



# Kerberos Deep Dive

## Part 1 - Introduction

July 2025, Alex Joss

# **Content Overview**

**Part 1 - Kerberos Introduction**

**Part 2 - Kerberoasting**

**Part 3 - AS-REP Roasting**

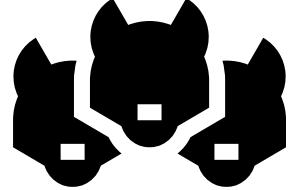
**Part 4 - Unconstrained Delegation**

**Part 5 - Constrained Delegation**

**Part 6 - Resource-Based Constrained Delegation**

# **What Is Kerberos?**

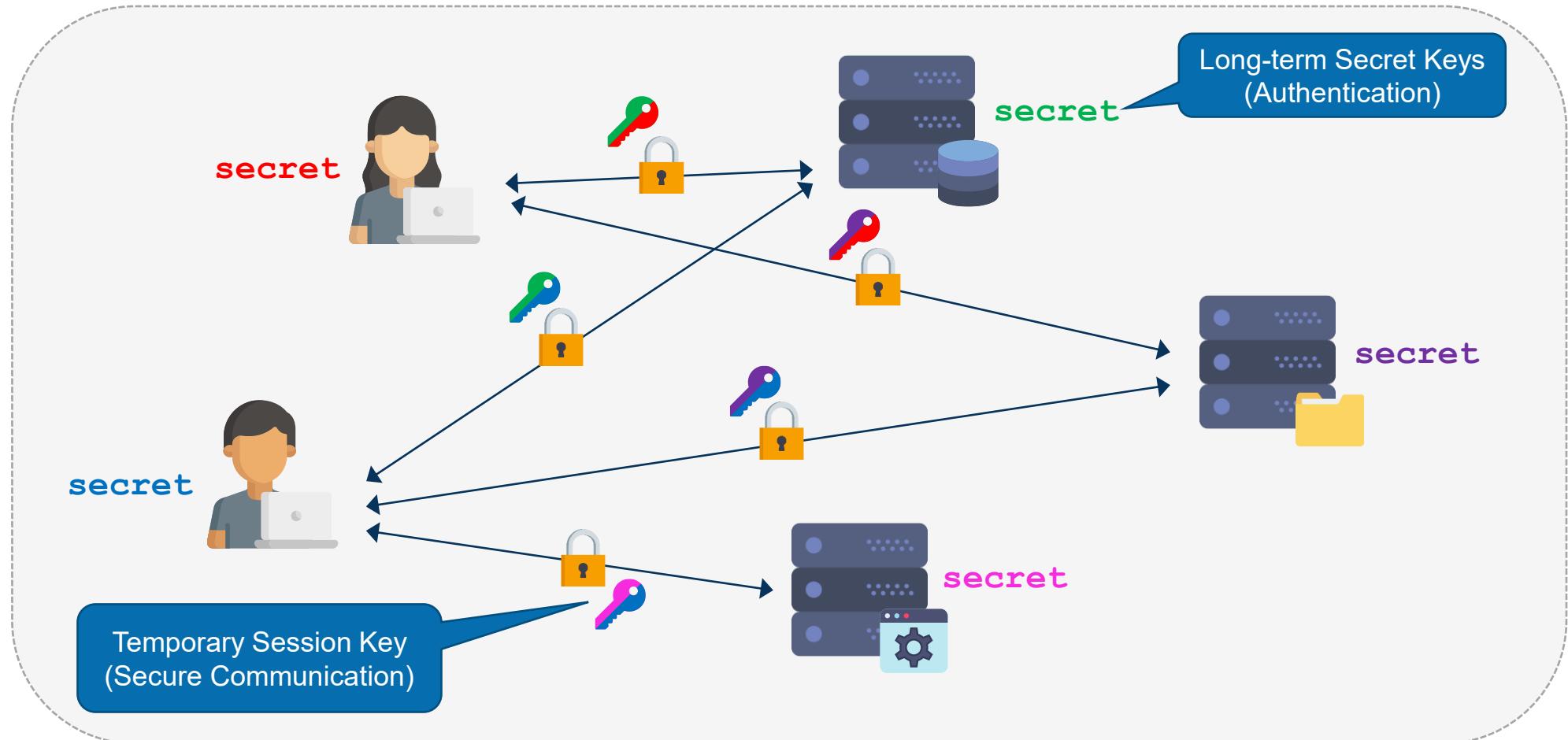
# Kerberos



- 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



# 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/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

## Kerberos Realm

Users / Clients



Key Distribution Center (KDC)

Authentication Server (AS)



Ticket Granting Server (TGS)



Services with SPNs



Secret & Session Keys



Ticket Granting Ticket (TGT)

Service Ticket (ST)

# Details: KDC



- The KDC is the **trusted 3<sup>rd</sup> 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](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

