# User Authentication and Access Control and wireless security and firewall

Instructor: Dr. Muhammad Umar Aftab
Credits and Prepared By: Miss Kanwal

# Remote User-Authentication Principles

What exactly is user authentication?

It's a two-step process:

**Identification:** This is you claiming an identity. You walk up to a system and say, 'Hi, I'm Alice.' You're presenting your username or User ID. This part is usually not a secret.

**Verification:** This is you proving you are really Alice. You do this by providing a secret(your password). The system checks this secret against what it has on file.

So, in our example:

- “ABTOKLAS” is Alice's public Identifier.
- Her “password” is the secret Authentication Information.

- If both match, the system trusts that you are who you claim to be and can then set your permissions and track your activity.

## NIST Model for Electronic User Authentication

- A standard framework for user authentication process, defined by the National Institute of Standards and Technology (NIST).
- According to the SP 800-63-2 Architectural Model. There are  different actor involve
- **Registration Authority (RA)**:  This is the trusted office that checks your physical ID (like a passport) and vouches for you. They do the initial 'identity proofing'.

- **Credential Service Provider (CSP)**:  This is the entity that creates and issues your electronic credential—like a digital ID card. In many systems, the RA and CSP are part of the same organization.

- **Subscriber/Claimant**: User being authenticated
- **Verifier**: When you later try to log in, the verifier is the software that challenges you for your password or token.

- **Relying Party (RP)**:  This is the final application or service you want to access (like your email server). It trusts the Verifier's decision.
- **Process**: Registration → Credential Issuance → Authentication → Authorization

(So the flow is: Get registered → Get your digital credential → Use it to prove who you are → Access services.)

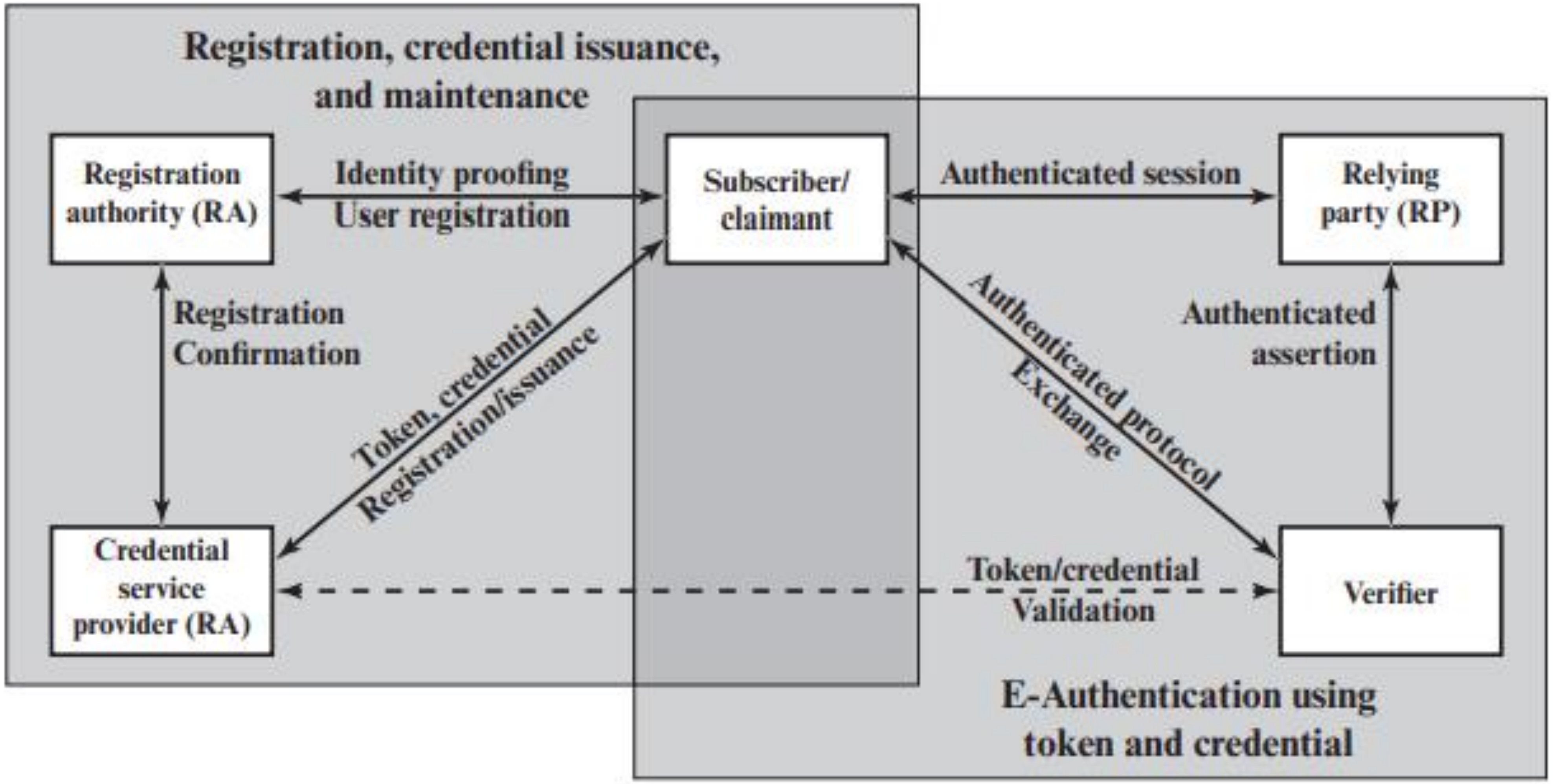

Registration, credential issuance, and maintenance
Registration authority (RA)
Identity proofing
User registration
Subscriber/ claimant
Authenticated session
Relying party (RP)
Registration Confirmation
Token, credential Registration/issuance
Authenticated protocol Exchange
Authenticated assertion
Credential service provider (RA)
Token/credential Validation
Verifier
E-Authentication using token and credential

**Figure 15.1** The NIST SP 800-63-2 E-Authentication Architectural Model

## Means of Authentication

How can you prove your identity? There are four general categories, which we call 'factors of authentication':

- **Something you know:** This is the most common—a password, a PIN, or answers to security questions.
- **Something you possess:** A physical object you have, like a smart card, a security token that generates codes, or even your physical house key.
- **Something you are (Static Biometrics):** This uses your unique physical characteristics, like your fingerprint, your retina pattern, or your face.
- **Something you do (Dynamic Biometrics):** This uses behavioral patterns, like your voice rhythm, your handwriting, or even your unique typing rhythm.
- **Multi-factor authentication** combines these for stronger security. For logging into a network, the most common combination is **something you know** (a password) used with **something you possess** (a cryptographic key).

## Mutual Authentication:

Often, it's not enough for just the server to verify you. You might also want to verify that you're talking to the *real* server and not a fake one set up by a hacker. This two-way verification is called **Mutual Authentication**.

The main goals here are to confirm each other's identity. Securely exchange a **session key** that they can use to encrypt the rest of their conversation. The biggest threat to this process is a **Replay Attack**. Imagine an eavesdropper records your login sequence and just plays it back later to impersonate you.

There are different types of replay attacks:

- ➢ Type 1: Just copy and replay the message later.
- ➢ Type 2 & 3: Replay a message while its timestamp is still valid, possibly while blocking the original message.
- ➢ Type 4: Send your own message back to you, which can cause confusion.

## Countering Replay Attacks

So, how do we stop these replay attacks? We have two main weapons:

- **Timestamps:** Every message gets a 'born-on' date. If a message arrives too late, it's rejected. Simple, right? The problem is **it requires all computers involved to have perfectly synchronized clocks**. If clocks drift apart, it causes big problems.

- **Challenge/Response:** This is like a secret handshake.
  - For Example: You (Party A) say, "Prove you're real by answering this random number I just made up: **57**."
  - The server (Party B) has to take that number, encrypt it with your shared secret key, and send the encrypted result back.
  - You decrypt it, and if it gives you back '57', you know it's the real server.
- Challenge/Response is very secure but requires extra back-and-forth messages (a 'handshake'), which isn't ideal for all situations.

## One-Way Authentication

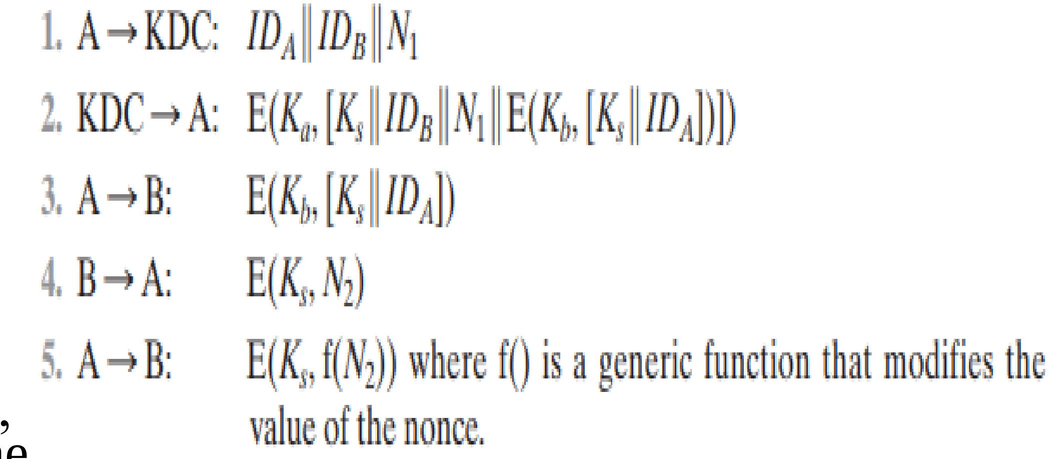
- Not all authentication needs to be mutual. The classic example is **email**. When you send an email, you and the recipient aren't online at the same time.This creates special requirements:
- The **envelope** (the 'To:' and 'From:' addresses) must be readable by the email system so it can route the message.
- But the **body** of the message should be encrypted so the mail servers can't read it.
- The recipient also wants **assurance that the email actually came from you**.
- So, we need a way to authenticate the sender without requiring the recipient to be online for a complex handshake

## Remote Authentication Using Symmetric Encryption

- Now, let's talk about how to do this authentication practically using symmetric encryption, where everyone shares a secret key.
- Managing secret keys between every possible pair of users is a nightmare. The solution is to use a trusted middleman called a **Key Distribution Center (KDC)**.Here's how it works:
- You and every other user have a single, long-term **master key** that you share *only* with the KDC.
- When you want to talk to someone else, you ask the KDC for a temporary **session key**.
- The KDC generates this session key and sends it to both of you, but it's encrypted *inside* a package using your respective master keys.
- This way, you only have to protect one master key, and the KDC handles the rest.

# Needham-Schroeder Protocol

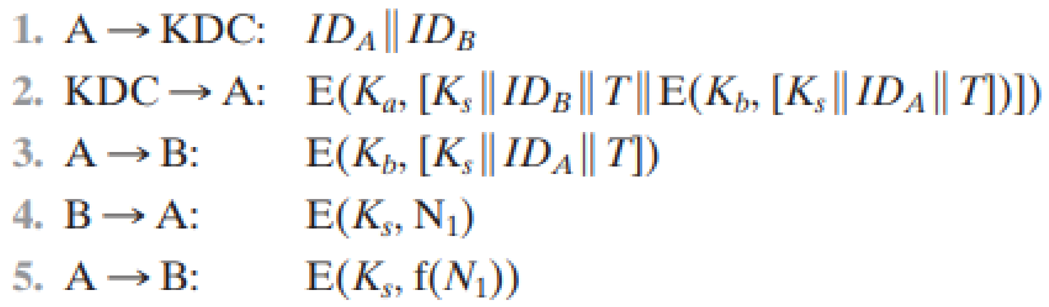
- Alice tells the KDC: "I'm Alice, I want to talk to Bob, here's a random number $N_1$."
- The KDC sends Alice a package containing a new session key (K_s). Part of this package is a **ticket for Bob**, encrypted with Bob's master key. This ticket also contains the session key.
- Alice forwards this ticket to Bob. Bob can open it because it's encrypted with *his* key, and he finds the session key inside.
- Bob sends Alice a random number $N_2$, encrypted with the new session key, to prove he got the key.
- Alice performs a simple operation on $N_2$ (like adding 1), encrypts it, and sends it back to prove *she* also knows the key.
- This 'handshake' in steps 4 and 5 prevents a simple replay of step 3. However, if an attacker ever stole an *old* session key, they could impersonate Alice. This is a vulnerability.
- **Vulnerability**: Susceptible to replay attacks if old session keys are compromised



1. $A \rightarrow KDC$: $ID_A \| ID_B \| N_1$
2. $KDC \rightarrow A$: $E(K_a, [K_s \| ID_B \| N_1 \| E(K_b, [K_s \| ID_A])])$
3. $A \rightarrow B$: $E(K_b, [K_s \| ID_A])$
4. $B \rightarrow A$: $E(K_s, N_2)$
5. $A \rightarrow B$: $E(K_s, f(N_2))$ where f() is a generic function that modifies the value of the nonce.

# Denning Protocol Improvement

- To fix the problem in Needham-Schroeder, Dorothy Denning added timestamps.
- The main change is that the KDC now puts a timestamp T inside the tickets. Now, when Bob receives the ticket from Alice, he checks the timestamp. If it's too old, he rejects it.
- Advantage: Even if an attacker steals an old session key, the timestamp on the ticket will have expired, making it useless.
- Disadvantage: We're back to the problem of requiring synchronized clocks across the network



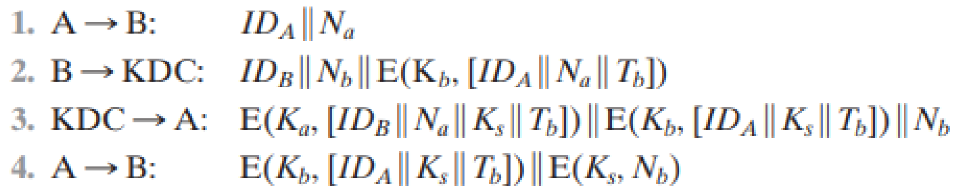
| | |
|---|---|
| 1. $A \rightarrow KDC$: | $ID_A \| ID_B$ |
| 2. $KDC \rightarrow A$: | $E(K_a, [K_s \| ID_B \| T \| E(K_b, [K_s \| ID_A \| T])])$ |
| 3. $A \rightarrow B$: | $E(K_b, [K_s \| ID_A \| T])$ |
| 4. $B \rightarrow A$: | $E(K_s, N_1)$ |
| 5. $A \rightarrow B$: | $E(K_s, f(N_1))$ |

## Suppress-Replay Attacks

- But timestamps introduce their own new, clever attack called a Suppress-Replay Attack.
- Imagine Alice's clock is 5 minutes ahead of Bob's.
- Alice sends a request with her timestamp, which is 5 minutes in Bob's future.
- An attacker, Eve, intercepts and holds onto this message.
- 5 minutes later, Bob's clock 'catches up' to the timestamp.
- Eve now replays the message. Bob sees a fresh, valid timestamp and accepts it!
- This is a tricky problem. The solutions are either to constantly synchronize clocks very carefully or to go back to using nonces (random challenges) which are not vulnerable to this timing trick.

# Enhanced Authentication Protocol

- Researchers kept refining the protocols to be more secure. This is one of the improved versions.
- It's more complex, but notice a few key things:
  1. It uses nonces (N-a, N-b) from both Alice and Bob, so it doesn't rely solely on timestamps.
  2. The KDC gives Alice a reusable 'ticket' to talk to Bob. She can keep this ticket and use it for a certain period without going back to the KDC.
  3. This ticket is encrypted with Bob's key, so Alice can't tamper with it.
  4. This protocol is a nice balance, providing strong security against replays while also being efficient for multiple connections.



1. $A \rightarrow B$: $ID_A \| N_a$
2. $B \rightarrow KDC$: $ID_B \| N_b \| E(K_b, [ID_A \| N_a \| T_b])$
3. $KDC \rightarrow A$: $E(K_a, [ID_B \| N_a \| K_s \| T_b]) \| E(K_b, [ID_A \| K_s \| T_b]) \| N_b$
4. $A \rightarrow B$: $E(K_b, [ID_A \| K_s \| T_b]) \| E(K_s, N_b)$

## Subsequent Session Protocol

- What happens when Alice wants to talk to Bob again, and her ticket is still valid?
- She doesn't need to bother the KDC again! She can just use the protocol on this slide:
- She sends the old ticket to Bob, plus a new random number.
- Bob responds with his own new random number.
- They complete a quick handshake.
- This is fast and efficient, and because they use new nonces each time, it's still secure against replays for this new session.

1. $\mathrm{A} \rightarrow \mathrm{B}$: $\mathrm{E}(K_b, [ID_A \| K_s \| T_b]) \| N'_a$
2. $\mathrm{B} \rightarrow \mathrm{A}$: $N'_b \| \mathrm{E}(K_s, N'_a)$
3. $\mathrm{A} \rightarrow \mathrm{B}$: $\mathrm{E}(K_s, N'_b)$

## One-Way Authentication with Symmetric Encryption

- Let's go back to our email example. How can we do one-way authentication with a KDC?
- The protocol is similar to the first part of Needham-Schroeder, but we remove the mutual handshake. Alice gets a ticket for Bob and the session key from the KDC. She then sends Bob two things:
- The ticket (which Bob can decrypt).
- The actual email message, encrypted with the session key.
- Bob uses his master key to open the ticket, gets the session key, and then uses that to decrypt the email. This proves the message came from someone who got a valid ticket from the KDC (i.e., Alice). It's not as strong as mutual authentication, but it works for email.

