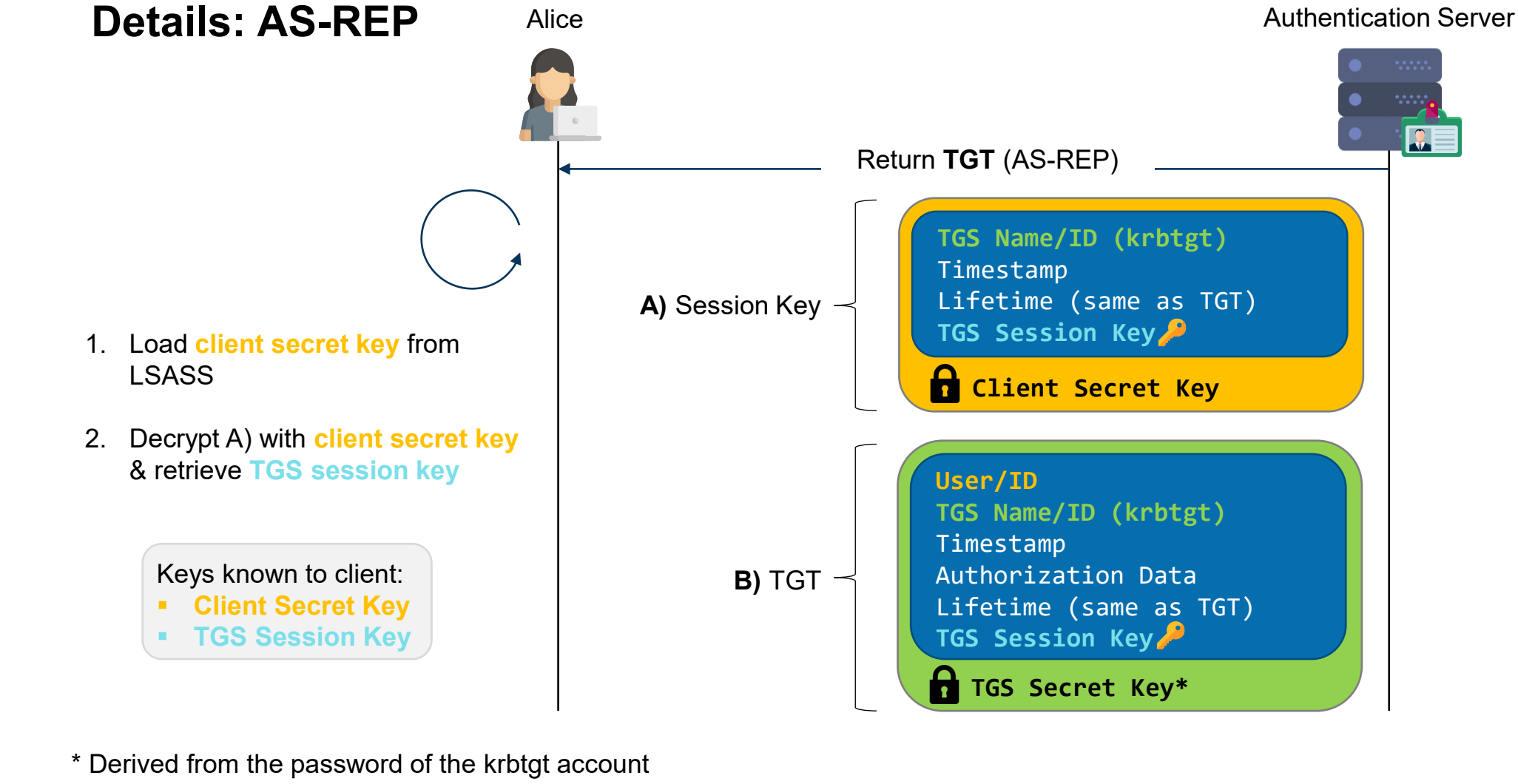
AS-REQ message

Pre-auth. Data (encrypted timestamp)