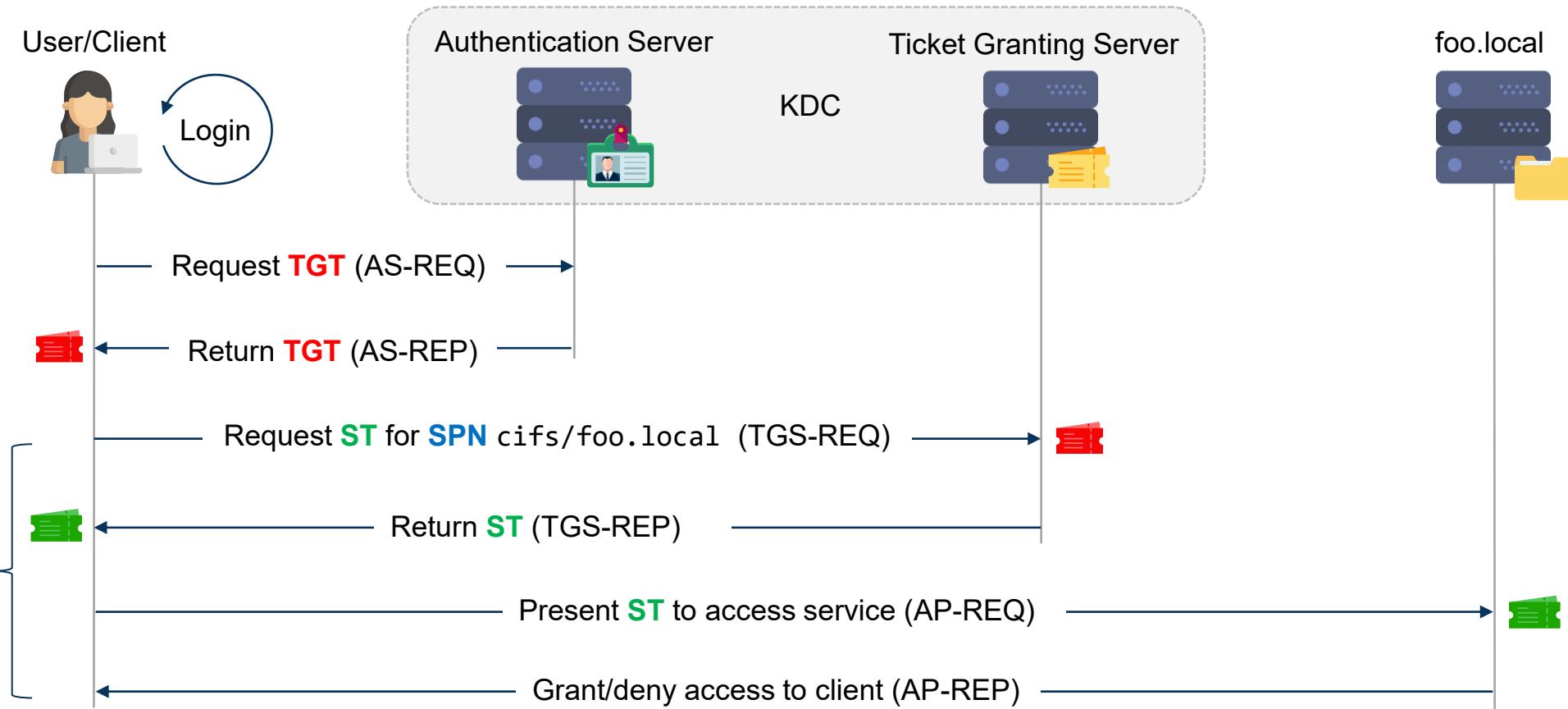
You can inspect your current tickets with the `klist` utility:

```
PS C:\> whoami  
winattacklab\tmassie  
PS C:\> klist  
  
Current LogonId is 0:0x8c81c1  
  
Cached Tickets: (2)  
  
#0> Client: tmassie @ WINATTACKLAB.LOCAL  
Server: krbtgt/WINATTACKLAB.LOCAL @ WINATTACKLAB.LOCAL  
KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96  
Ticket Flags 0x40e10000 -> Forwardable renewable initial pre_authent name_canonicalize  
Start Time: 2/23/2022 14:43:31 (local)  
End Time: 2/24/2022 0:43:31 (local)  
Renew Time: 3/2/2022 14:43:31 (local)  
Session Key Type: AES-256-CTS-HMAC-SHA1-96  
Cache Flags: 0x1 -> PRIMARY  
Kdc Called: DC1.winattacklab.local  
  
#1> Client: tmassie @ WINATTACKLAB.LOCAL  
Server: MSSQLSvc/ws1.winattacklab.local:1433 @ WINATTACKLAB.LOCAL  
KerbTicket Encryption Type: RSADSI RC4-HMAC(NT)  
Ticket Flags 0x40a10000 -> Forwardable renewable pre_authent name_canonicalize  
Start Time: 2/23/2022 14:43:31 (local)  
End Time: 2/24/2022 0:43:31 (local)  
Renew Time: 3/2/2022 14:43:31 (local)  
Session Key Type: RSADSI RC4-HMAC(NT)  
Cache Flags: 0  
Kdc Called: DC1.winattacklab.local  
  
PS C:\>
```

TGT

ST for MSSQL  
on WS1

# High Level Kerberos Authentication Flow



**TGT: Ticket Granting Ticket**

**ST: Service Ticket**

**SPN: Service Principal Name**

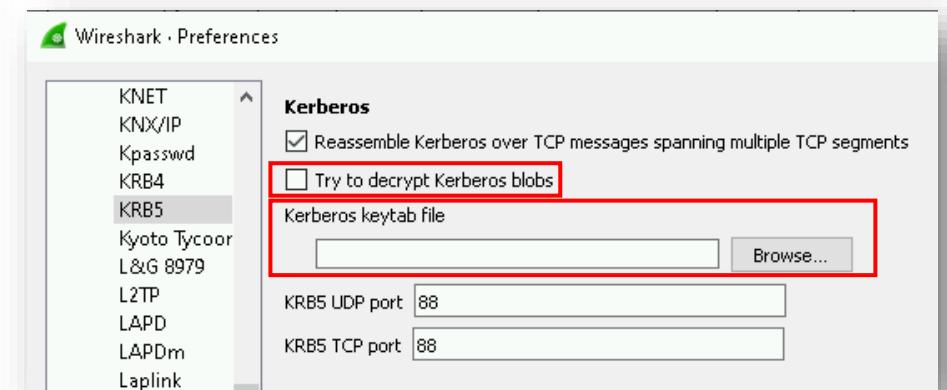
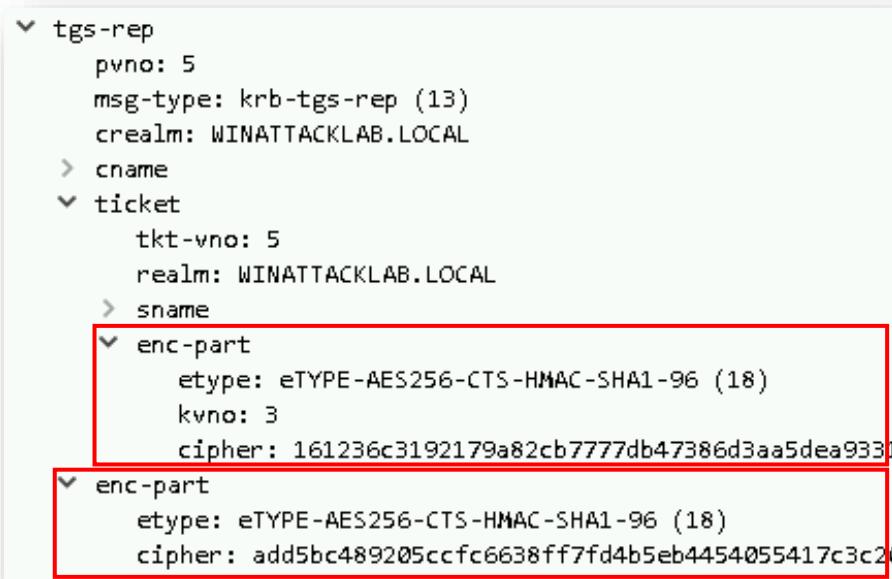
# Kerberos & Wireshark

- 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)