## Kerberos:

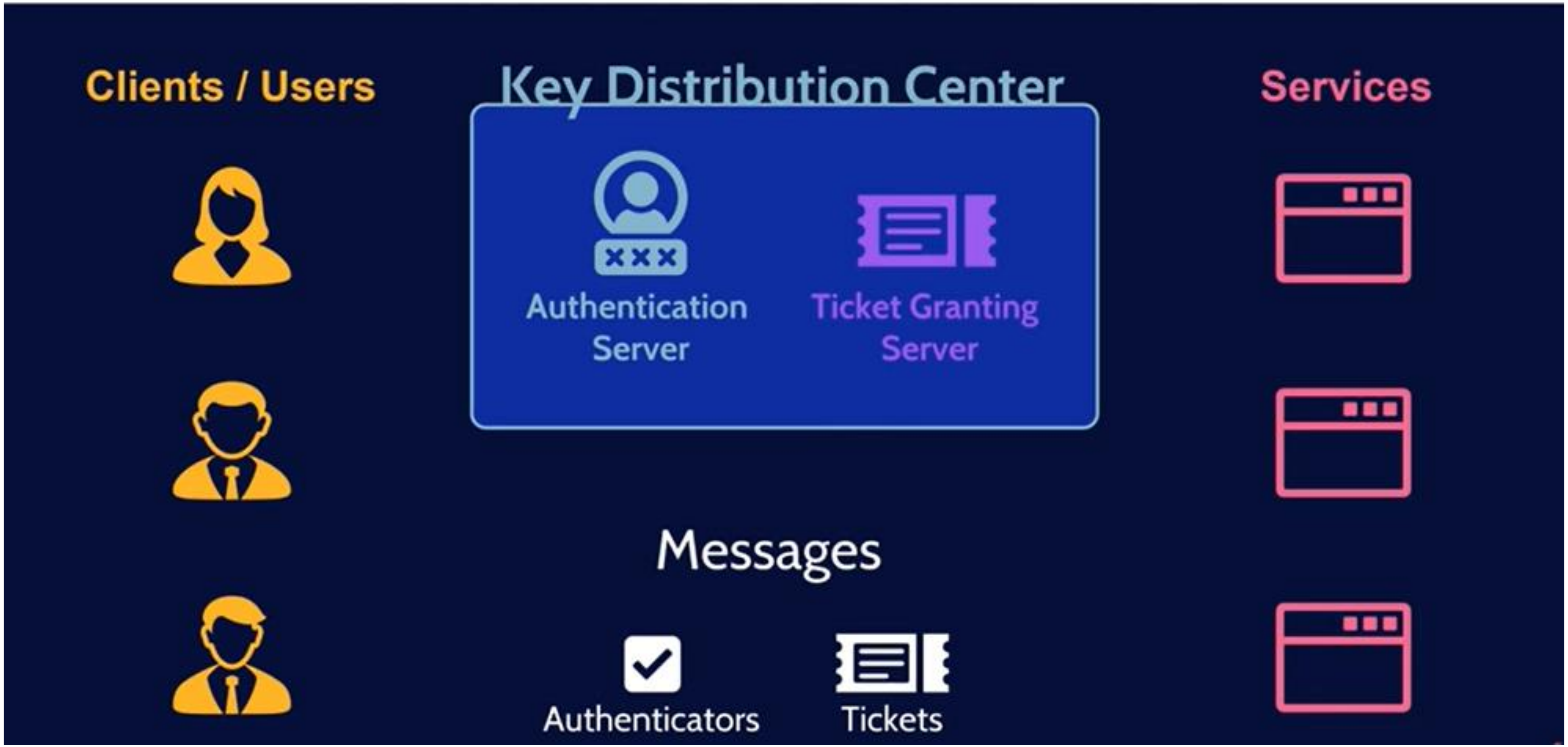
- A trusted third-party authentication service.
- Developed at MIT as part of Project Athena.
- Relies exclusively on symmetric encryption (no public-key crypto).
- Provide secure mutual authentication between users and servers in a network.
- Uses a protocol based on Needham-Schroeder with enhancements to prevent replay attacks.

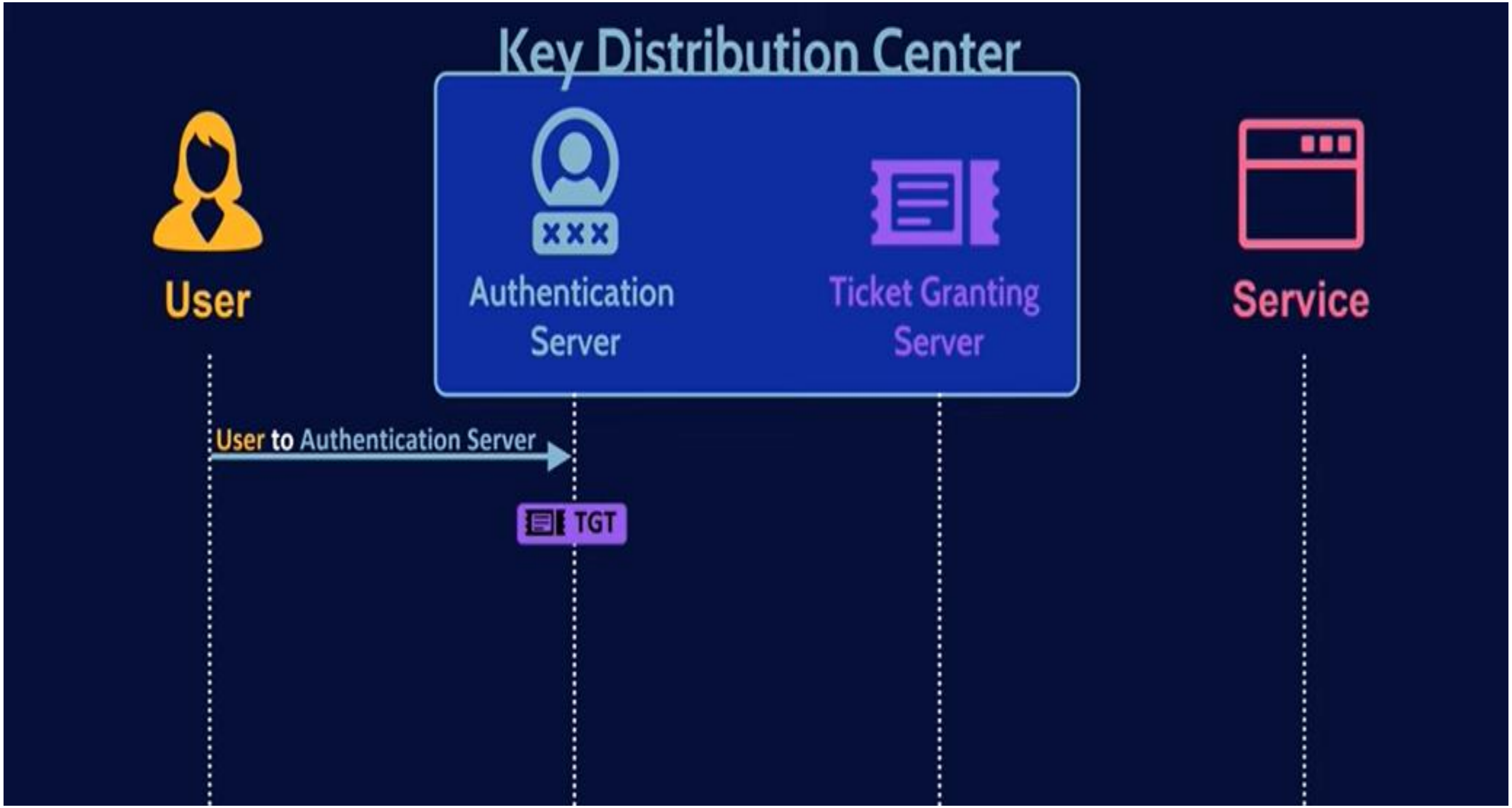
## Kerberos Requirements:

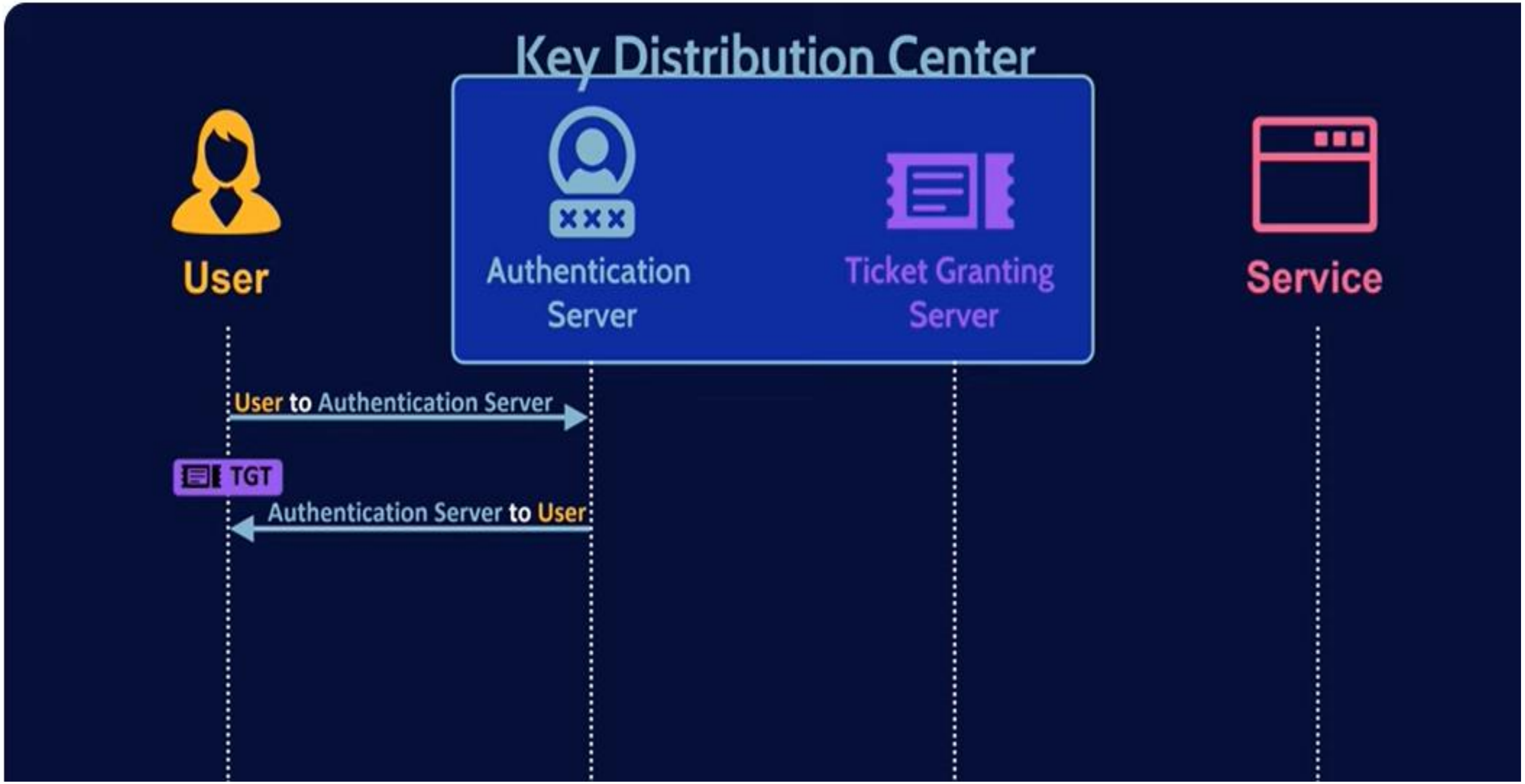
- **Secure:** A network eavesdropper cannot obtain information to impersonate a user.
- **Reliable:** Highly available using a distributed server architecture.
- **Transparent:** Users are unaware of authentication beyond entering a password.
- **Scalable:** Supports large numbers of clients and servers with a modular design.

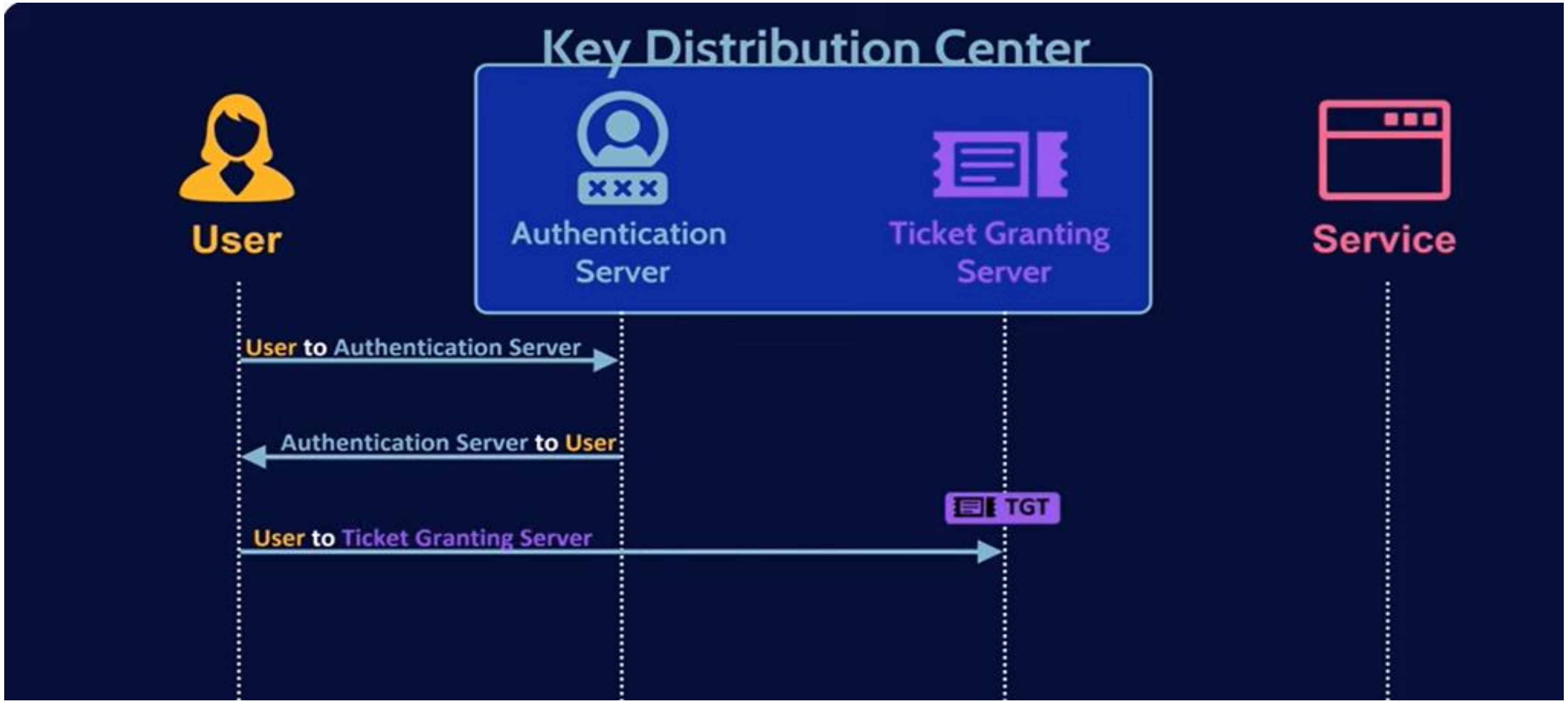
## Kerberos Key Components

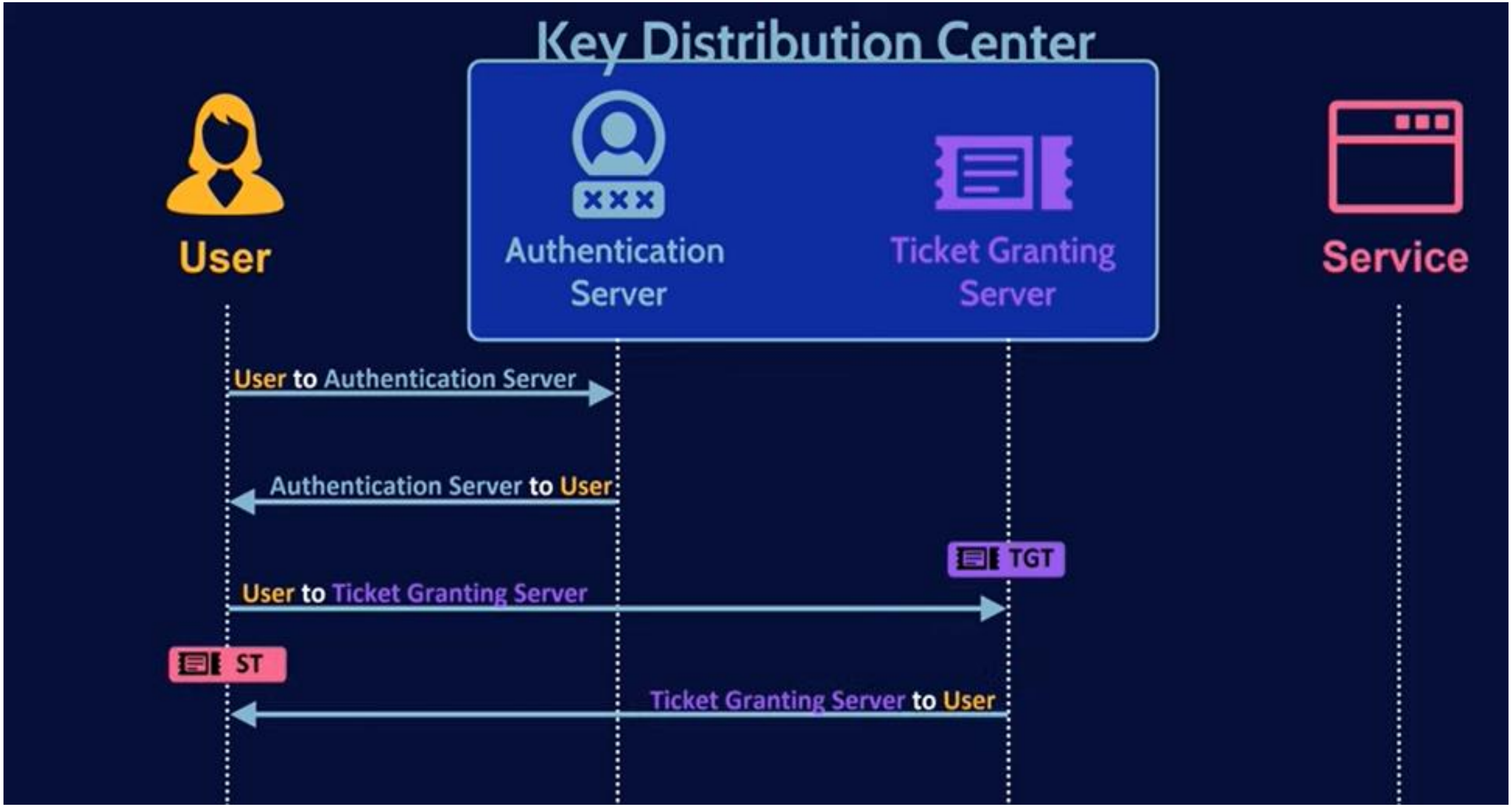
- **Authentication Server (AS):**
  - Verifies user identity at login.
  - Issues Ticket-Granting Tickets (TGTs).
- **Ticket-Granting Server (TGS):**
  - Issues service-specific tickets.
  - Acts as a trusted intermediary.
- **Ticket:**
  - A reusable credential proving the user's identity to a service.
  - Encrypted with the server's secret key.
- **Authenticator:**
  - A one-time-use credential proving the user is the ticket owner.
  - Encrypted with a session key.