Client name/ID

Requested service, always `krbtgt` for AS-REQ

Details: AS-REP

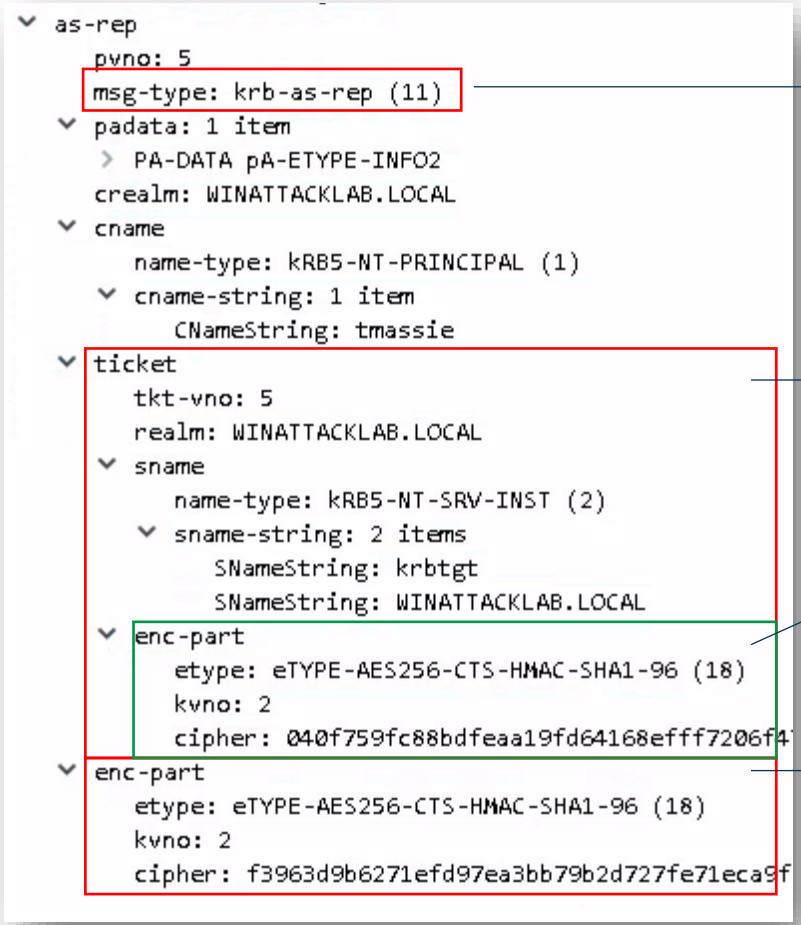


1. Load **client secret key** from LSASS
2. Decrypt A) with **client secret key** & retrieve **TGS session key**

Keys known to client:
- **Client Secret Key**
- **TGS Session Key**

* Derived from the password of the krbtgt account

Details: AS-REP in Wireshark

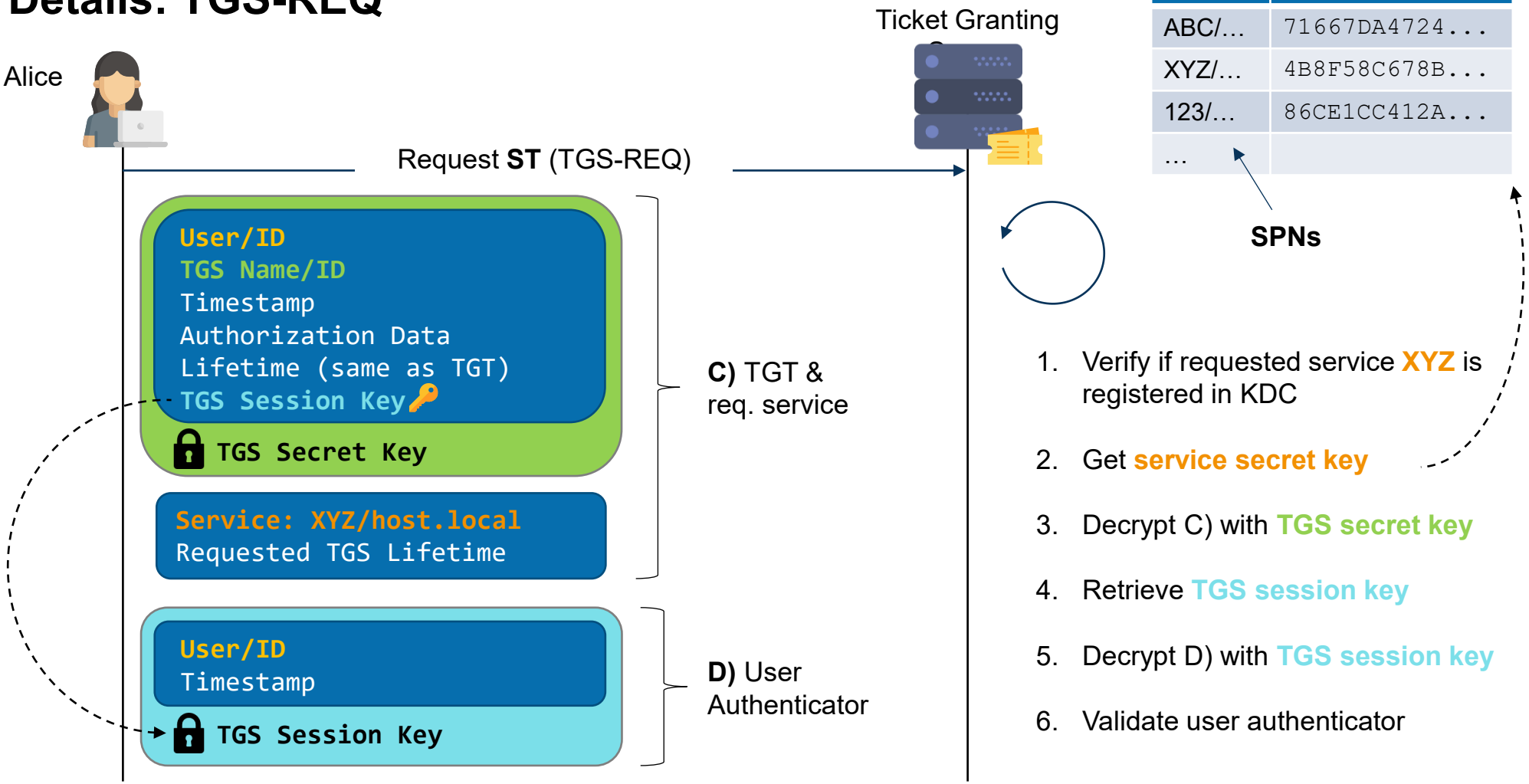


AS-REP message

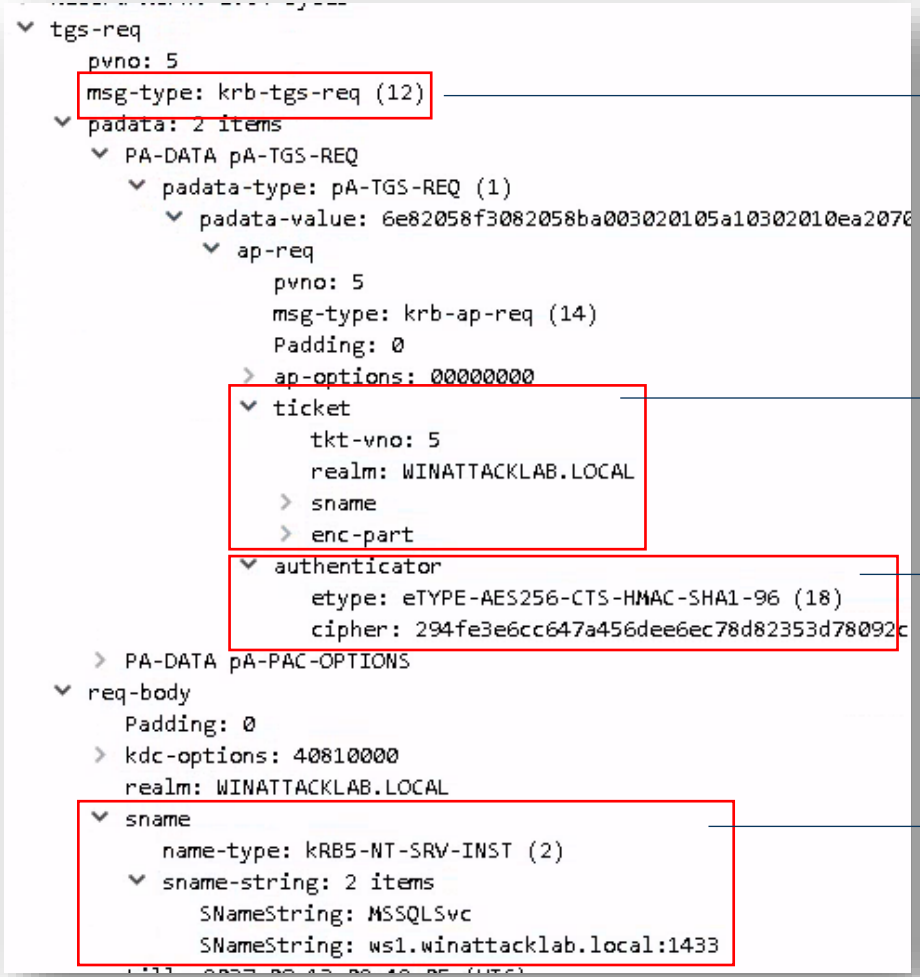
TGT with **encrypted part** (includes TGS session key)