No.	Time	Source	Destination	Protocol	Length	Info
546	14.111583	10.0.1.10	10.0.1.100	KRB5	298	AS-REQ
547	14.113279	10.0.1.100	10.0.1.10	KRB5	278	KRB Error: KRB5KDC_ERR_PREAMUTH_REQUIRED
554	14.114462	10.0.1.10	10.0.1.100	KRB5	378	AS-REQ
556	14.115904	10.0.1.100	10.0.1.10	KRB5	358	AS-REP
564	14.117342	10.0.1.10	10.0.1.100	KRB5	1833	TGS-REQ
567	14.119312	10.0.1.100	10.0.1.10	KRB5	343	TGS-REP
575	14.120234	10.0.1.10	10.0.1.100	KRB5	1638	TGS-REQ
578	14.121014	10.0.1.100	10.0.1.10	KRB5	248	TGS-REP
581	14.121332	10.0.1.10	10.0.1.100	SMB2	3591	Session Setup Request
585	14.122546	10.0.1.100	10.0.1.10	SMB2	315	Session Setup Response

# 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](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

The screenshot shows two network frames in Wireshark. Frame 556 is expanded to show its structure:

- enc-part**:
  - etype: eTYPE-AES256-CTS-HMAC-SHA1-96 (18)
  - kvno: 3
  - cipher: 161236c3192179a82cb7777db47386d3aa5dea9331038d11f8e7e2e865549045737fcraf1...** (highlighted in red)
  - Decrypted keytype 18 usage 2 using keytab principal DC1\$@WINATTACKLAB.LOCAL (id=556)
  - encTicketPart

A blue arrow points from the "Was encrypted with long-term key of DC1\$ account" annotation to the "Decrypted keytype 18 usage 2 using keytab principal DC1\$@WINATTACKLAB.LOCAL (id=556)" text.

Frame 557 is also expanded:

- cipher: add5bc489205ccfc6638ff7fd4b5eb4454055417c3c20c84055e3b0118169f1912fe289c...** (highlighted in red)
- Decrypted keytype 18 usage 8 using learnt encTicketPart\_key in frame 556 (id=556)
- encTGSRepPart
  - key
    - Learnt encTGSRepPart\_key keytype 18 (id=567.2) (8324a1ac...)

Two blue arrows point from the "Was encrypted with session key, which was exchanged in a previous message (frame 556 in Wireshark)" annotation to the "Decrypted keytype 18 usage 8 using learnt encTicketPart\_key in frame 556 (id=556)" text, and from the "Automatically extracted and stored a new session-key contained in this message" annotation to the "Learnt encTGSRepPart\_key keytype 18 (id=567.2) (8324a1ac...)" text.

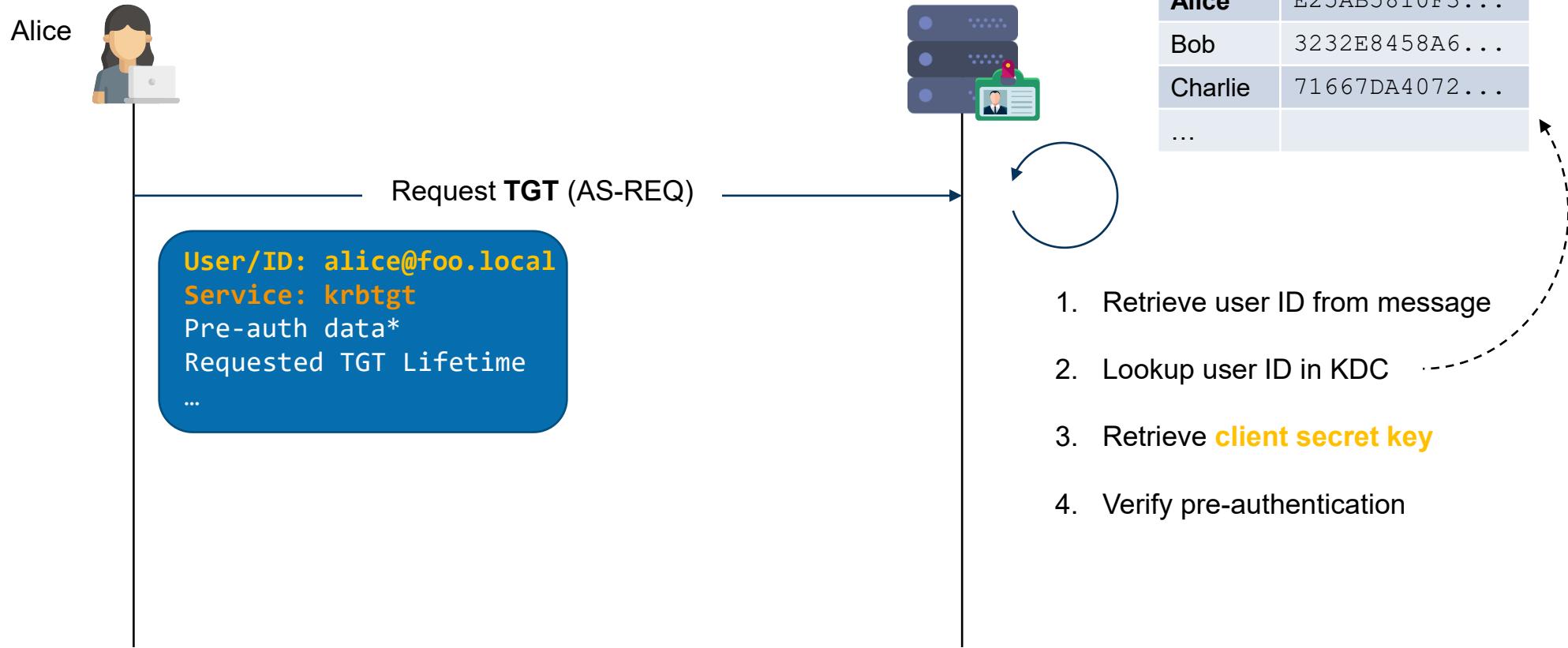
Was encrypted with long-term key of DC1\$ account