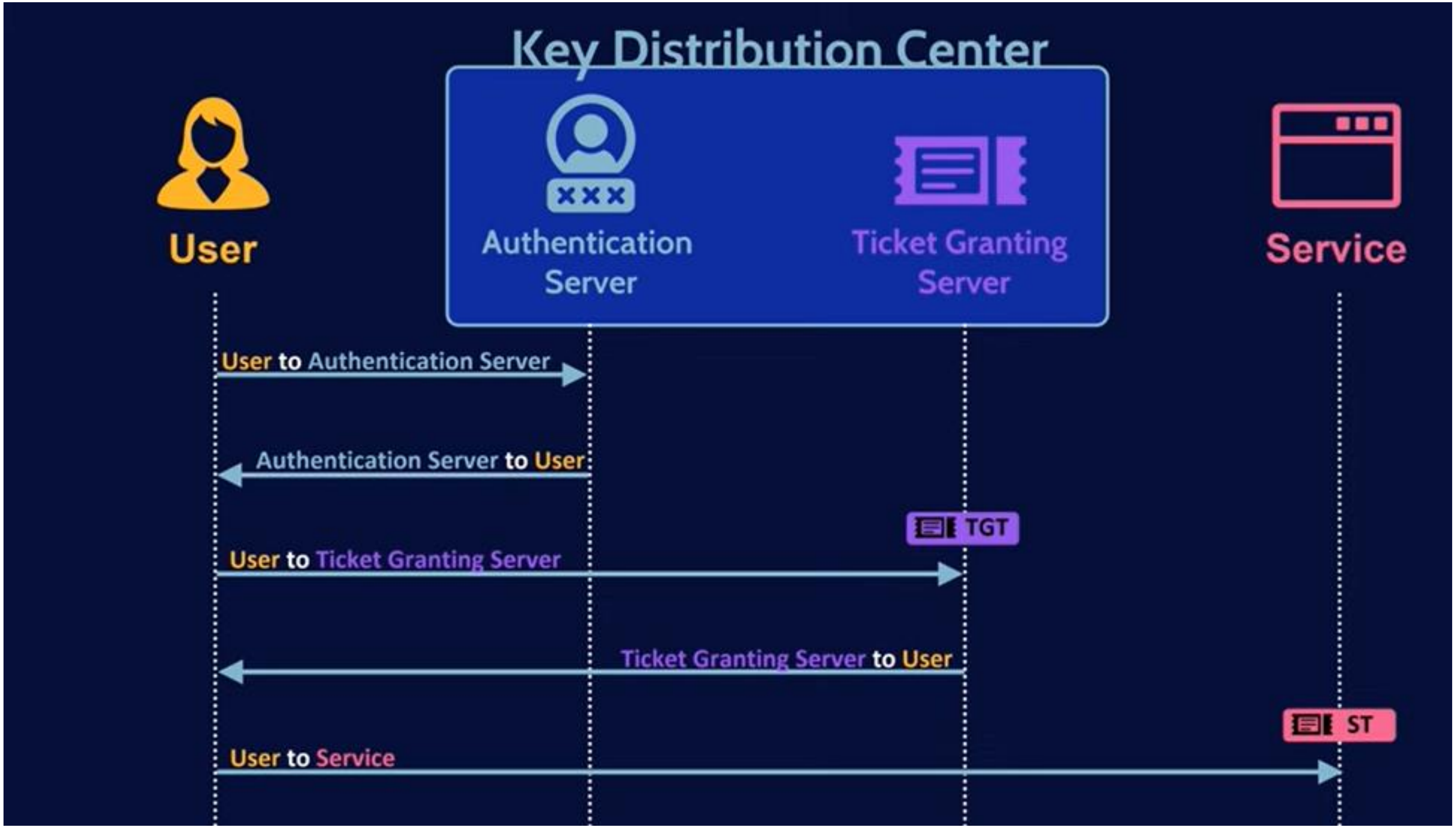

Clients / Users
Key Distribution Center
Services
Authentication Server
Ticket Granting Server
Messages
Authenticators
Tickets

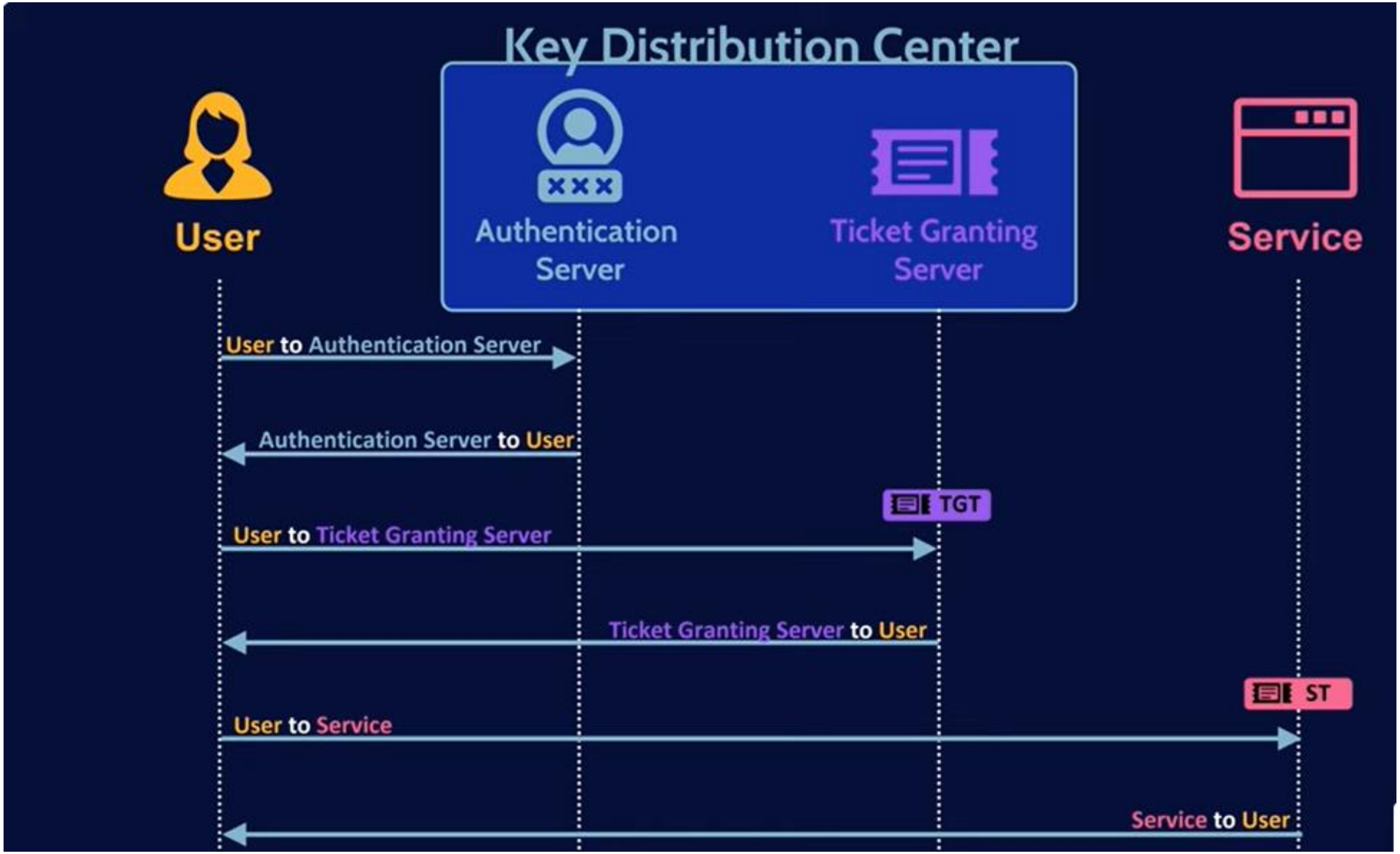

Key Distribution Center
User
Authentication Server
Ticket Granting Server
Service
User to Authentication Server
TGT

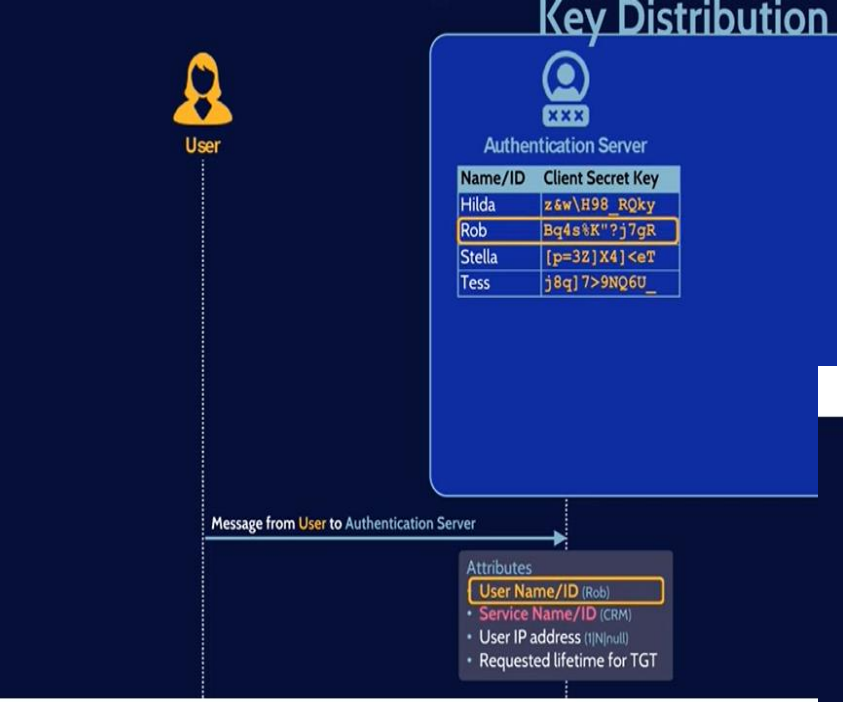

Key Distribution Center
User
Authentication Server
Ticket Granting Server
Service
User to Authentication Server
TGT
Authentication Server to User


Key Distribution Center
User
Authentication Server
Ticket Granting Server
Service
User to Authentication Server
Authentication Server to User
TGT
User to Ticket Granting Server


Key Distribution Center
User
Authentication Server
Ticket Granting Server
Service
User to Authentication Server
Authentication Server to User
TGT
User to Ticket Granting Server
ST
Ticket Granting Server to User

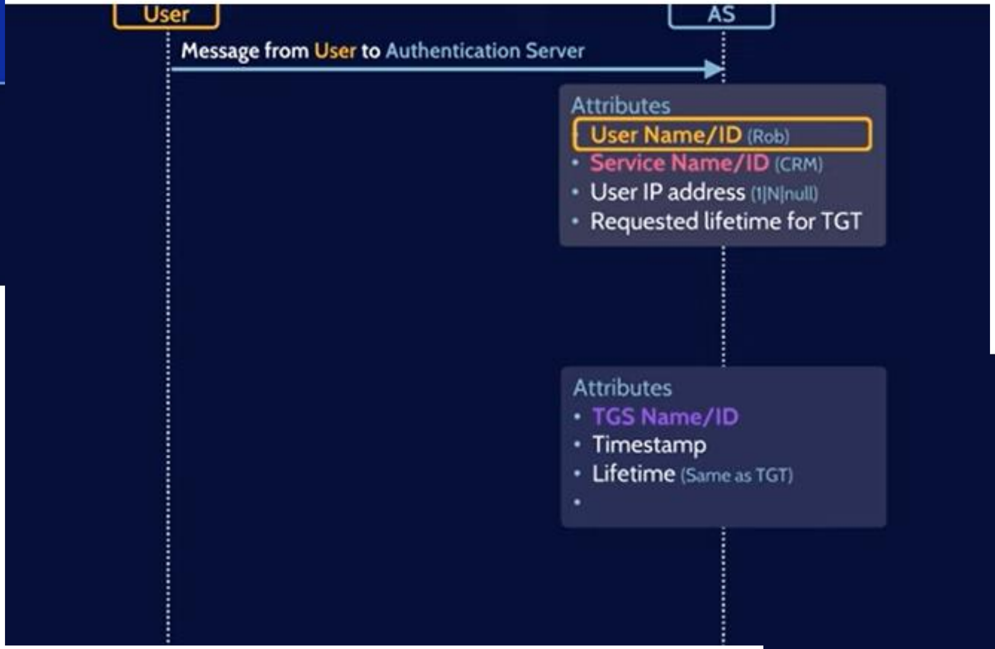

Key Distribution Center
User
Authentication Server
Ticket Granting Server
Service
User to Authentication Server
Authentication Server to User
TGT
User to Ticket Granting Server
Ticket Granting Server to User
ST
User to Service


Key Distribution Center
User
Authentication Server
Ticket Granting Server
Service
User to Authentication Server
Authentication Server to User
TGT
User to Ticket Granting Server
Ticket Granting Server to User
ST
User to Service
Service to User



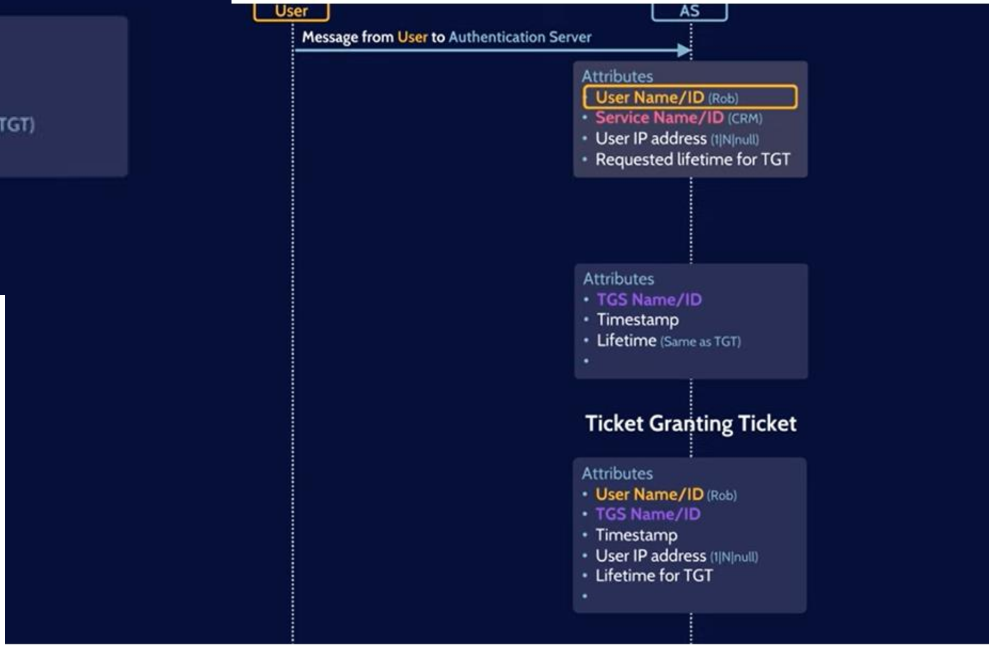
| Name/ID | Client Secret Key |
|---|---|
| Hilda | z&w\H98_RQky |
| Rob | Bq4s%K"?j7gR |
| Stella | [p=3Z]X4]<eT |
| Tess | j8q]7>9NQ6U_ |

Attributes
- User Name/ID (Rob)
- Service Name/ID (CRM)
- User IP address (1\|N\|null)
- Requested lifetime for TGT



Attributes
- User Name/ID (Rob)
- Service Name/ID (CRM)
- User IP address (1\|N\|null)
- Requested lifetime for TGT