TGS session key (encrypted with client secret key)

Details: TGS-REQ



Details: TGS-REQ in Wireshark



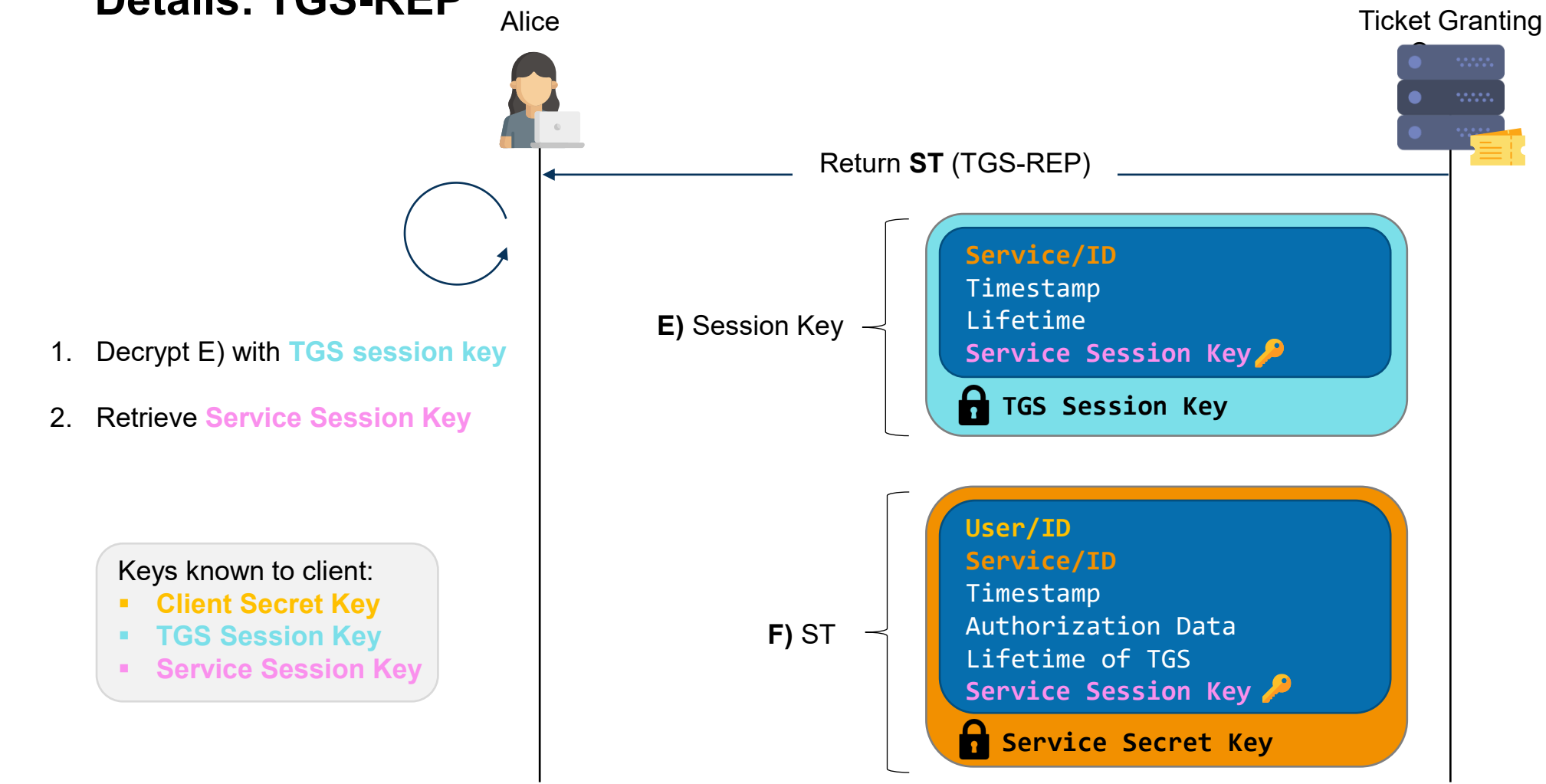
TGS-REQ message

TGT with enc. TGS session key (from previous step)

User Authenticator (encrypted with TGS session key)