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

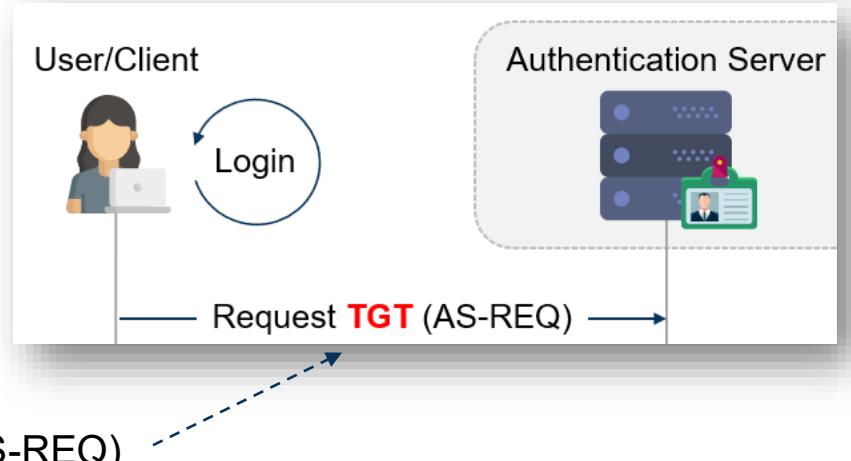
# Details: AS-REQ



\* 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!**

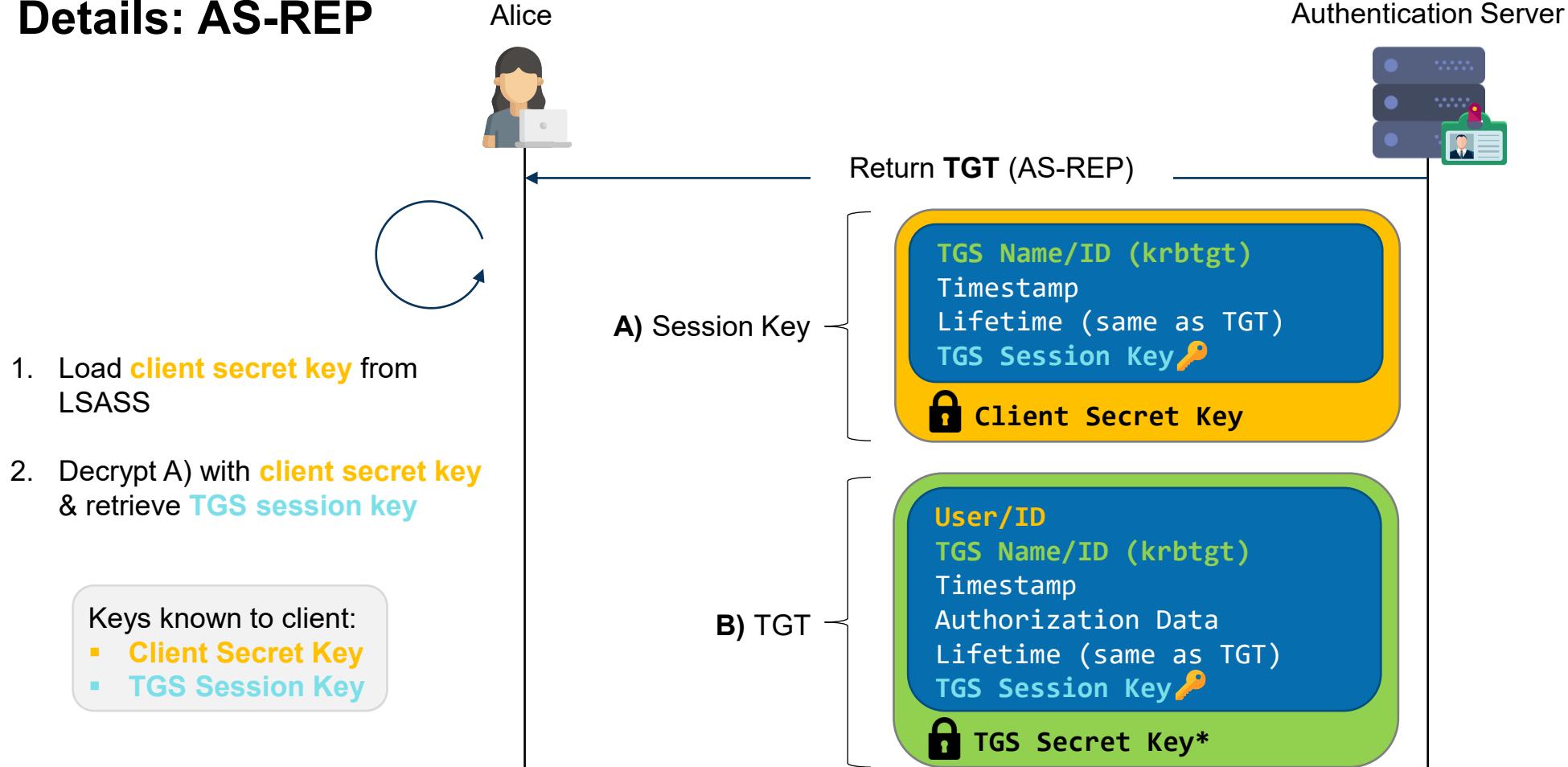


# Details: AS-REQ in Wireshark

The image shows a Wireshark tree view of an AS-REQ message. The message structure is as follows:

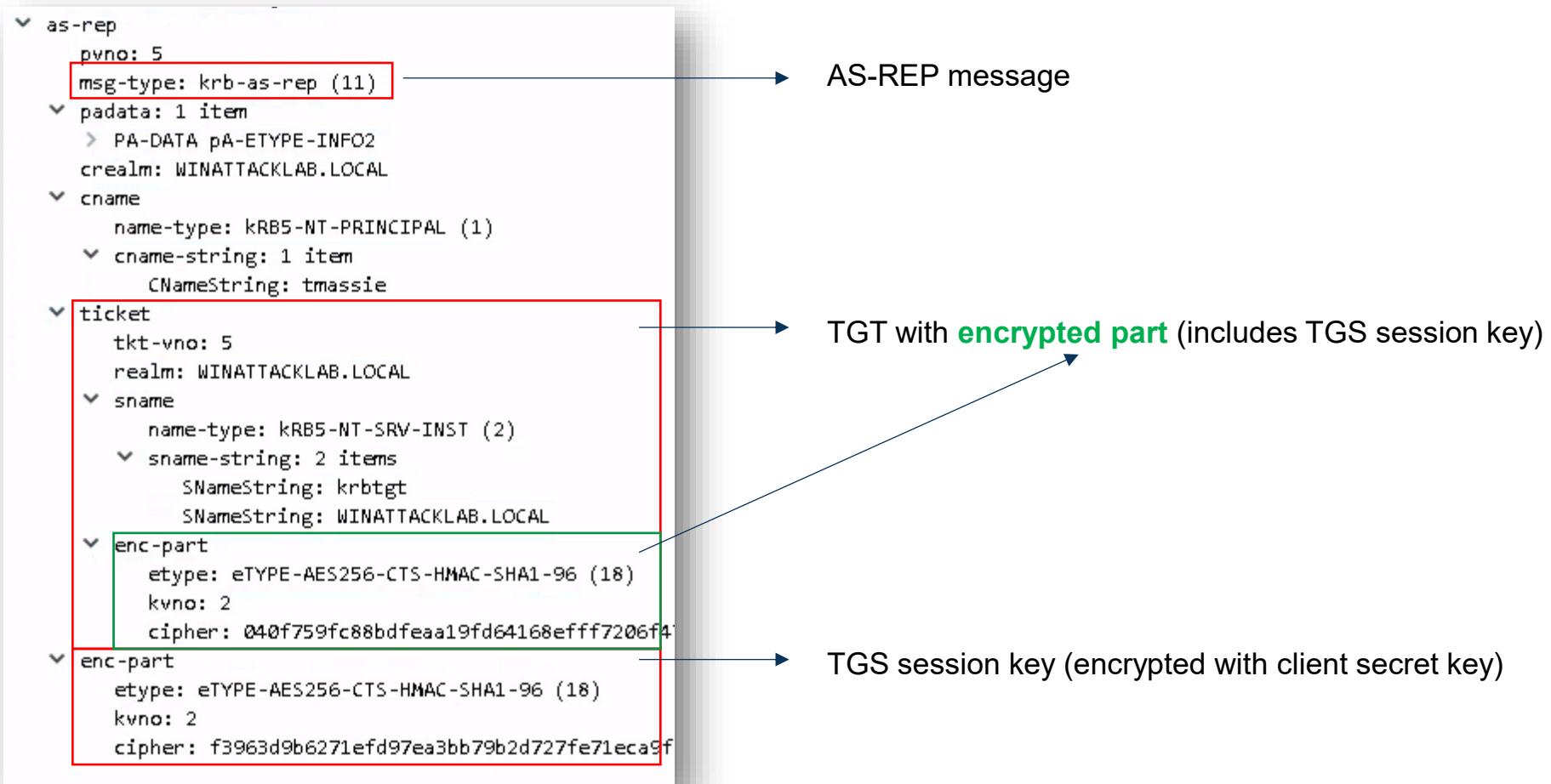
- as-req**
  - pvno: 5**
  - msg-type: krb-as-req (10)** → AS-REQ message
  - padata: 2 items**
    - > PA-DATA pA-ENC-TIMESTAMP
    - > PA-DATA pA-PAC-REQUEST
  - req-body**
    - Padding: 0
    - > kdc-options: 40810010
    - cname**
      - name-type: KRB5-NT-PRINCIPAL (1)**
      - cname-string: 1 item**
        - CNameString: tmassie
    - realm: WINATTACKLAB.LOCAL**
    - sname**
      - name-type: KRB5-NT-SRV-INST (2)**
      - sname-string: 2 items**
        - SNameString: krbtgt
        - SNameString: WINATTACKLAB.LOCAL
    - till: 2037-09-13 02:48:05 (UTC)**
    - rtime: 2037-09-13 02:48:05 (UTC)**
    - nonce: 1789796729**
    - > **etype: 6 items**
    - addresses: 1 item Client1<20>**
      - HostAddress Client1<20>**
        - addr-type: nETBIOS (20)**
        - NetBIOS Name: Client1<20> (Server service)**