Attributes
- TGS Name/ID
- Timestamp
- Lifetime (Same as TGT)
-

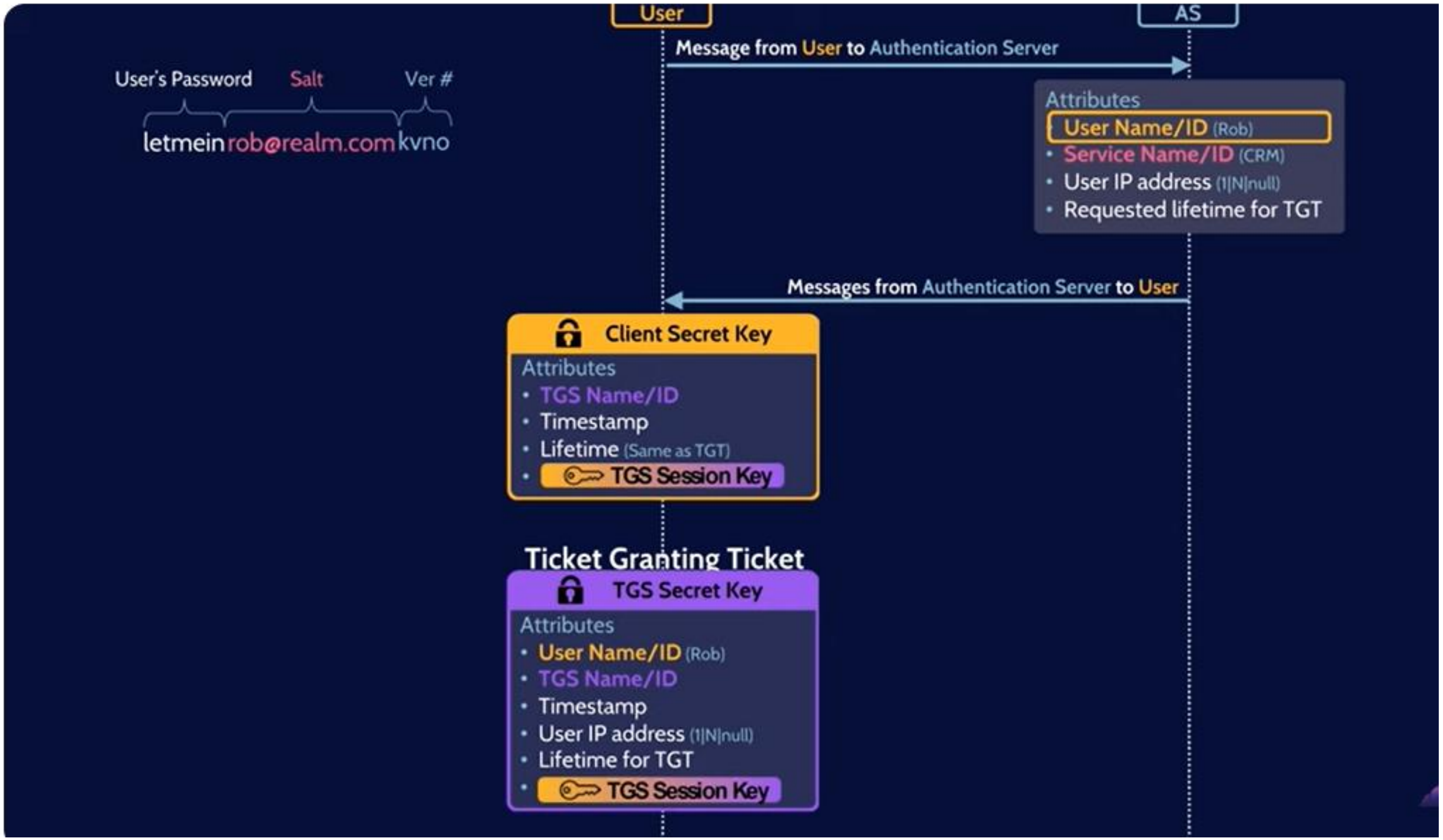


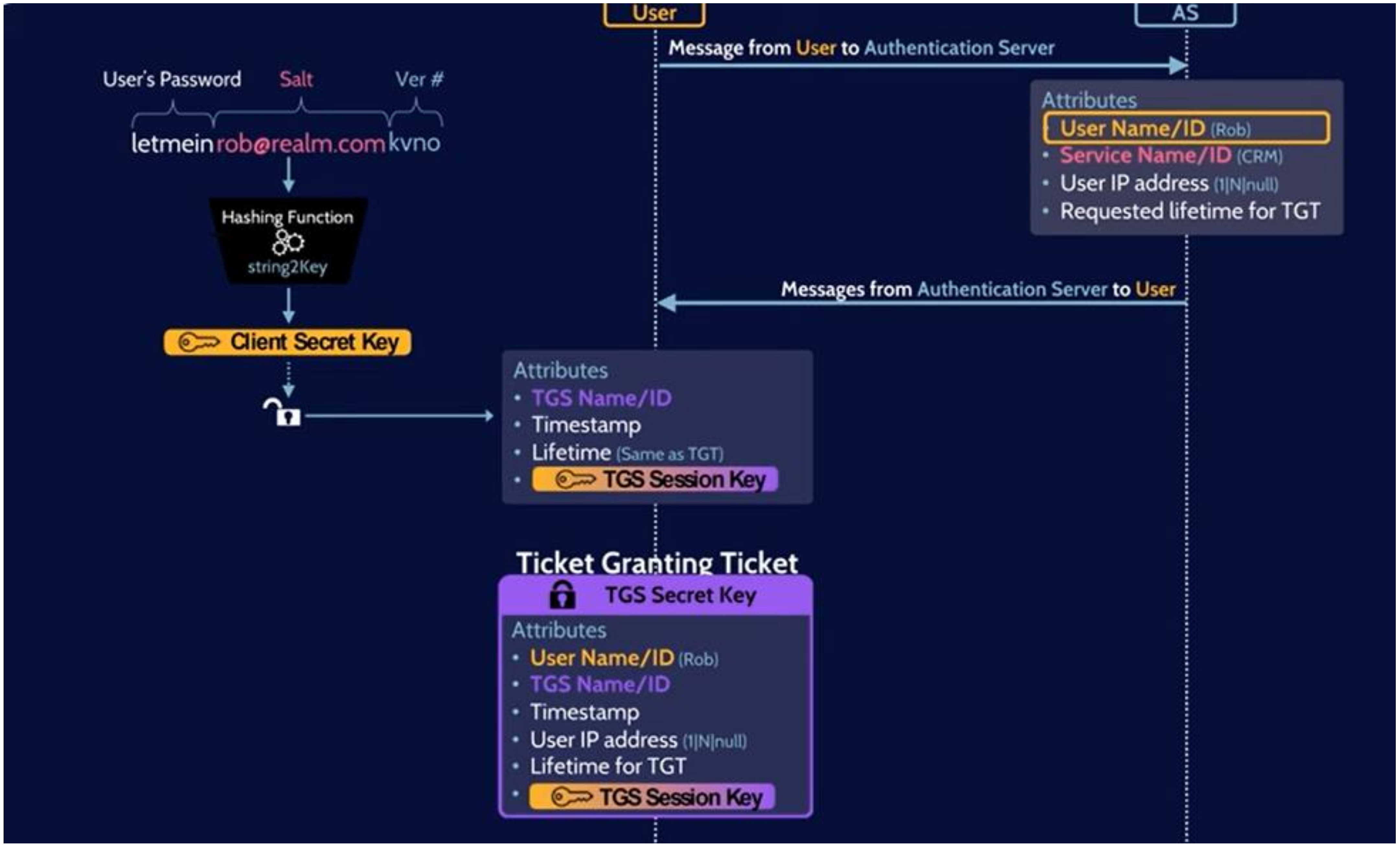
Attributes
- User Name/ID (Rob)
- Service Name/ID (CRM)
- User IP address (1\|N\|null)
- Requested lifetime for TGT

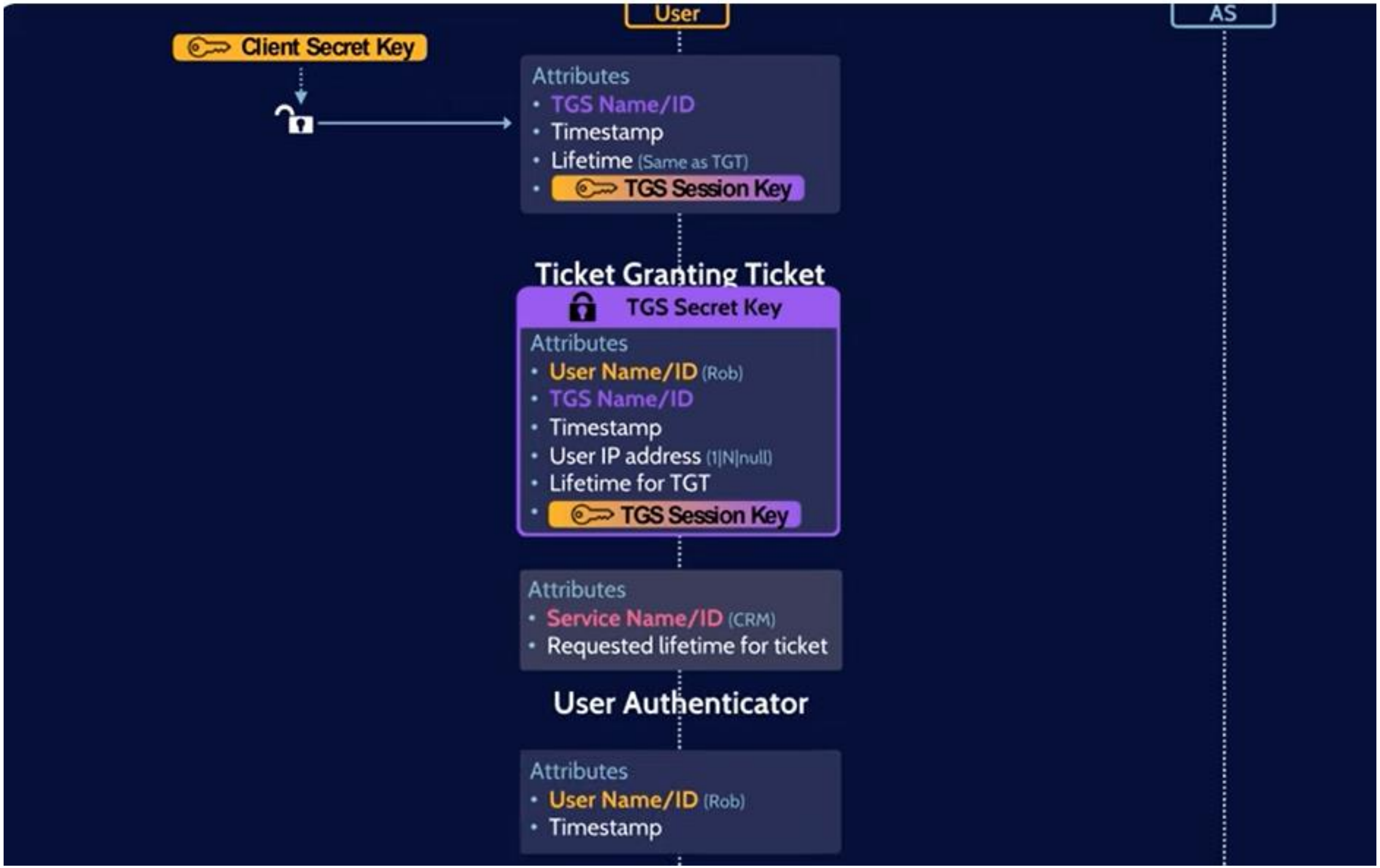
Attributes
- TGS Name/ID
- Timestamp
- Lifetime (Same as TGT)
-

Ticket Granting Ticket

Attributes
- User Name/ID (Rob)
- TGS Name/ID
- Timestamp
- User IP address (1\|N\|null)
- Lifetime for TGT
-


User
AS
User's Password
Salt
Ver #
letmein rob@realm.com kvno
Message from User to Authentication Server
Attributes
User Name/ID (Rob)
Service Name/ID (CRM)
User IP address (1|N|null)
Requested lifetime for TGT
Messages from Authentication Server to User
Client Secret Key
Attributes
TGS Name/ID
Timestamp
Lifetime (Same as TGT)
TGS Session Key
Ticket Granting Ticket
TGS Secret Key
Attributes
User Name/ID (Rob)
TGS Name/ID
Timestamp
User IP address (1|N|null)
Lifetime for TGT
TGS Session Key


User
AS
Message from User to Authentication Server
Attributes
User Name/ID (Rob)
Service Name/ID (CRM)
User IP address (1|N|null)
Requested lifetime for TGT
User's Password
Salt
Ver #
letmein rob@realm.com kvno
Hashing Function
string2Key
Client Secret Key
Messages from Authentication Server to User
Attributes
TGS Name/ID
Timestamp
Lifetime (Same as TGT)
TGS Session Key
Ticket Granting Ticket
TGS Secret Key
Attributes
User Name/ID (Rob)
TGS Name/ID
Timestamp
User IP address (1|N|null)
Lifetime for TGT
TGS Session Key