SPN of requested service

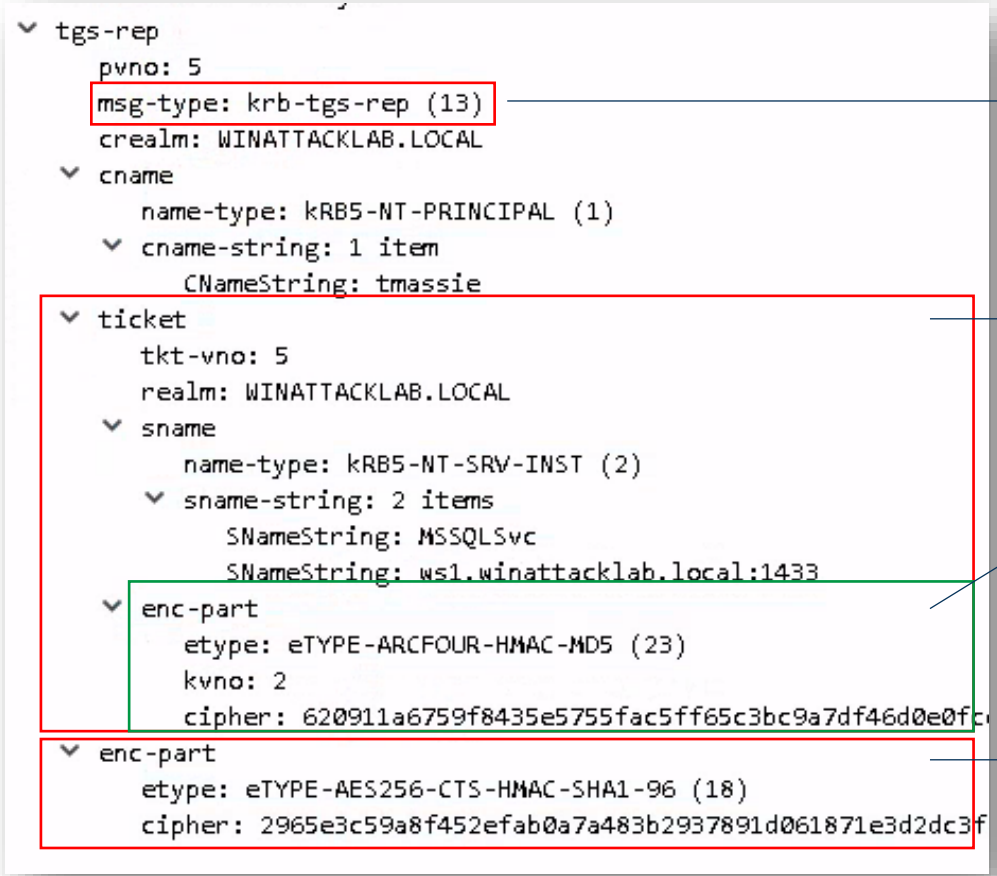
Details: TGS-REP



1. Decrypt E) with TGS session key
2. Retrieve Service Session Key

Keys known to client:
- Client Secret Key
- TGS Session Key
- Service Session Key

Details: TGS-REP in Wireshark

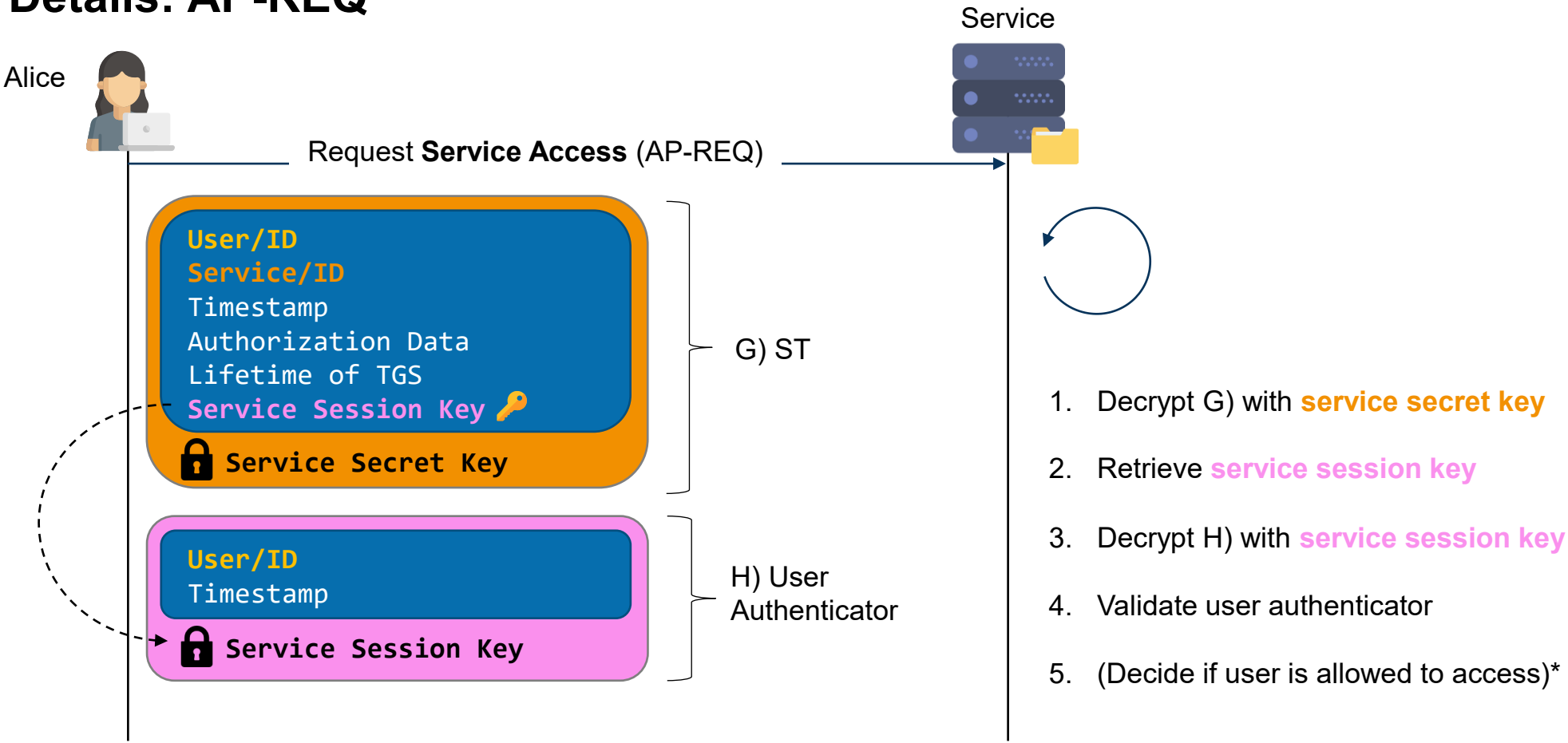


TGS-REQ message

ST with **encrypted part** (incl. service session key)