# Details: AS-REP

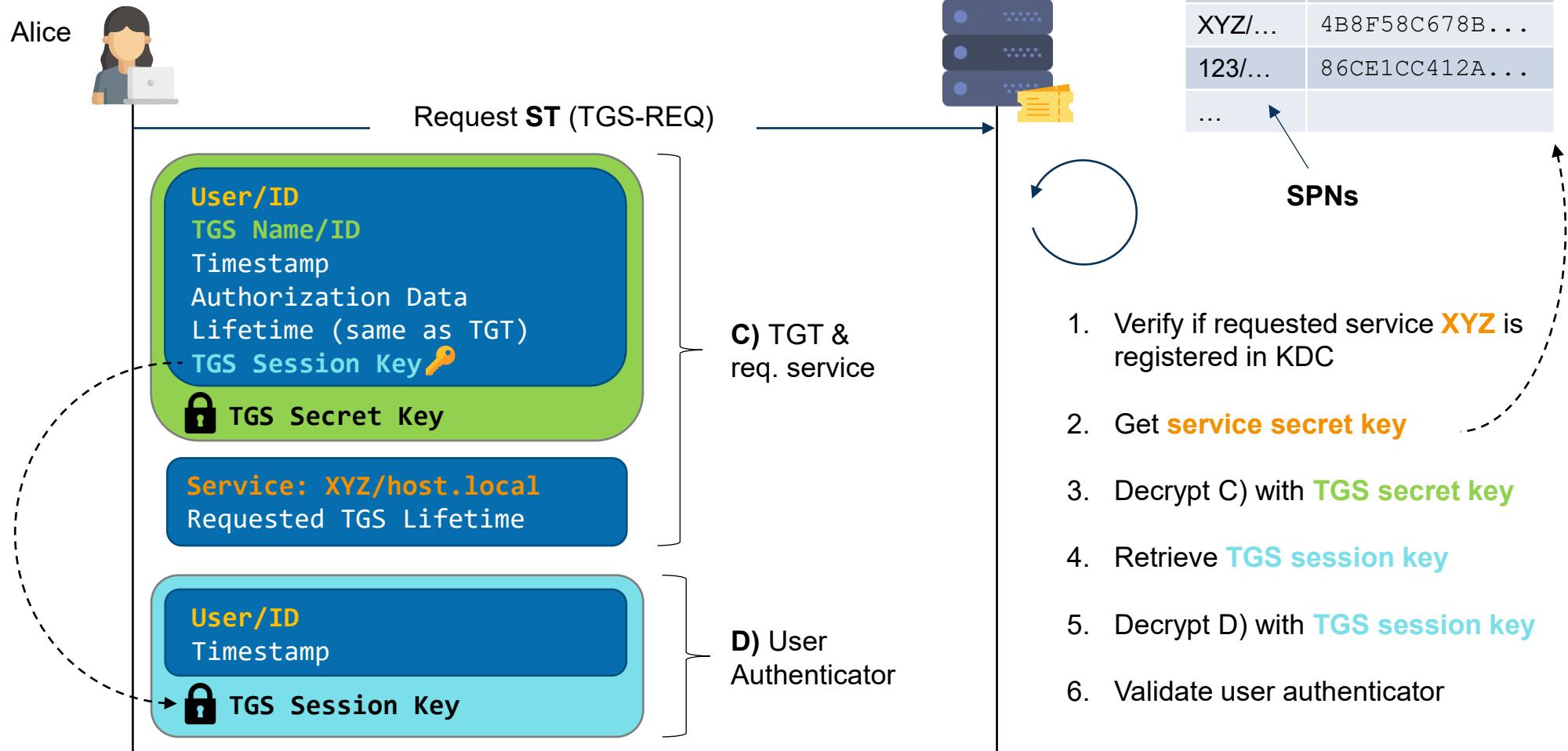


\* Derived from the password of the krbtgt account

# Details: AS-REP in Wireshark



# Details: TGS-REQ

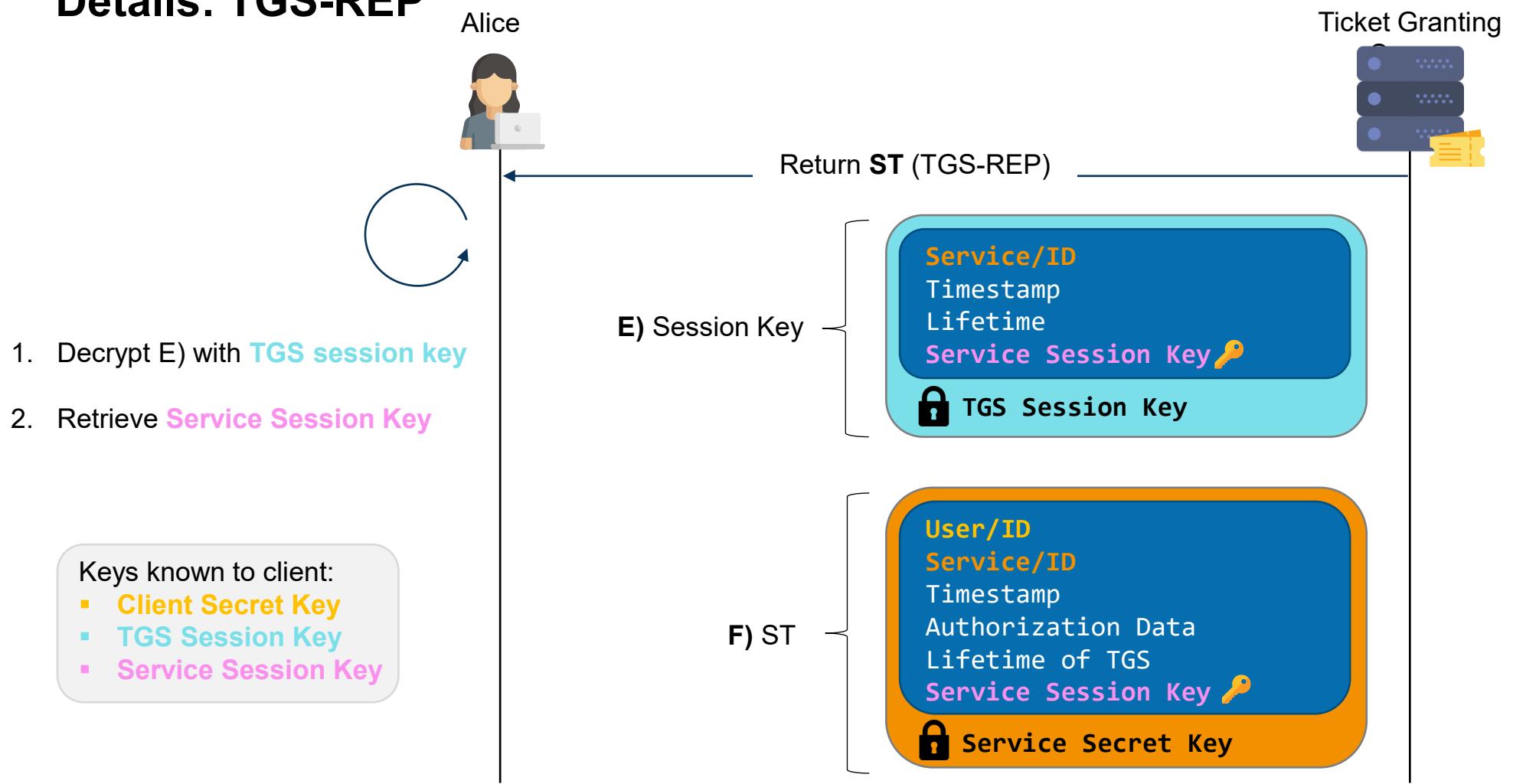


# Details: TGS-REQ in Wireshark

The image shows a Wireshark tree view of a TGS-REQ message. The message structure is as follows:

- tgs-req**:
  - pvno: 5**
  - msg-type: krb-tgs-req (12)** → TGS-REQ message
  - padata: 2 items**
    - PA-DATA pA-TGS-REQ**
      - padata-type: pA-TGS-REQ (1)**
      - padata-value: 6e82058f3082058ba003020105a10302010ea2070**
    - ap-req**
      - pvno: 5**
      - msg-type: krb-ap-req (14)**
      - Padding: 0**
      - ap-options: 00000000**
      - ticket**
        - tkt-vno: 5**
        - realm: WINATTACKLAB.LOCAL**
        - sname**
        - enc-part**
      - authenticator**
        - etype: eTYPE-AES256-CTS-HMAC-SHA1-96 (18)**
        - cipher: 294fe3e6cc647a456dee6ec78d82353d78092c**
  - PA-DATA pA-PAC-OPTIONS**
  - req-body**
    - Padding: 0**
    - kdc-options: 40810000**
    - realm: WINATTACKLAB.LOCAL**
    - sname**
      - name-type: KRB5-NT-SRV-INST (2)**
      - sname-string: 2 items**
        - SNameString: MSSQLSvc**
        - SNameString: ws1.winattacklab.local:1433**