User
AS
Client Secret Key

Attributes
- TGS Name/ID
- Timestamp
- Lifetime (Same as TGT)
- TGS Session Key

Ticket Granting Ticket

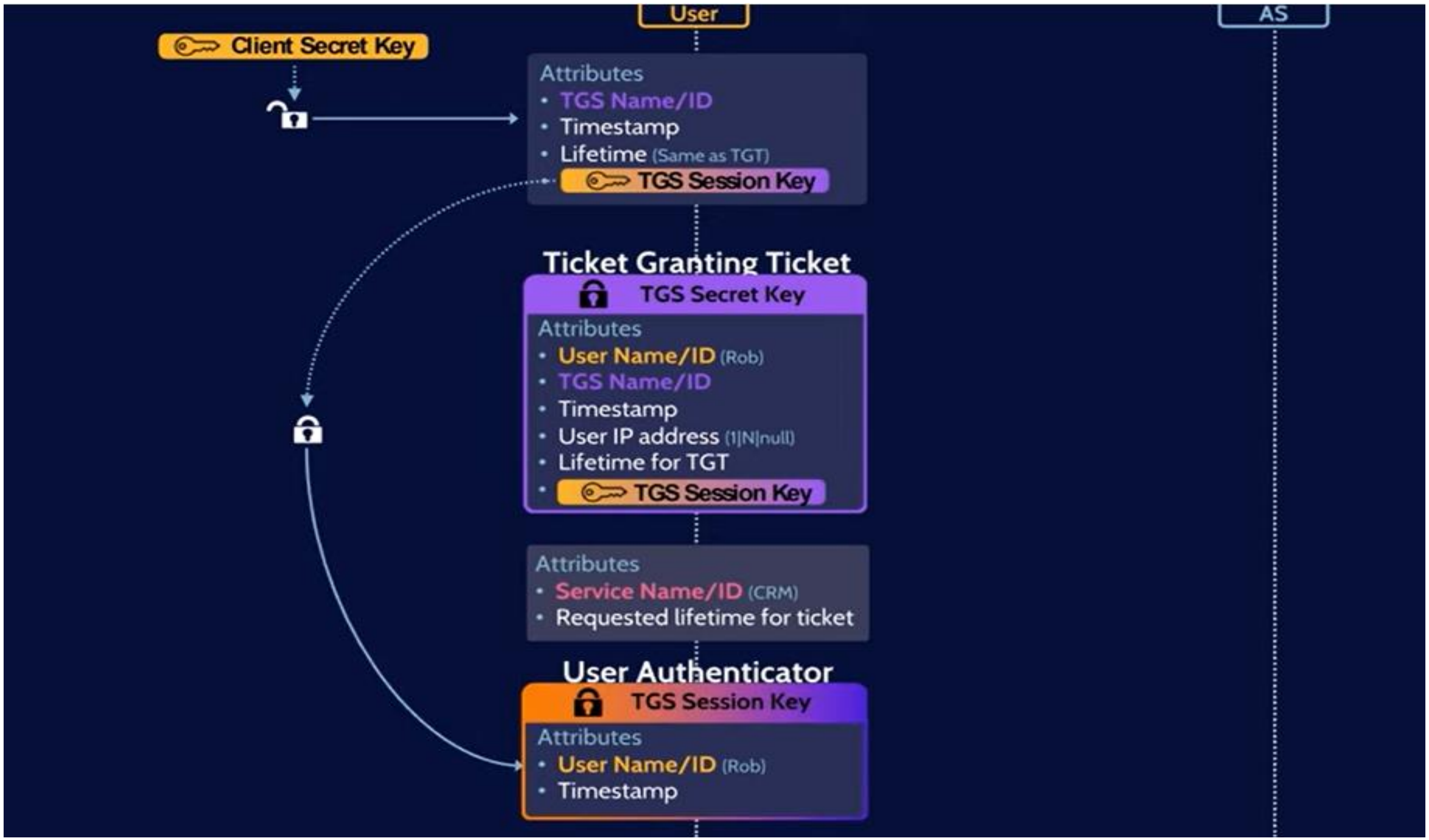
TGS Secret Key

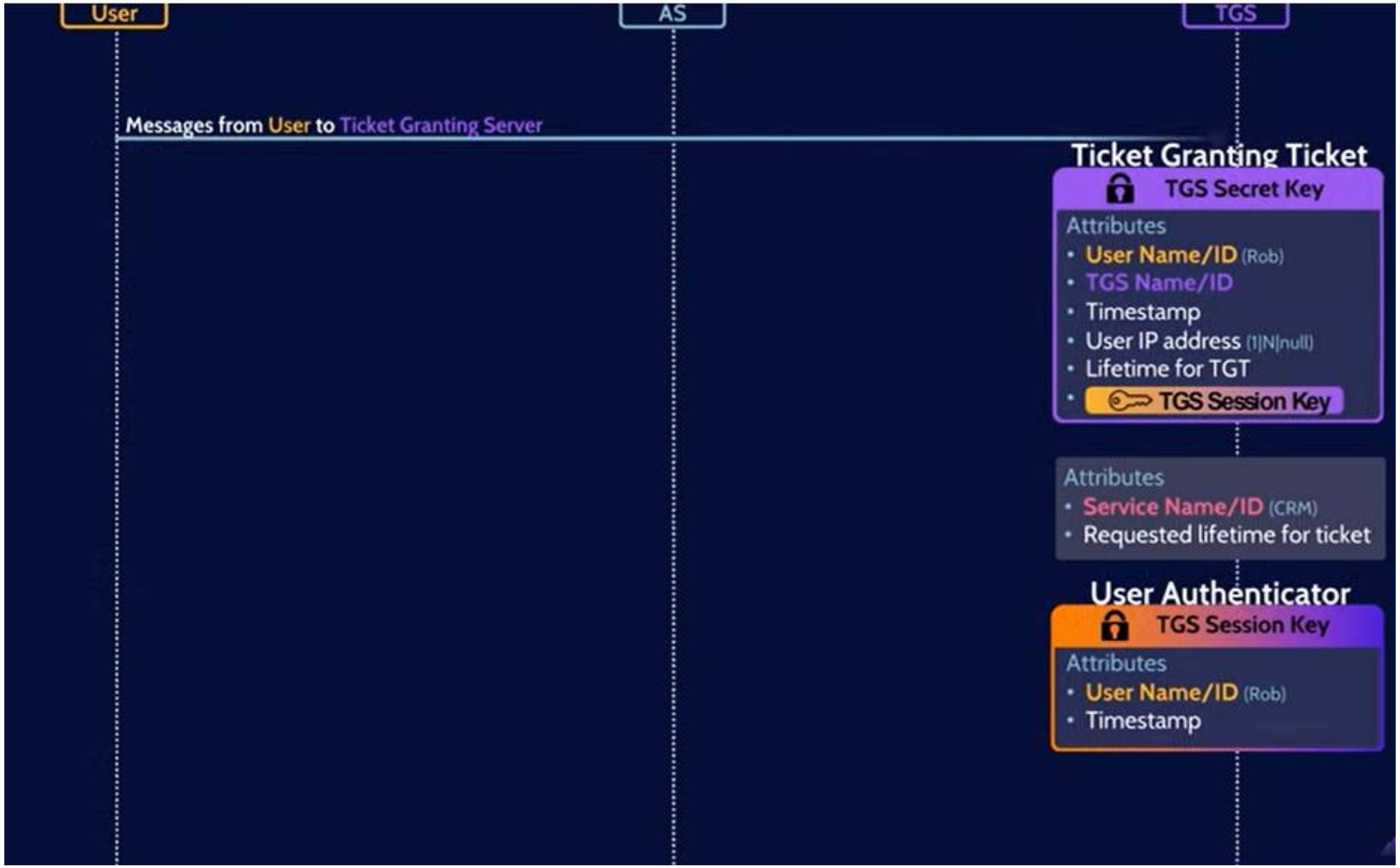
Attributes
- User Name/ID (Rob)
- TGS Name/ID
- Timestamp
- User IP address (1|N|null)
- Lifetime for TGT
- TGS Session Key

Attributes
- Service Name/ID (CRM)
- Requested lifetime for ticket

User Authenticator

Attributes
- User Name/ID (Rob)
- Timestamp


User
AS
Client Secret Key
Attributes
• TGS Name/ID
• Timestamp
• Lifetime (Same as TGT)
• TGS Session Key
Ticket Granting Ticket
TGS Secret Key
Attributes
• User Name/ID (Rob)
• TGS Name/ID
• Timestamp
• User IP address (1|N|null)
• Lifetime for TGT
• TGS Session Key
Attributes
• Service Name/ID (CRM)
• Requested lifetime for ticket
User Authenticator
TGS Session Key
Attributes
• User Name/ID (Rob)
• Timestamp



Messages from User to Ticket Granting Server

**Ticket Granting Ticket**

TGS Secret Key

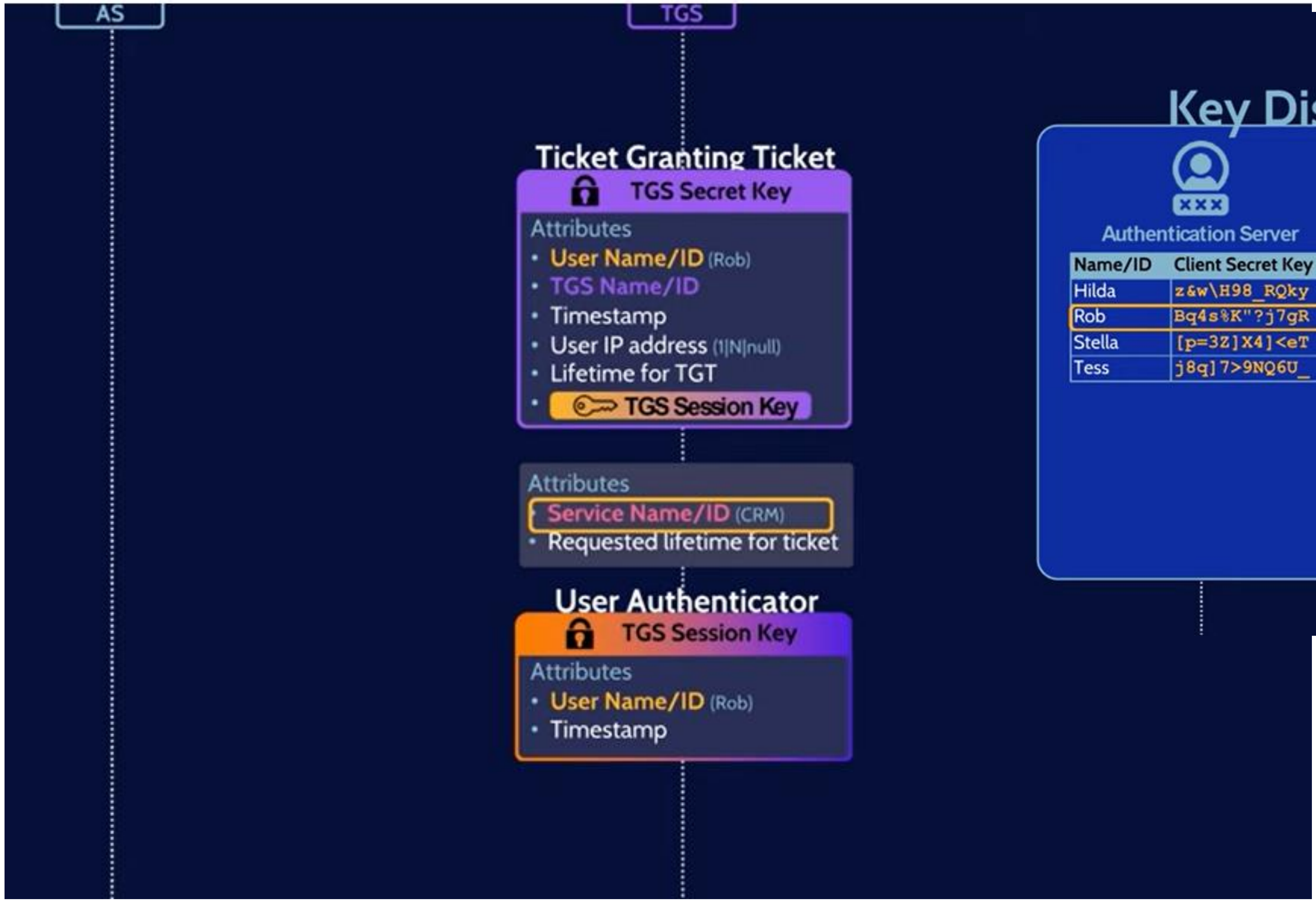
Attributes
- User Name/ID (Rob)
- TGS Name/ID
- Timestamp
- User IP address (1|N|null)
- Lifetime for TGT
- TGS Session Key

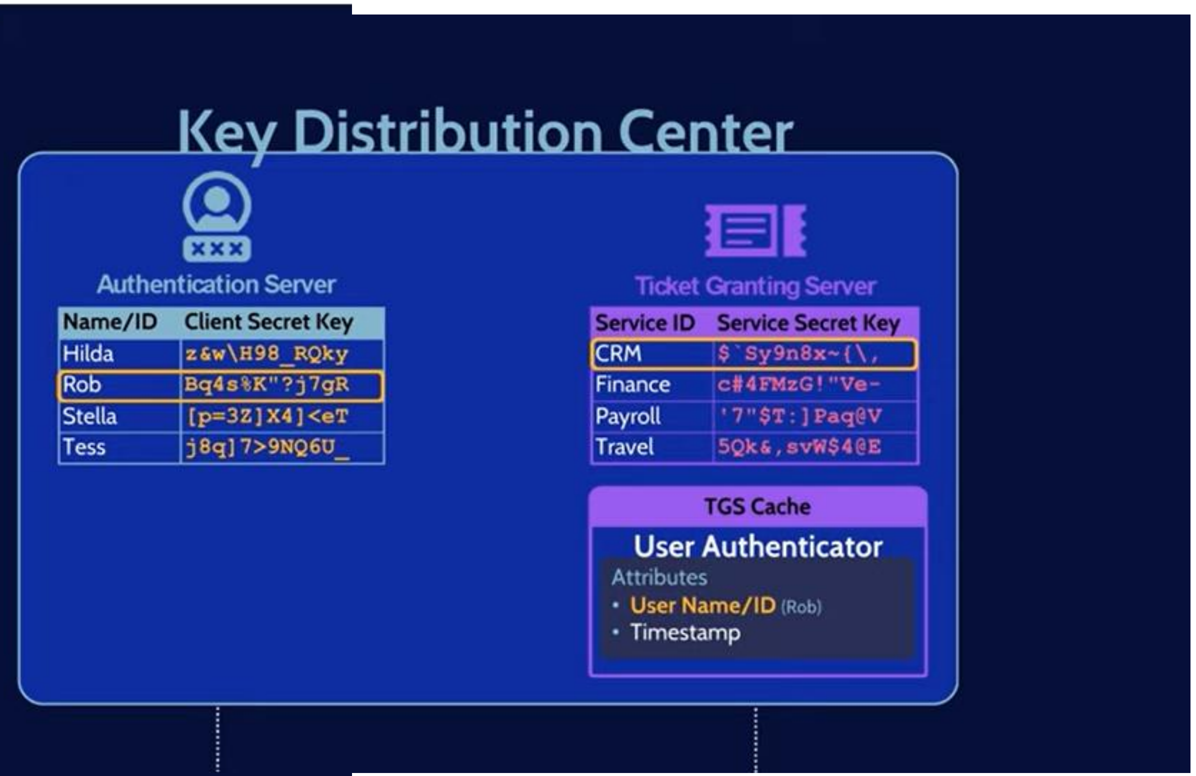
Attributes
- Service Name/ID (CRM)
- Requested lifetime for ticket

**User Authenticator**

TGS Session Key

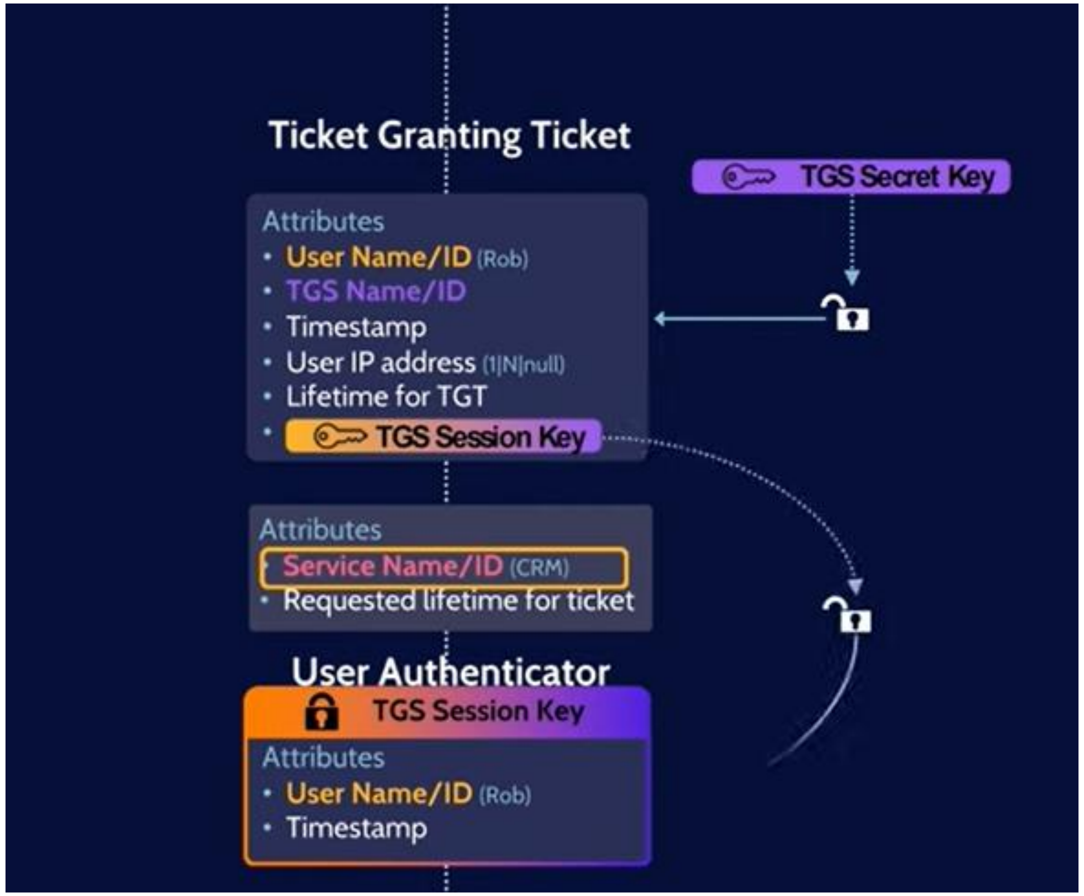
Attributes
- User Name/ID (Rob)
- Timestamp