Service session key (encrypted with TGS secret key)

Details: AP-REQ



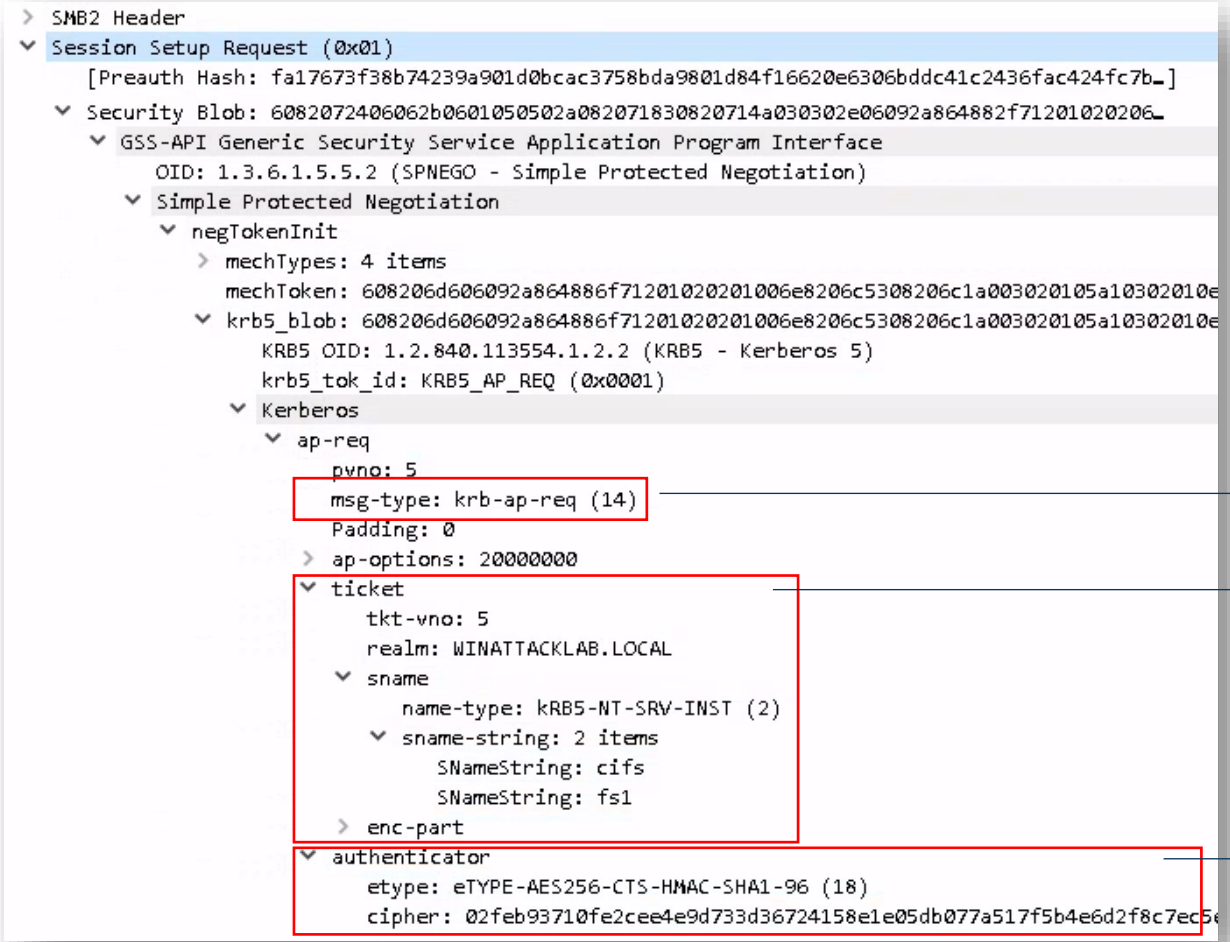
1. Decrypt G) with **service secret key**
2. Retrieve **service session key**
3. Decrypt H) with **service session key**
4. Validate user authenticator
5. (Decide if user is allowed to access)*

* Not actually part of Kerberos

Details: AP-REQ in Wireshark

AP-REQ is not performed via Kerberos protocol, but instead wrapped into the respective application protocol