# Details: TGS-REP



# Details: TGS-REP in Wireshark

The image shows a Wireshark tree view of a TGS-REP message. The message structure is as follows:

- tgs-rep**:
  - pvno: 5**
  - msg-type: krb-tgs-rep (13)** → TGS-REQ message
  - crealm: WINATTACKLAB.LOCAL**
- cname**:
  - name-type: kRB5-NT-PRINCIPAL (1)**
  - cname-string: 1 item**
    - CNameString: tmassie**
- ticket**:
  - tkt-vno: 5**
  - realm: WINATTACKLAB.LOCAL**
  - sname**:
    - name-type: kRB5-NT-SRV-INST (2)**
    - sname-string: 2 items**
      - SNameString: MSSQLSvc**
      - SNameString: ws1.winattacklab.local:1433**
  - enc-part**:
    - etype: eTYPE-ARCFOUR-HMAC-MD5 (23)**
    - kvno: 2**
    - cipher: 620911a6759f8435e5755fac5ff65c3bc9a7df46d0e0fc...**
  - enc-part**:
    - etype: eTYPE-AES256-CTS-HMAC-SHA1-96 (18)**
    - cipher: 2965e3c59a8f452efab0a7a483b2937891d061871e3d2dc3f...**

# Details: AP-REQ

Alice



Request Service Access (AP-REQ)

User/ID  
Service/ID  
Timestamp  
Authorization Data  
Lifetime of TGS  
**Service Session Key** 

 **Service Secret Key**

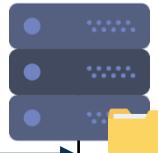
User/ID  
Timestamp

 **Service Session Key**

G) ST

H) User  
Authenticator

Service



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

The image shows a Wireshark tree view of an SMB Session Setup Request message. The message structure is as follows:

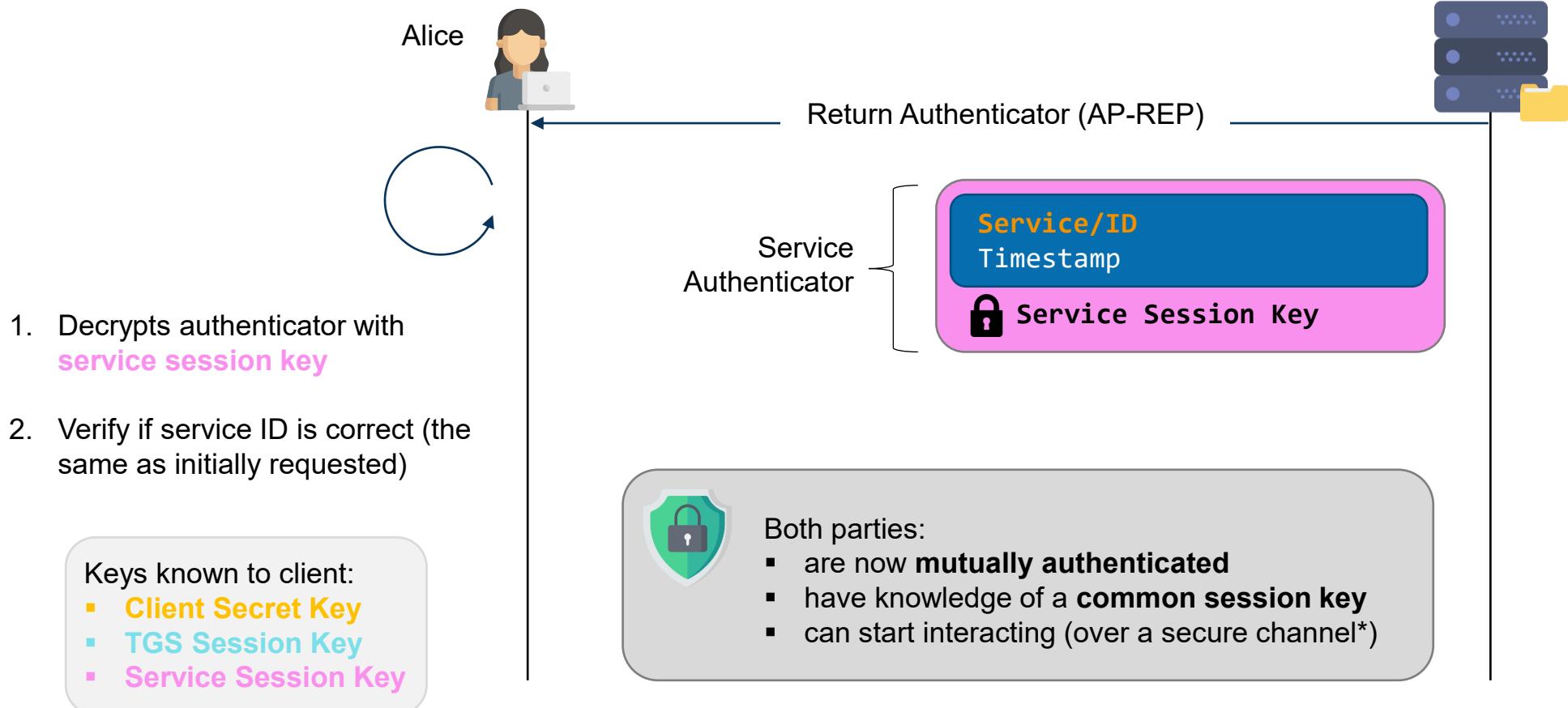
- SMB2 Header
- Session Setup Request (0x01)
  - [Preauth Hash: fa17673f38b74239a901d0bcac3758bda9801d84f16620e6306bddc41c2436fac424fc7b...]
- Security Blob: 6082072406062b0601050502a082071830820714a030302e06092a864882f71201020206
  - GSS-API Generic Security Service Application Program Interface
    - OID: 1.3.6.1.5.5.2 (SPNEGO - Simple Protected Negotiation)
  - Simple Protected Negotiation
    - negTokenInit
      - mechTypes: 4 items
        - mechToken: 608206d606092a864886f71201020201006e8206c5308206c1a003020105a10302010e
    - krb5\_blob: 608206d606092a864886f71201020201006e8206c5308206c1a003020105a10302010e
      - KRB5 OID: 1.2.840.113554.1.2.2 (KRB5 - Kerberos 5)
      - krb5\_tok\_id: KRB5\_AP\_REQ (0x0001)
    - Kerberos
      - ap-req
        - pvno: 5
          - msg-type: krb-ap-req (14)
        - Padding: 0
        - ap-options: 20000000
      - ticket
        - tkt-vno: 5
        - realm: WINATTACKLAB.LOCAL
        - sname
          - name-type: KRB5-NT-SRV-INST (2)
          - sname-string: 2 items
            - SNameString: cifs
            - SNameString: fs1
      - enc-part
    - authenticator
      - etype: eTYPE-AES256-CTS-HMAC-SHA1-96 (18)
      - cipher: 02feb93710fe2cee4e9d733d36724158e1e05db077a517f5b4e6d2f8c7ec5e

→ AP-REQ message

→ ST, containing the service session key (encrypted)