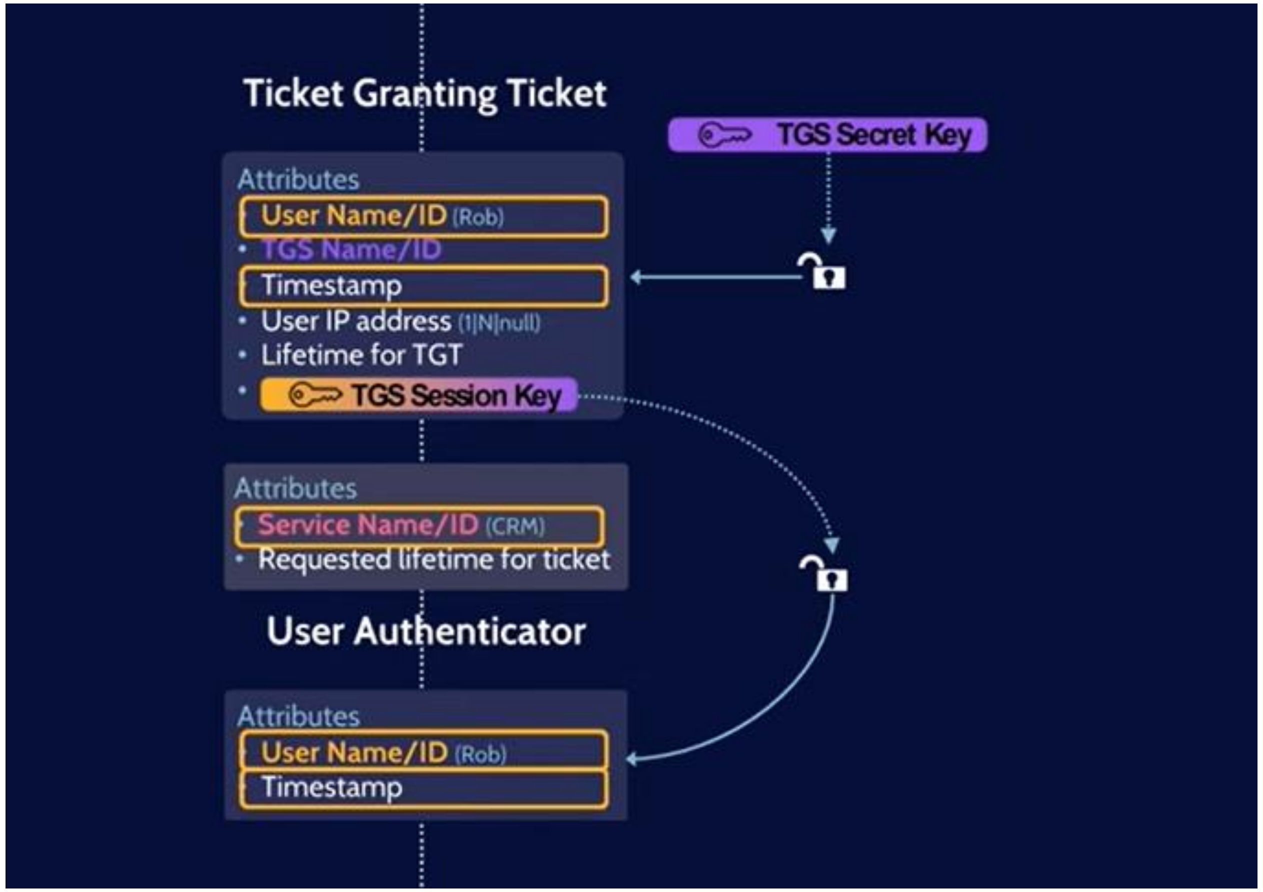


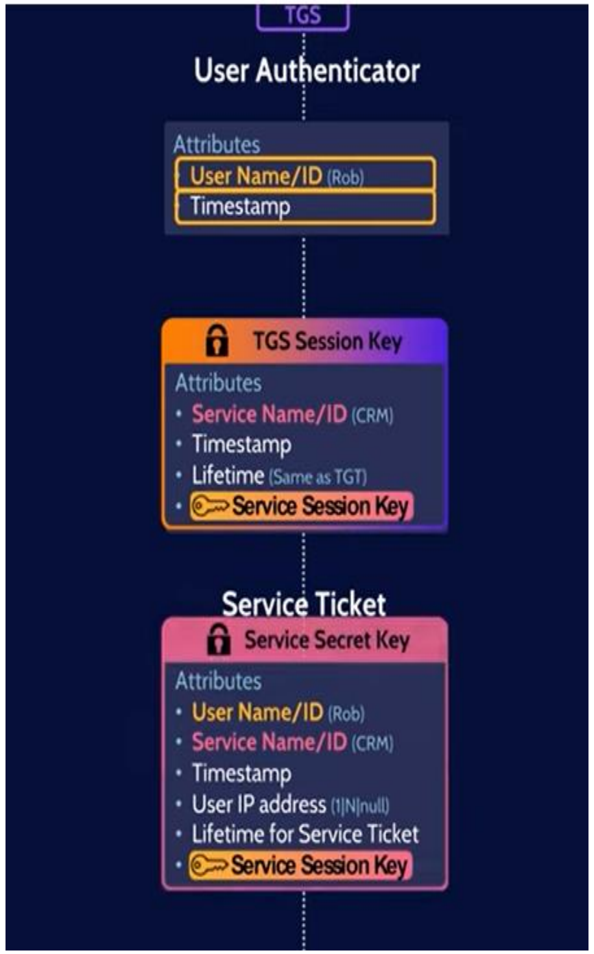


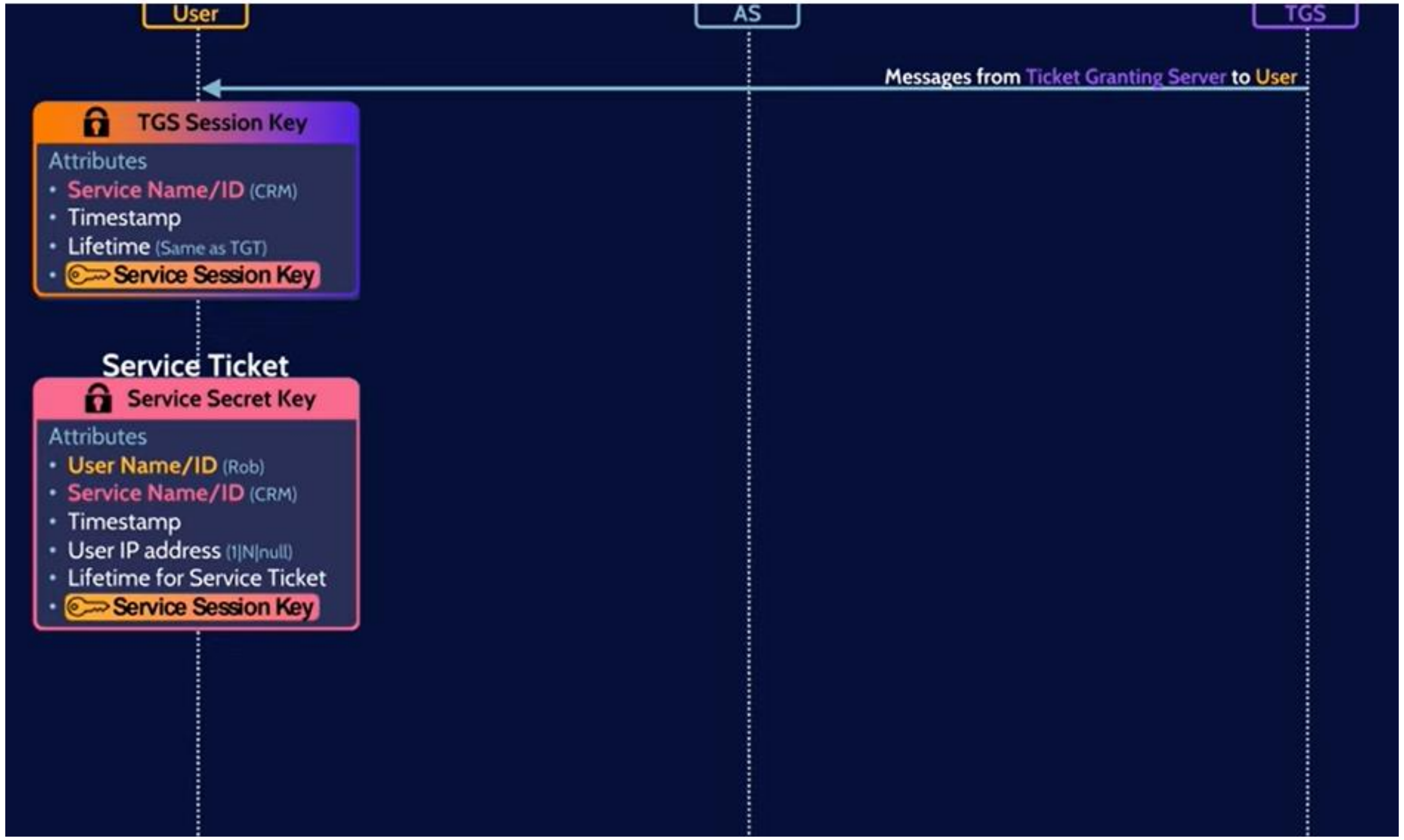
| Name/ID | Client Secret Key |
|---|---|
| Hilda | z&w\H98_RQky |
| Rob | Bq4s%K"?j7gR |
| Stella | [p=3Z]X4]<eT |
| Tess | j8q]7>9NQ6U_ |

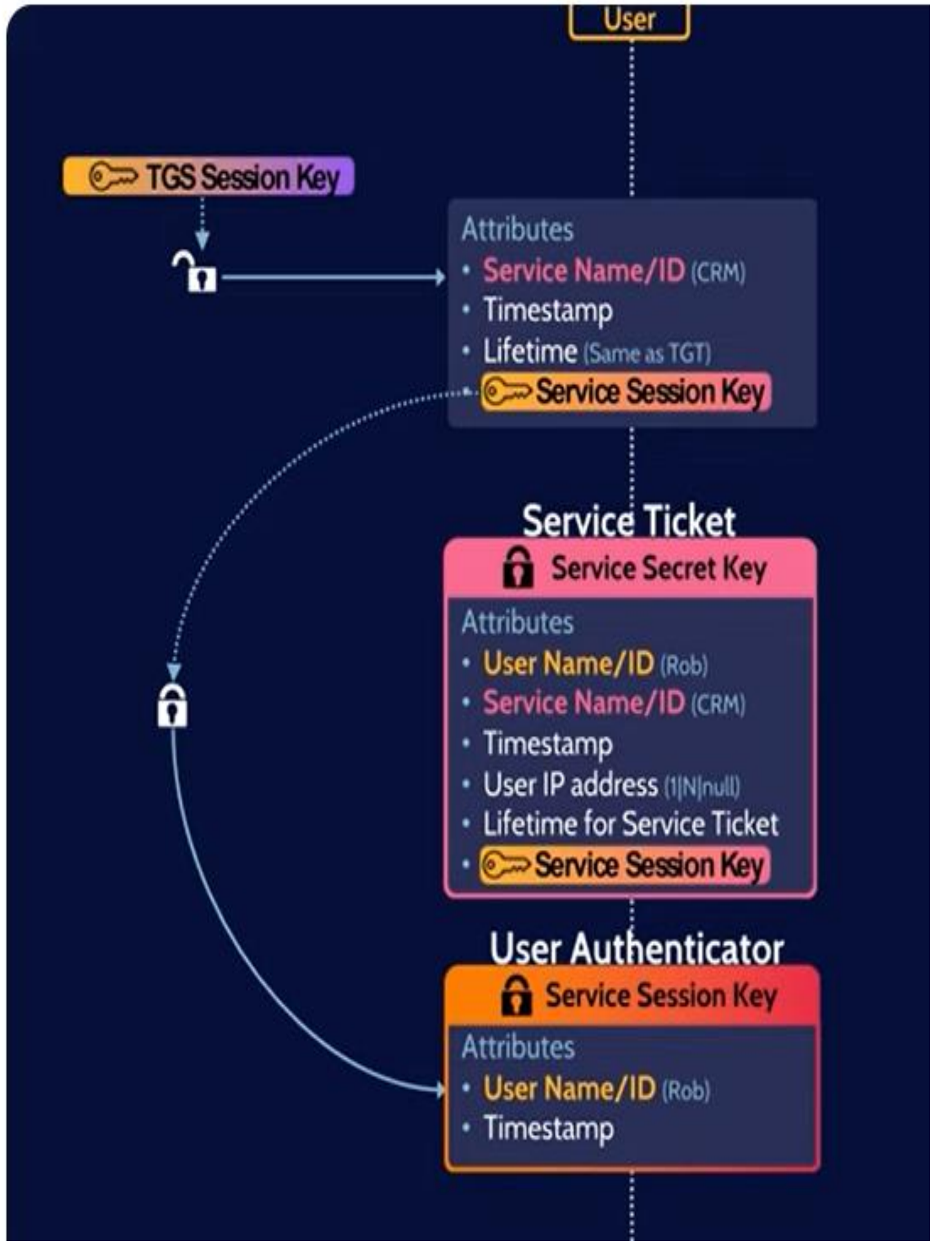
| Service ID | Service Secret Key |
|---|---|
| CRM | $`Sy9n8x~{\, |
| Finance | c#4FMzG!"Ve- |
| Payroll | '7"$T:]Paq@V |
| Travel | 5Qk&,svW$4@E |

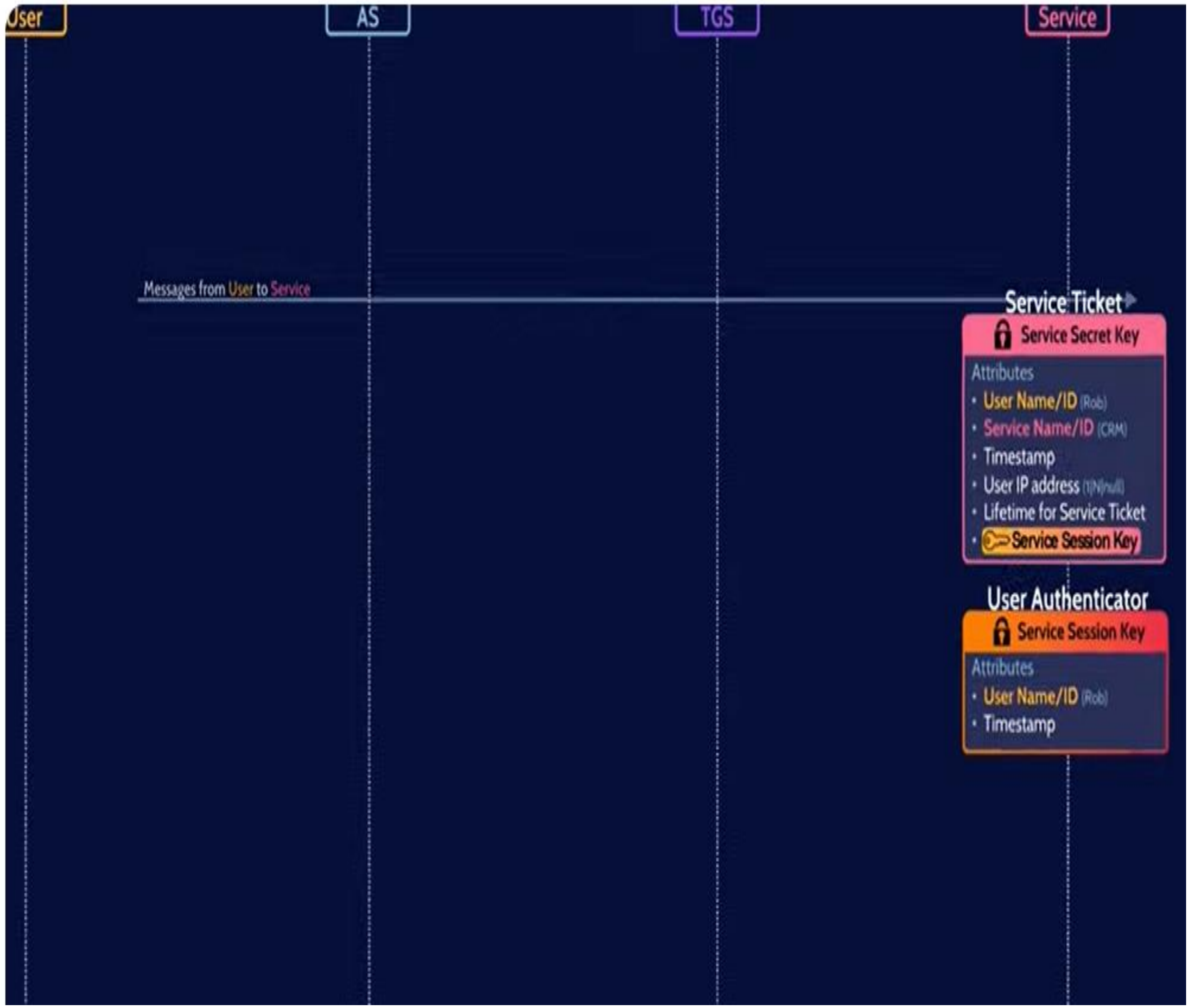

Ticket Granting Ticket
TGS Secret Key
Attributes
• User Name/ID (Rob)
• TGS Name/ID
• Timestamp
• User IP address (1|N|null)
• Lifetime for TGT
• TGS Session Key
Attributes
• Service Name/ID (CRM)
• Requested lifetime for ticket
User Authenticator
TGS Session Key
Attributes
• User Name/ID (Rob)
• Timestamp

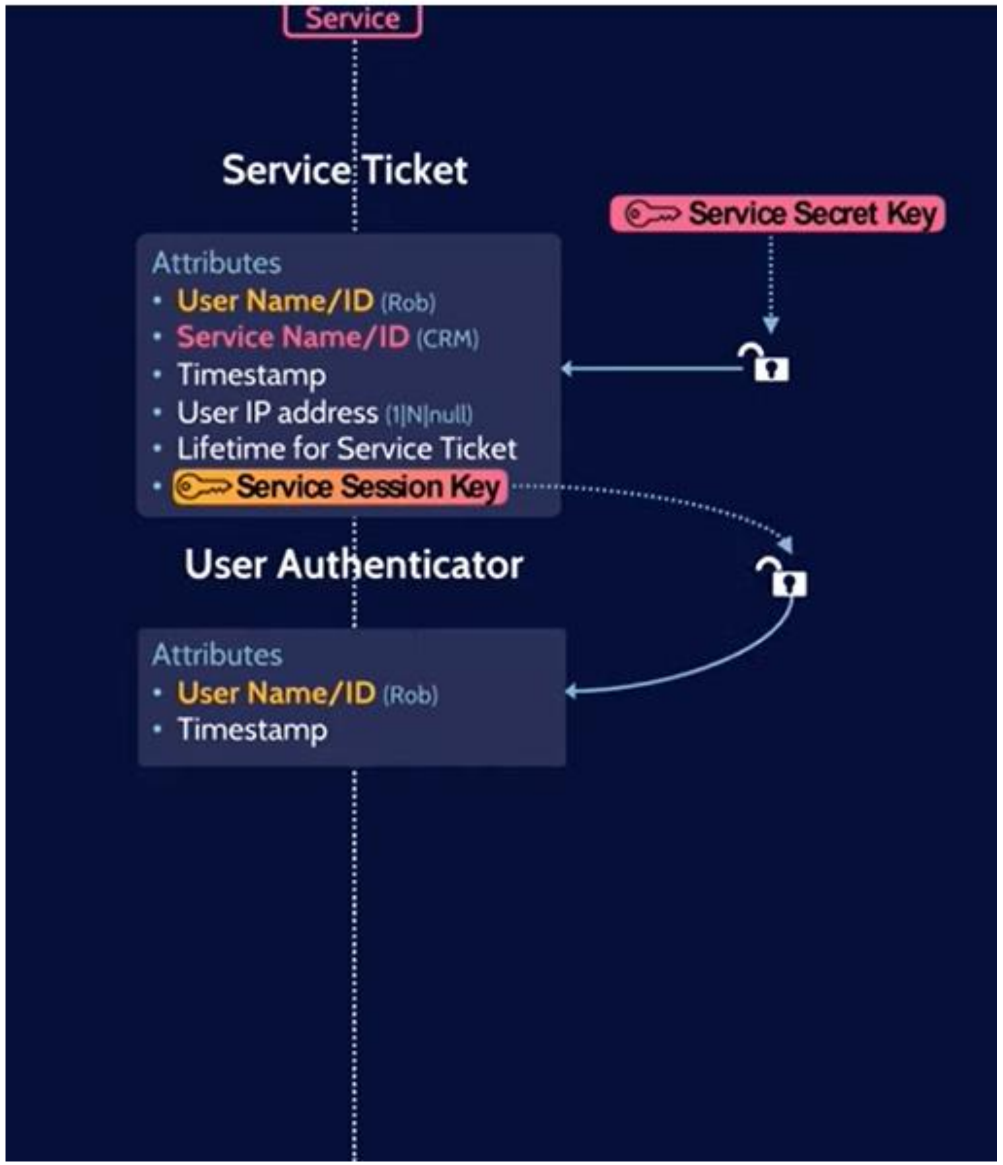

Ticket Granting Ticket
TGS Secret Key
Attributes
User Name/ID (Rob)
• TGS Name/ID
Timestamp
• User IP address (1|N|null)
• Lifetime for TGT
• TGS Session Key
Attributes
Service Name/ID (CRM)
• Requested lifetime for ticket
User Authenticator
Attributes
User Name/ID (Rob)
Timestamp