Details: AP-REQ in Wireshark – SMB Example

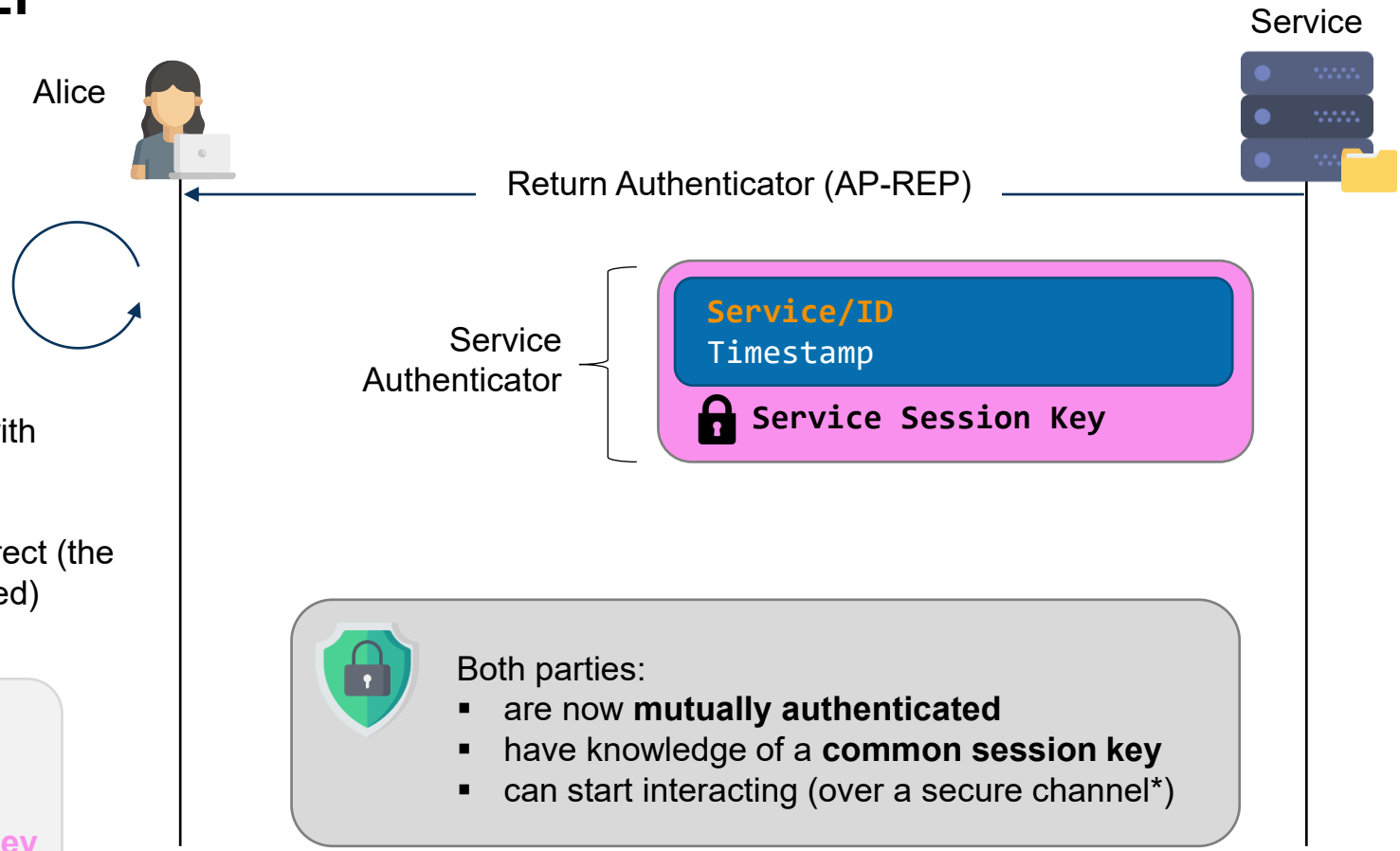


AP-REQ message

ST, containing the service session key (encrypted)

User authenticator, encrypted with service session key

Details: AP-REP



1. Decrypts authenticator with **service session key**
2. Verify if service ID is correct (the same as initially requested)

Keys known to client:
- **Client Secret Key**
- **TGS Session Key**
- **Service Session Key**

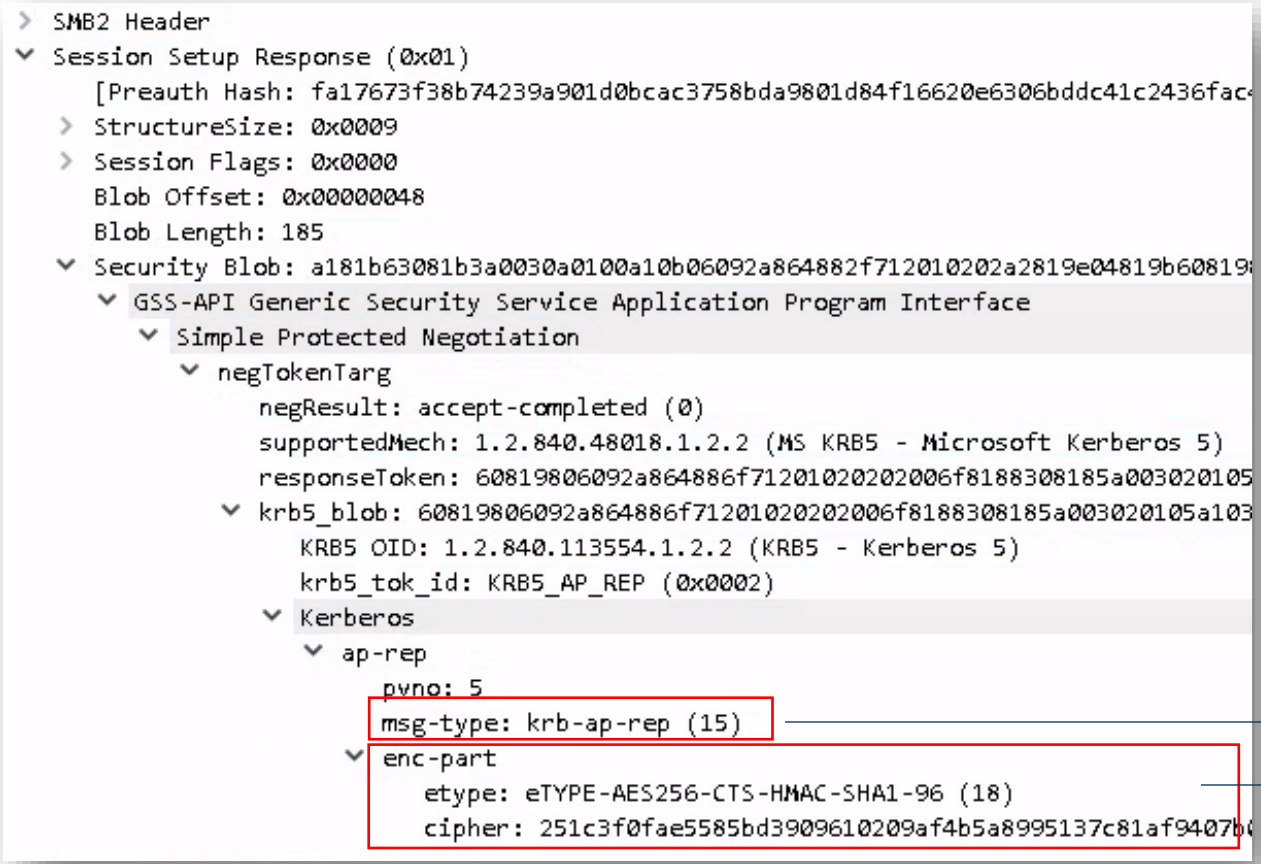
Both parties:
- are now **mutually authenticated**
- have knowledge of a **common session key**
- can start interacting (over a secure channel*)