→ User authenticator, encrypted with service session key

# Details: AP-REP



\* 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

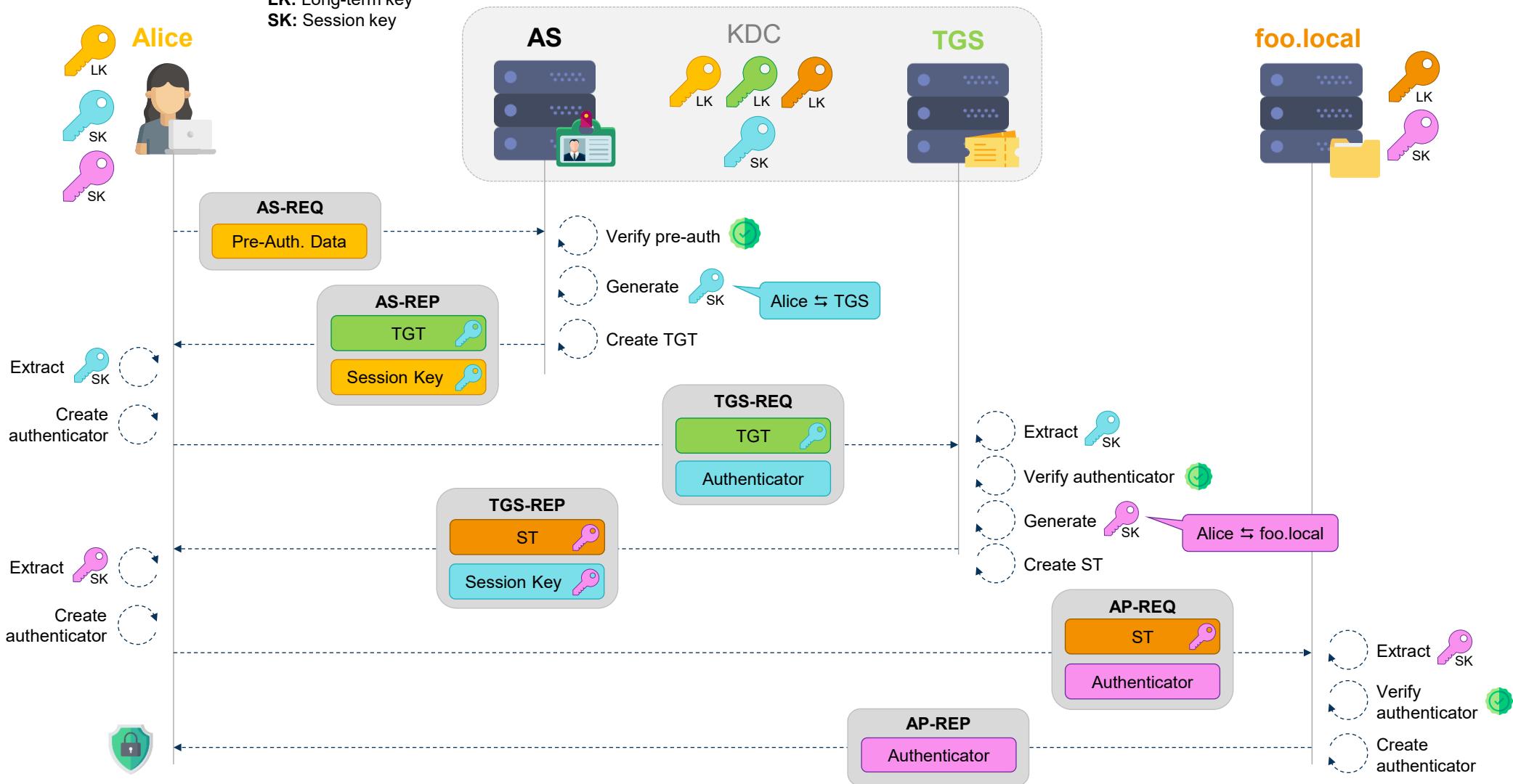
# Details: AP-REP in Wireshark – SMB Example

```
> SMB2 Header
└ Session Setup Response (0x01)
  [Preatuh Hash: fa17673f38b74239a901d0bcac3758bda9801d84f16620e6306bddc41c2436fac]
    > StructureSize: 0x0009
    > Session Flags: 0x0000
    Blob Offset: 0x00000048
    Blob Length: 185
  └ Security Blob: a181b63081b3a0030a0100a10b06092a864882f712010202a2819e04819b60819
    └ GSS-API Generic Security Service Application Program Interface
      └ Simple Protected Negotiation
        └ negTokenTarg
          negResult: accept-completed (0)
          supportedMech: 1.2.840.48018.1.2.2 (MS KRB5 - Microsoft Kerberos 5)
          responseToken: 60819806092a864886f71201020202006f8188308185a003020105
        └ krb5_blob: 60819806092a864886f71201020202006f8188308185a003020105a103
          KRB5 OID: 1.2.840.113554.1.2.2 (KRB5 - Kerberos 5)
          krb5_tok_id: KRB5_AP REP (0x0002)
        └ Kerberos
          └ ap-rep
            pwno: 5
            msg-type: krb-ap-rep (15)
        └ enc-part
          etype: eTYPE-AES256-CTS-HMAC-SHA1-96 (18)
          cipher: 251c3f0fae5585bd3909610209af4b5a8995137c81af9407b4
```

→ AP-REP message

→ Service authenticator  
(encrypted with service session key)

LK: Long-term key  
SK: Session key



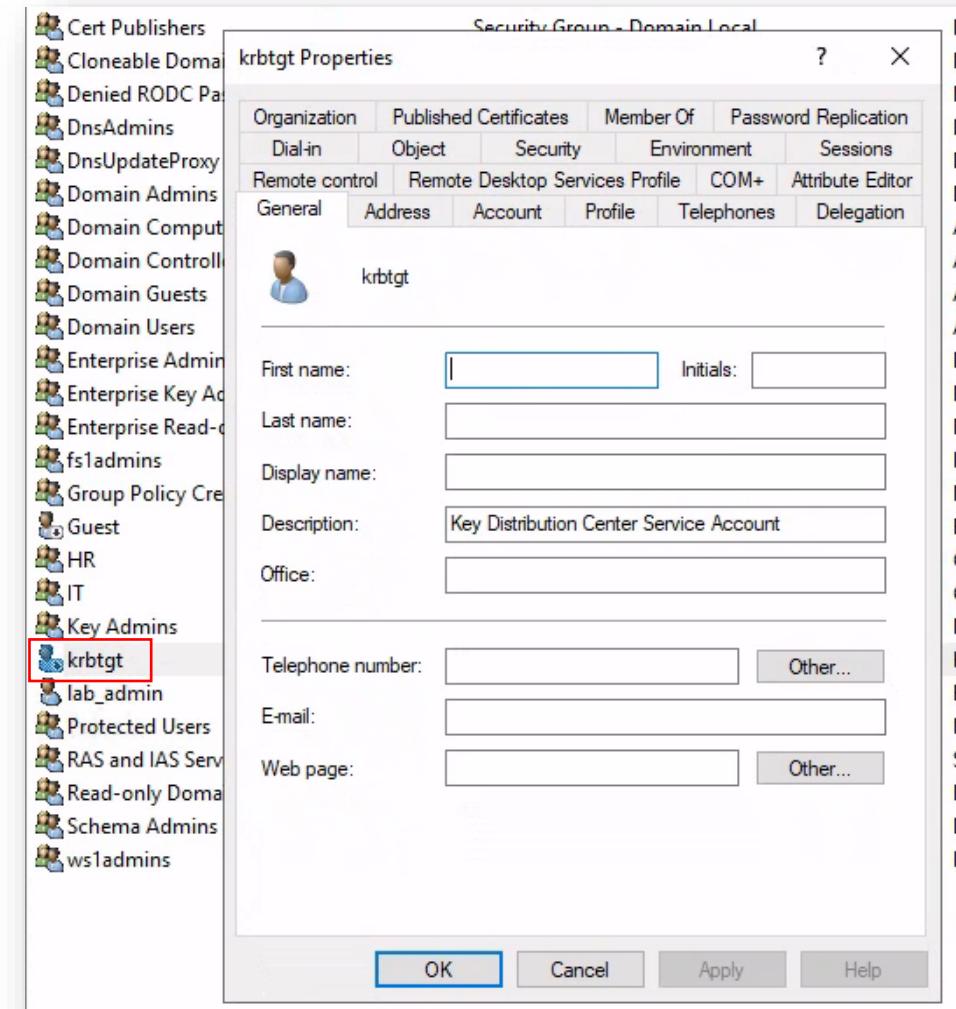
## Details – Authorization Data

- 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](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](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)