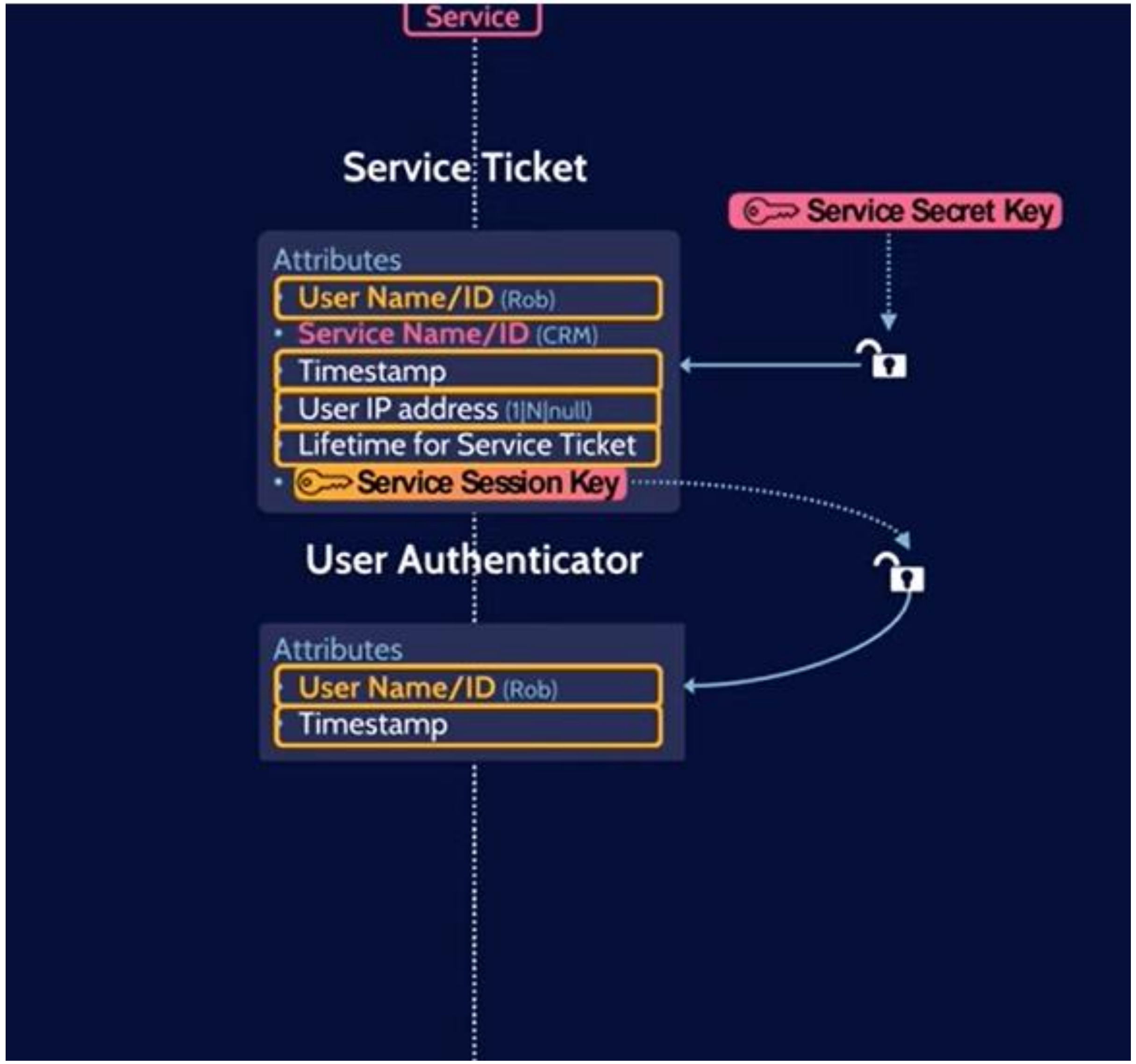

TGS
User Authenticator
Attributes
User Name/ID (Rob)
Timestamp
TGS Session Key
Attributes
Service Name/ID (CRM)
Timestamp
Lifetime (Same as TGT)
Service Session Key
Service Ticket
Service Secret Key
Attributes
User Name/ID (Rob)
Service Name/ID (CRM)
Timestamp
User IP address (1|N|null)
Lifetime for Service Ticket
Service Session Key

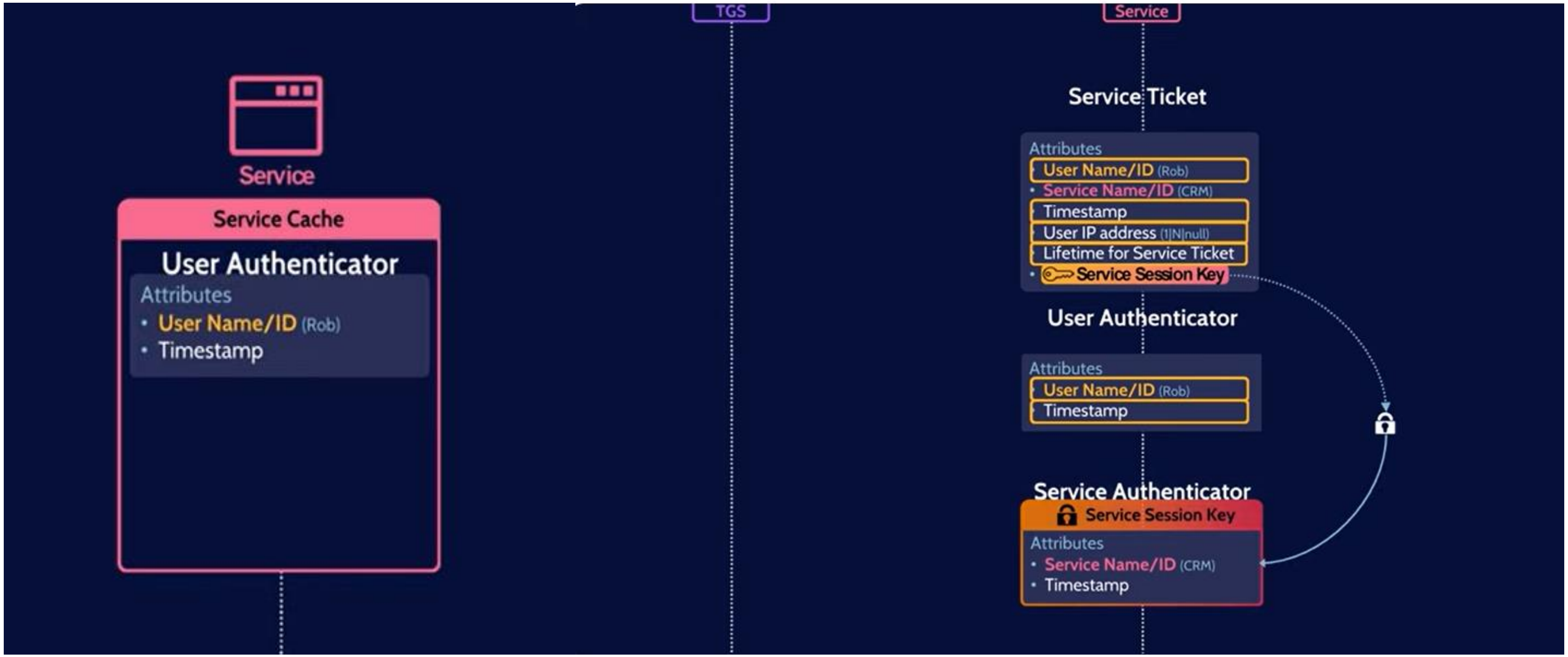

User
AS
TGS
Messages from Ticket Granting Server to User
TGS Session Key
Attributes
Service Name/ID (CRM)
Timestamp
Lifetime (Same as TGT)
Service Session Key
Service Ticket
Service Secret Key
Attributes
User Name/ID (Rob)
Service Name/ID (CRM)
Timestamp
User IP address (1|N|null)
Lifetime for Service Ticket
Service Session Key

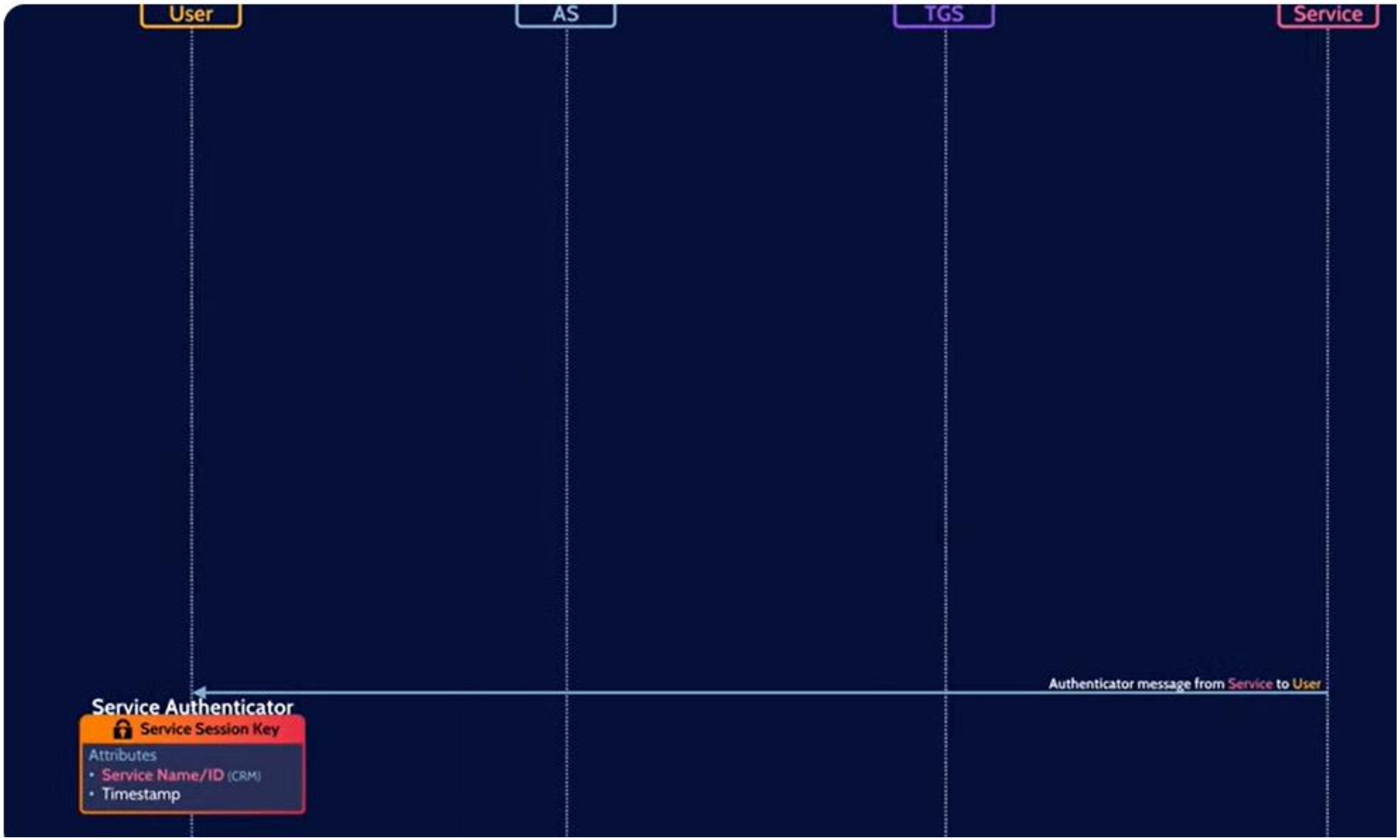

User
TGS Session Key
Attributes
Service Name/ID (CRM)
Timestamp
Lifetime (Same as TGT)
Service Session Key
Service Ticket
Service Secret Key
Attributes
User Name/ID (Rob)
Service Name/ID (CRM)
Timestamp
User IP address (1|N|null)
Lifetime for Service Ticket
Service Session Key
User Authenticator
Service Session Key
Attributes
User Name/ID (Rob)
Timestamp

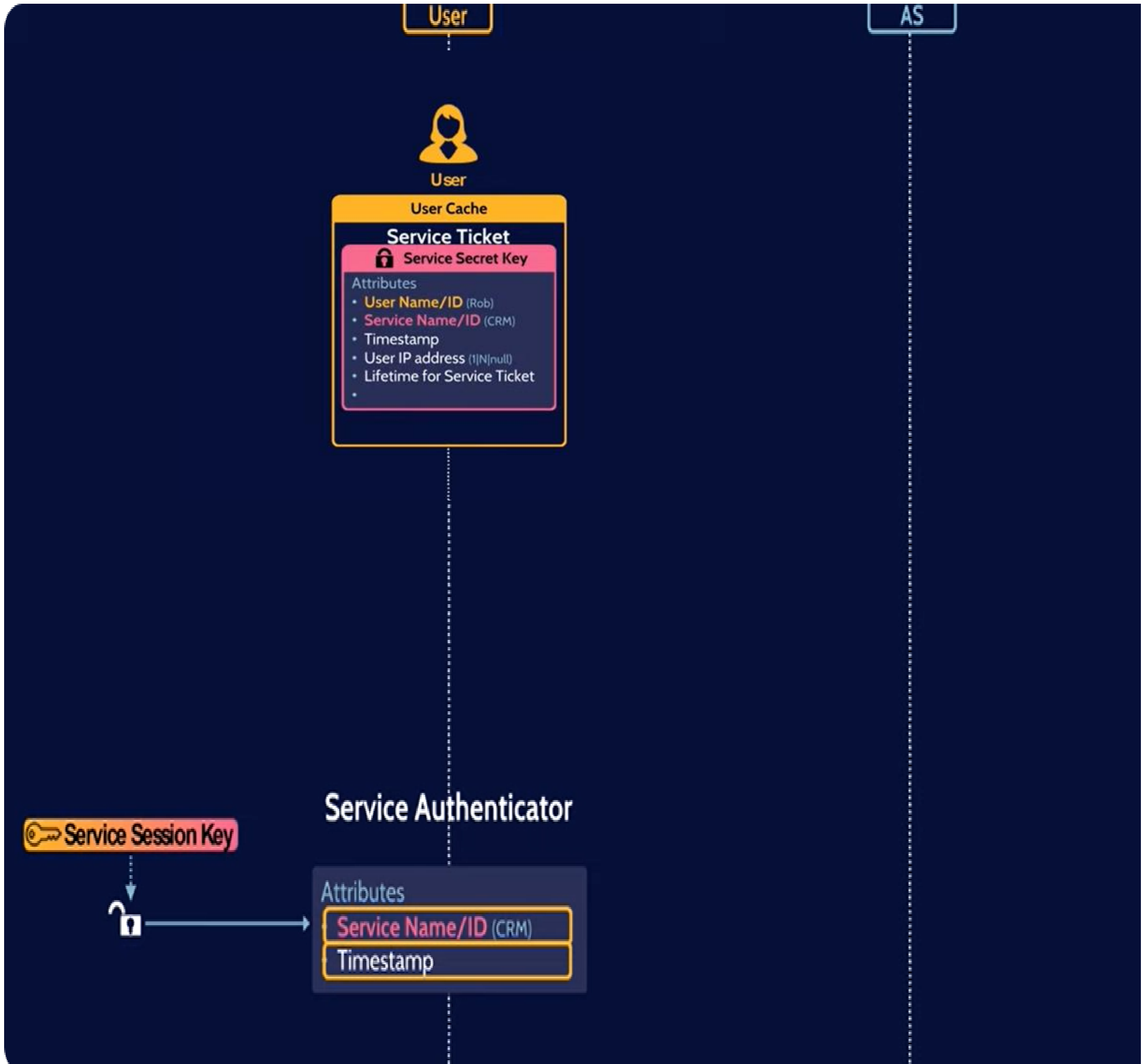

User
AS
TGS
Service
Messages from User to Service
Service Ticket
Service Secret Key
Attributes
User Name/ID (Rob)
Service Name/ID (CRM)
Timestamp
User IP address (1|N|null)
Lifetime for Service Ticket
Service Session Key
User Authenticator
Service Session Key
Attributes
User Name/ID (Rob)
Timestamp