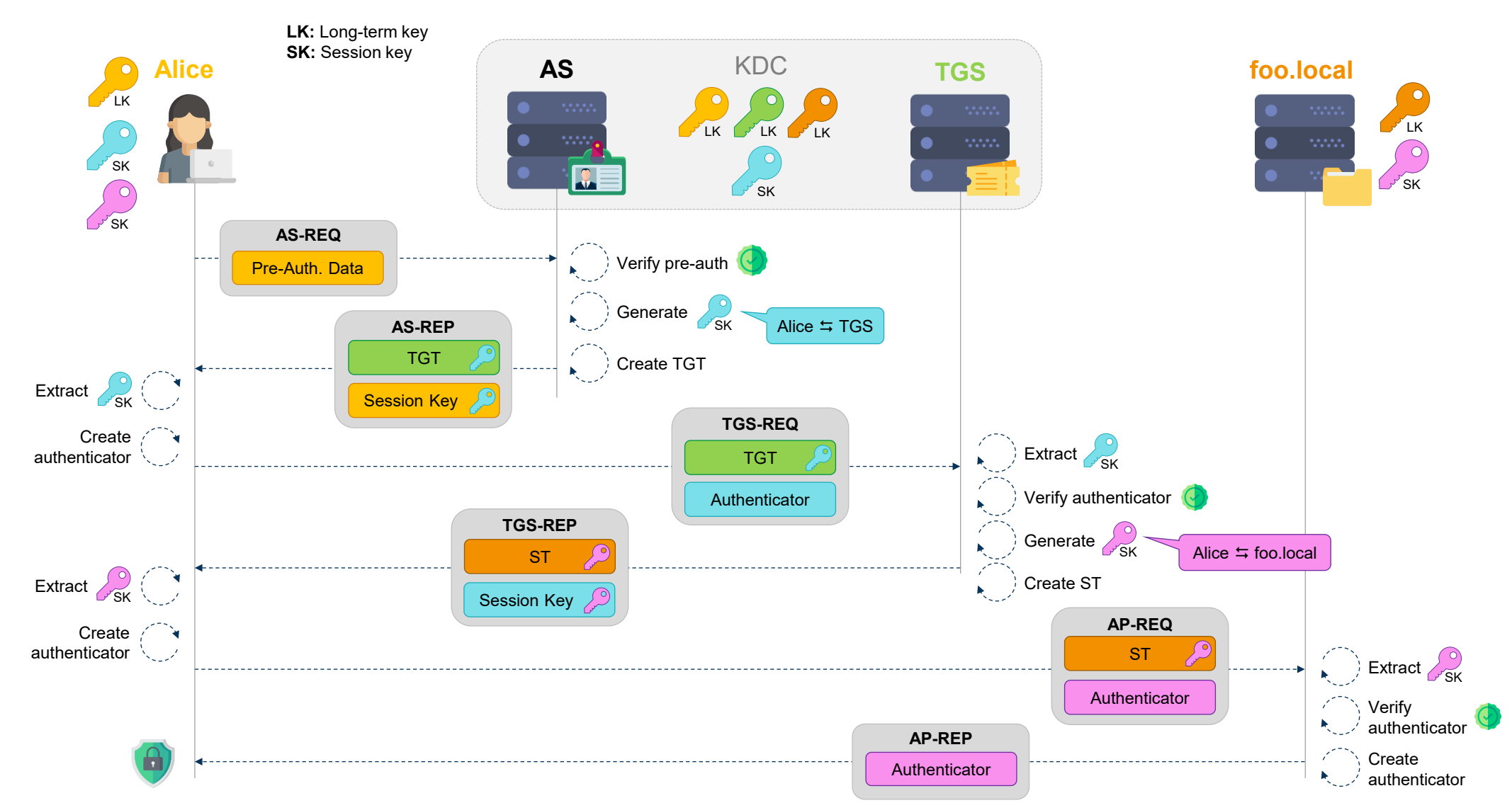
* Depends on the protocol

Details: AP-REP in Wireshark

AP-REP is not performed via Kerberos protocol, but instead wrapped into the respective application protocol

Details: AP-REP in Wireshark – SMB Example




LK: Long-term key
SK: Session key
Alice
LK
SK
SK
AS
KDC
TGS
LK
LK
LK
SK
foo.local
LK
SK
AS-REQ
Pre-Auth. Data
Verify pre-auth
Generate
SK
Alice ⇆ TGS
AS-REP
TGT
Session Key
Create TGT
Extract
SK
Create authenticator
TGS-REQ
TGT
Authenticator
Extract
SK
Verify authenticator
Generate
SK
Alice ⇆ foo.local
Create ST
TGS-REP
ST
Session Key
Extract
SK
Create authenticator
AP-REQ
ST
Authenticator
Extract
SK
Verify authenticator
Create authenticator
AP-REP
Authenticator

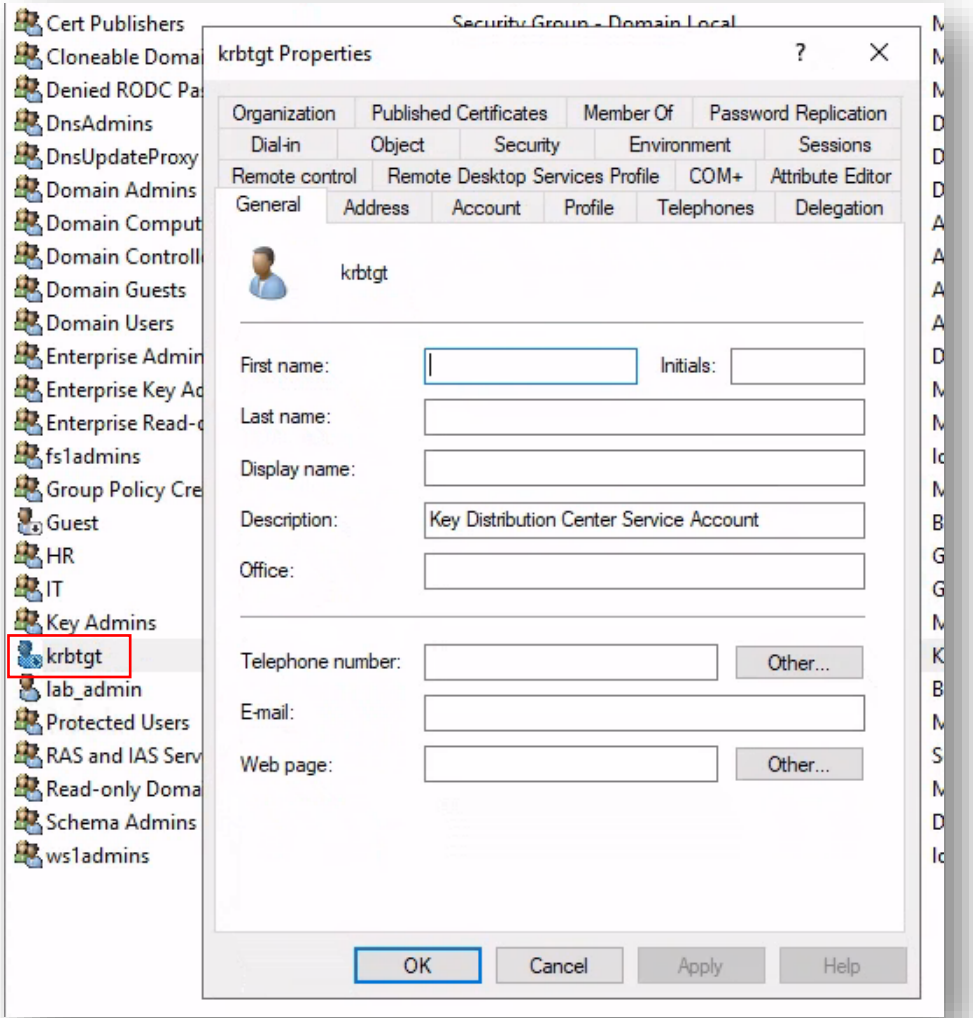
Details – Authorization Data

- Both TGTs and STs contain authorization information about the respective user
- Information is represented in the **Privileged Attribute Certificate** (PAC)
- The PAC contains:
  - User information (name, profile, home directory etc.)
  - Account information (bad password count, last logon, last password change etc.)
  - Group information (membership of AD groups)
- Services receiving tickets can use the PAC to perform authorization decisions without contacting the domain controller separately
- The PAC is signed by the KDC
- Services **may** send the PAC checksum to the KDC for validation (KERB_VERIFY_PAC)
- However, PAC validation via KDC is only performed in specific cases*

* https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-apds/0926c518-19a2-4981-a1ac-67694f078930

Details: krbtgt Account

- Built-in account for the KDC service
- Its key material is used to encrypt TGTs
- Disabled by default, can't be activated
- The account's password is generated randomly when setting up the domain



Details: krbtgt Account Compromise

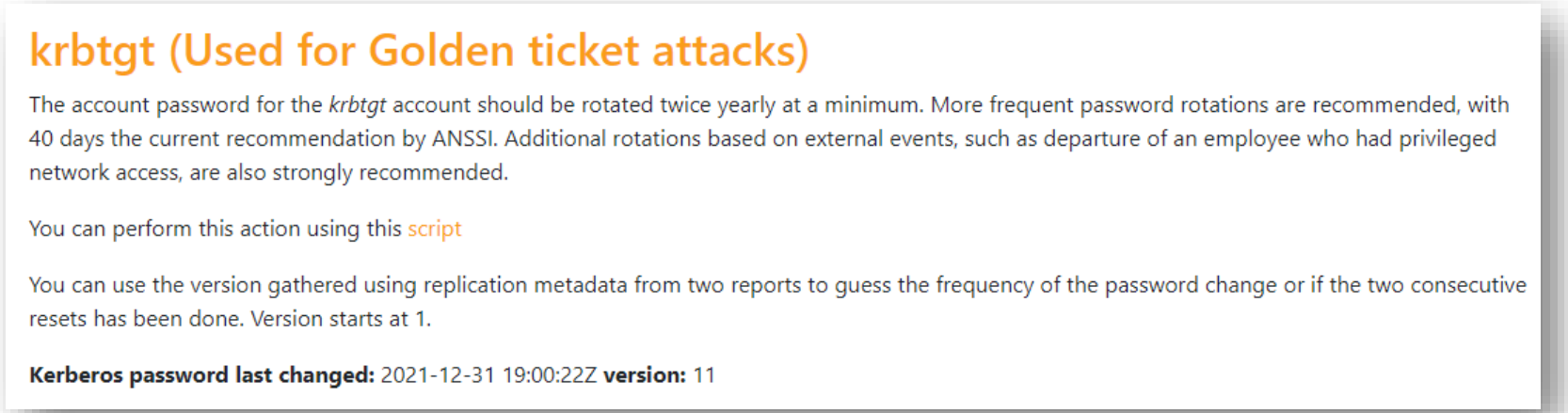
- **Compromising** the **krbtgt** account grants **full control** over the domain
- Often, this is done via DCSync, which typically requires domain admin privileges*
- With the account's Kerberos keys, one can craft their own TGTs for whatever service and user they desire
- This is known as a so-called "**Golden Ticket**" attack
- Golden Tickets can have arbitrary life-times, granting attackers access over extended time

- Compromising a service account (with an SPN) allows an attacker to forge service tickets for this specific service only (but as any user)
- This is known as a so-called "**Silver Ticket**" attack

* Or specifically DS-Replication-Get-Changes and DS-Replication-Get-Changes-All

Details: krbtgt Account Password Change

- Microsoft recommends changing the krbtgt password regularly (twice a year or more)
- This requires a domain functional level of Windows Server 2008 or greater
- Some specific measures apply when changing the password*
- Also reported by PingCastle:



krbtgt (Used for Golden ticket attacks)

The account password for the *krbtgt* account should be rotated twice yearly at a minimum. More frequent password rotations are recommended, with 40 days the current recommendation by ANSSI. Additional rotations based on external events, such as departure of an employee who had privileged network access, are also strongly recommended.

You can perform this action using this script

You can use the version gathered using replication metadata from two reports to guess the frequency of the password change or if the two consecutive resets has been done. Version starts at 1.

Kerberos password last changed: 2021-12-31 19:00:22Z **version:** 11

* More information: https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-R2-and-2012/dn745899(v=ws.11)?redirectedfrom=MSDN#krbtgt-account-maintenance-considerations


COMPASS
SECURITY