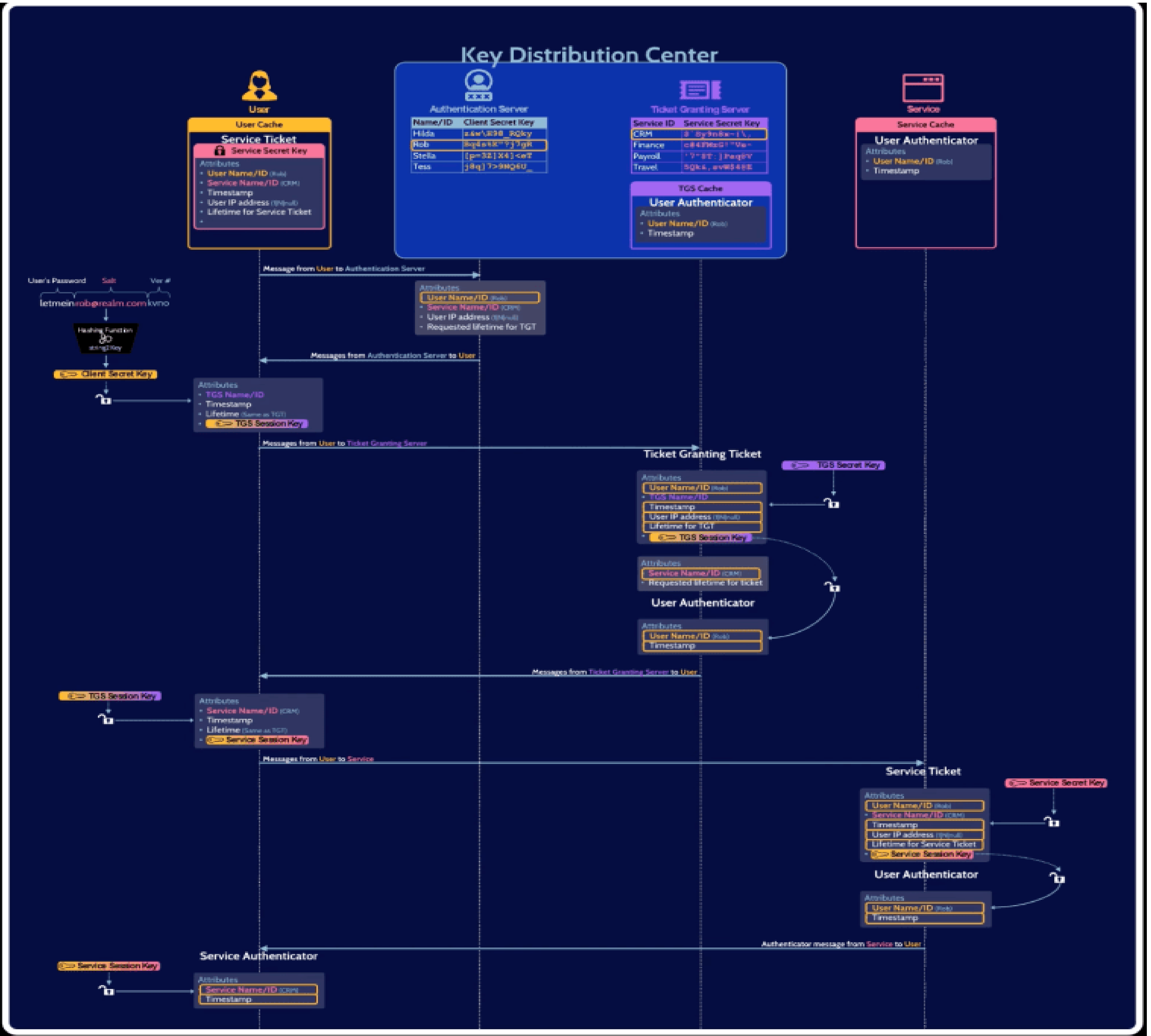

Service
Service Ticket
Service Secret Key
Attributes
User Name/ID (Rob)
Service Name/ID (CRM)
Timestamp
User IP address (1|N|null)
Lifetime for Service Ticket
Service Session Key
User Authenticator
Attributes
User Name/ID (Rob)
Timestamp


Service
Service Ticket
Service Secret Key
Attributes
User Name/ID (Rob)
Service Name/ID (CRM)
Timestamp
User IP address (1|N|null)
Lifetime for Service Ticket
Service Session Key
User Authenticator
Attributes
User Name/ID (Rob)
Timestamp


TGS
Service
Service
Service Cache
User Authenticator
Attributes
User Name/ID (Rob)
Timestamp
Service Ticket
Attributes
User Name/ID (Rob)
Service Name/ID (CRM)
Timestamp
User IP address (1|N|null)
Lifetime for Service Ticket
Service Session Key
User Authenticator
Attributes
User Name/ID (Rob)
Timestamp
Service Authenticator
Service Session Key
Attributes
Service Name/ID (CRM)
Timestamp


User
AS
TGS
Service
Authenticator message from Service to User
Service Authenticator
Service Session Key
Attributes
• Service Name/ID (CRM)
• Timestamp


User
AS
User
User Cache
Service Ticket
Service Secret Key
Attributes
User Name/ID (Rob)
Service Name/ID (CRM)
Timestamp
User IP address (1|N|null)
Lifetime for Service Ticket
Service Authenticator
Service Session Key
Attributes
Service Name/ID (CRM)
Timestamp



| Name/ID | Client Secret Key |
|---|---|
| Hilda | z&w\X98_RQky |
| Rob | 8q4s%X^?j7gR |
| Stella | [p=3Z]X4]<eT |
| Tess | j8q]?>9HQ6U_ |

| Service ID | Service Secret Key |
|---|---|
| CRM | #`8y9n8e-|\, |
| Finance | c#4PMxG!"Ve- |
| Payroll | '?"8T:]PaqBV |
| Travel | 5Qk&,av@$4$E |

## Access Control

| Action | Description | Scenario example | Computer process |
|---|---|---|---|
| Identification | Review of credentials | Delivery person shows employee badge | User enters username |
| Authentication | Validate credentials as genuine | Mia reads badge to determine it is real | User provides password |
| Authorization | Permission granted for admittance | Mia opens door to allow delivery person in | User authorized to log in |
| Access | Right given to access specific resources | Delivery person can only retrieve box by door | User allowed to access only specific data |

## Functionality vs Security

**Authentication**
The process of confirming that a user is who they claim to be. For example, a user or computer might prove their identity to a server or client by using a username and password.
**Authorization (Access Control)**
The process of determining what resources a user can access and granting access based on that level. For example, a user might have access to a restaurant, but there are specific things they are not allowed to do.

DOCTOR FUN
WE WANT TO IMPLANT THIS RFID TAG IN YOU.
THAT VIOLATES MY RIGHTS!
WE WANT TO IMPLANT THIS RFID TAG IN YOU AND IT'S ALSO A CELLPHONE, DIGITAL CAMERA, AND MP3 PLAYER.
COOL!
WRONG
RIGHT

# Authentication and Access control

**Authentication**: Verifying identity using credentials like passwords, biometrics, or tokens.

- **Authentication Factors**:
    - *Knowledge-Based*: Something the user knows (e.g., passwords, PINs).
    - *Possession-Based*: Something the user has (e.g., smart cards, tokens).
    - *Inherence-Based*: Something the user is (e.g., fingerprints, retina scans).
- **Multifactor Authentication (MFA)**: Combines two or more factors to enhance security (e.g., password + smartphone app).
- **Single Sign-On (SSO)**: Enables one login for multiple systems, reducing password fatigue but increasing risk if compromised.
- **Authentication Protocols**: Examples include Kerberos, OAuth, and OpenID Connect, securing authentication during data exchange.

## Common Authentication Methods

- **Passwords:**
  - The most widely used method but vulnerable to brute force, phishing, and dictionary attacks.
  - Best practices include using strong, unique passwords and storing them securely (e.g., hashing with salt).
- **Biometrics:**
  - Includes fingerprints, voice recognition, and facial scans.
  - Offers high security but raises privacy concerns and potential risks of spoofing.
- **Two-Factor Authentication (2FA):**
  - Adds an extra layer of security by requiring a secondary verification step, such as a one-time password (OTP) sent to a device.
- **Public Key Infrastructure (PKI):**
  - Uses cryptographic techniques to authenticate entities, ensuring secure communication.

## Emerging Trends in Authentication Methods

- **Passwordless Authentication:**
  - Relies on technologies like biometrics and security keys, reducing dependency on traditional passwords.
- **Behavioral Biometrics:**
  - Analyzes patterns like typing speed and mouse movements for continuous authentication.
- **Decentralized Authentication:**
  - Blockchain-based systems that eliminate central authorities, reducing risks of breaches.

## Challenges in Authentication

- **Credential Theft:**
- Stolen credentials through phishing or social engineering pose significant risks.
- **Usability vs. Security:**
- Balancing user convenience with robust security measures is challenging.
- **Scalability:**
- Adapting authentication systems for large-scale environments with diverse users and devices.

## Best Practices for Authentication

- Employ MFA (Multi-factor authentication) wherever possible.
- Regularly audit and update authentication policies.
- Educate users on recognizing and avoiding phishing attempts.
- Use adaptive authentication systems that assess the context and risk level (e.g., unusual locations or devices).

Authentication is not a one-size-fits-all solution but a continuously evolving component of information security, adapting to new threats and technologies.

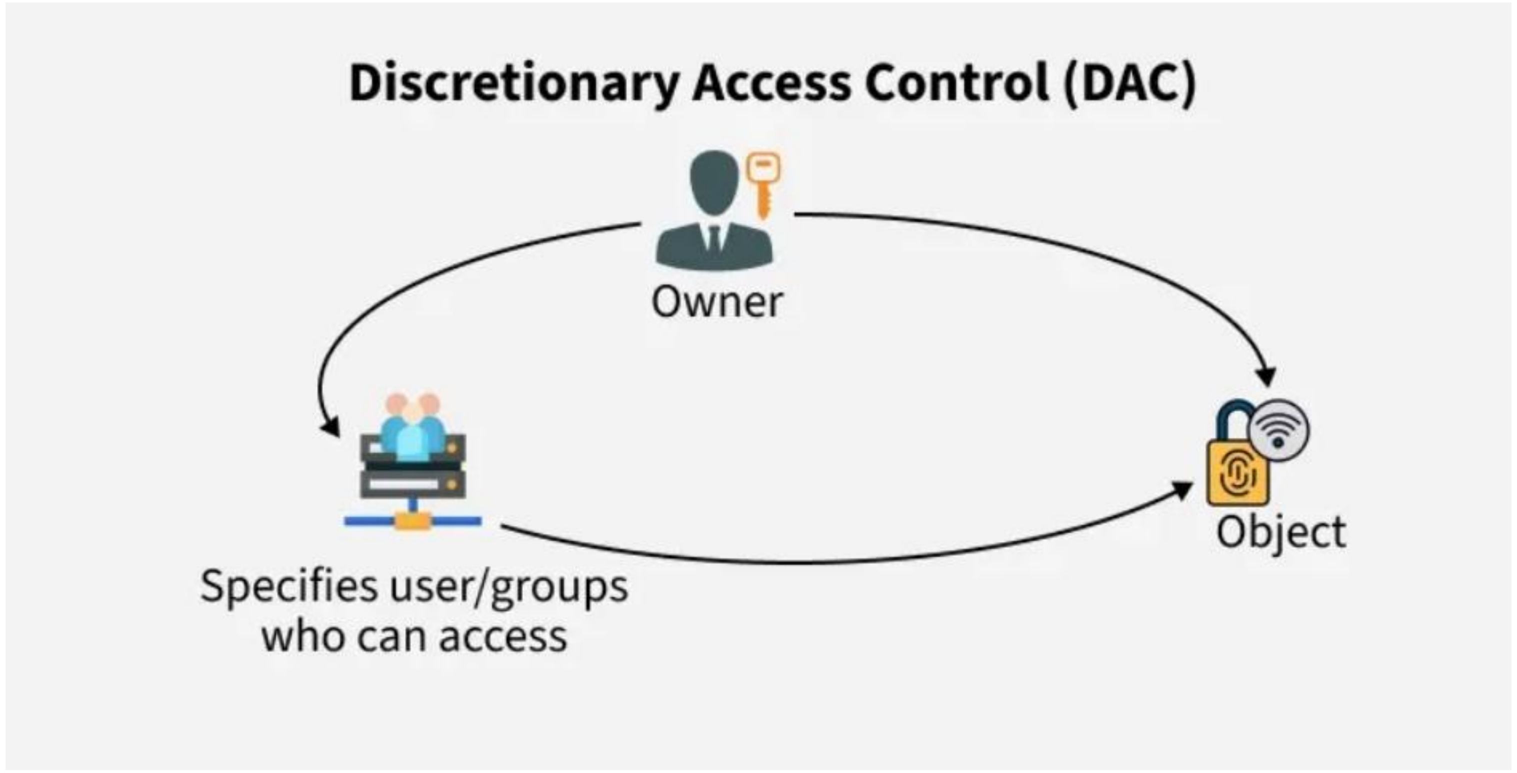
## Access Control Models

- 1. Discretionary Access Control (DAC)
- 2. Mandatory Access Control (MAC)
- 3. Role-Based Access Control (RBAC)
- 4. Attribute-Based Access Control (ABAC)

## Discretionary Access Control(DAC)

- Discretionary Access Control empowers the owners of resources to decide who can access them. The control over permissions lies entirely with the resource owner.
- **Core Characteristics**
    - **Ownership-Centric:** The individual who owns the resource holds complete authority over its access rights.
    - **Highly Customizable:** Permissions can be tailored for specific users or groups, allowing detailed control.
    - **Access Control Lists (ACLs):** Typically managed through lists that define which users or groups can perform actions like reading, writing, or executing the resource.
- **Benefits**
    - Straightforward and easy for users to manage.
    - Provides fine-grained control over who can do what.
- **Drawbacks**
    - Susceptible to risks such as privilege escalation and threats from insiders.
    - The absence of centralized oversight complicates the enforcement of organization-wide security policies.
- **Illustration** For example, when a user creates a file on their device, they can specify exactly which other users are allowed to read, modify, or run that file.

## DAC Diagram:


Discretionary Access Control (DAC)
Owner
Specifies user/groups who can access
Object

# Mandatory Access Control (MAC)

Mandatory Access Control is a security model where access rules are strictly enforced by a central authority, and users have no ability to alter these rules.

•**Core Characteristics**

•Security Labels and Clearance Levels: Both users and resources receive specific security classifications (such as "Top Secret" or "Confidential"), and access is permitted only when these labels align.

•Non-Discretionary Access: Users cannot modify permissions themselves.

•Centralized Authority: Only system administrators have the power to set or change access policies.

•**Benefits**

•Delivers a high level of security.

•Perfectly suited for settings demanding rigorous confidentiality, like military or government operations.

•**Drawbacks**

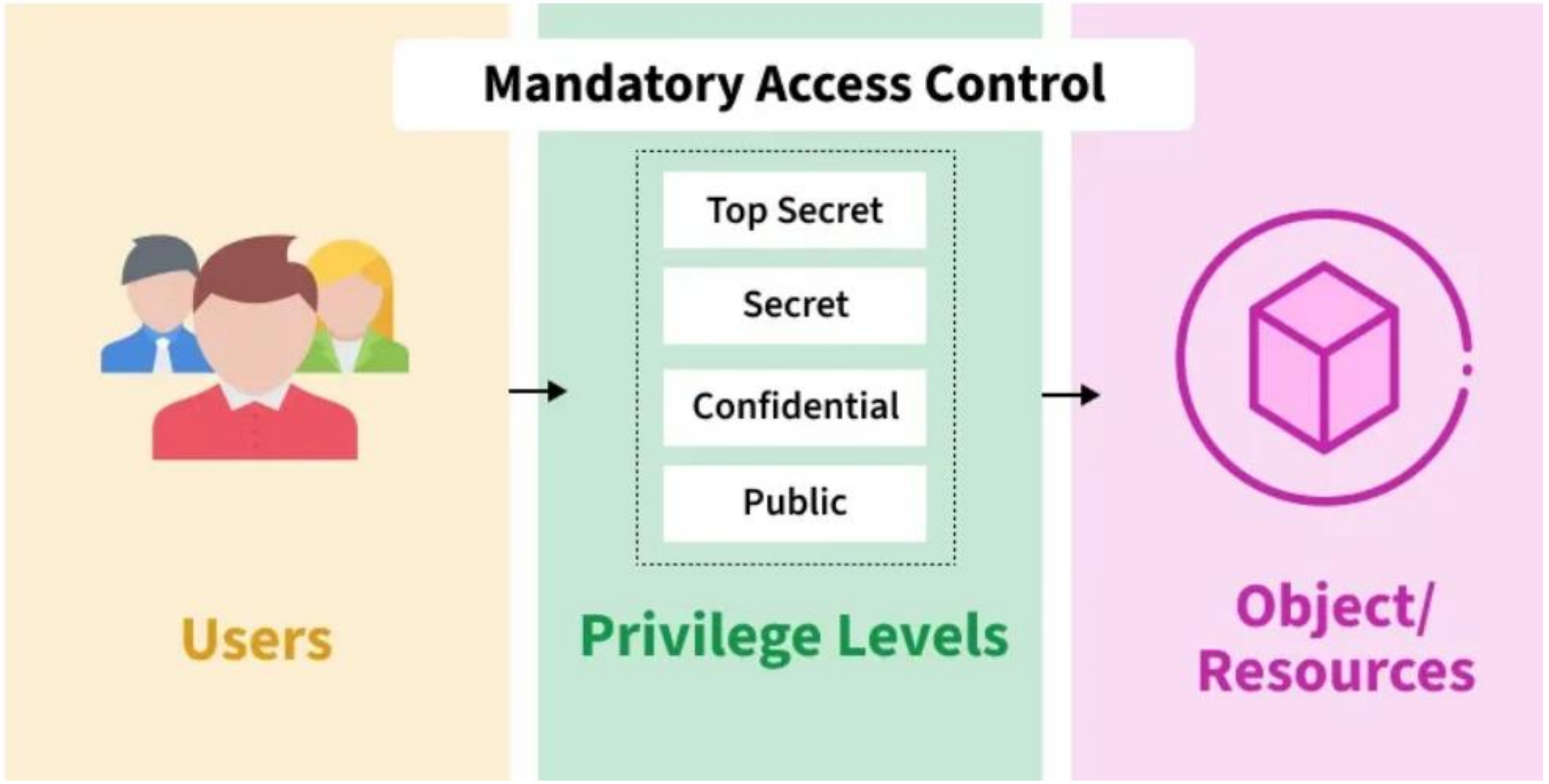
•Can be rigid and less intuitive for users.

•Challenging to deploy and maintain in environments that require frequent changes.

•**Illustration**

•For example, a document marked "Top Secret" is accessible exclusively to individuals holding "Top Secret" clearance.

## MAC Diagram:

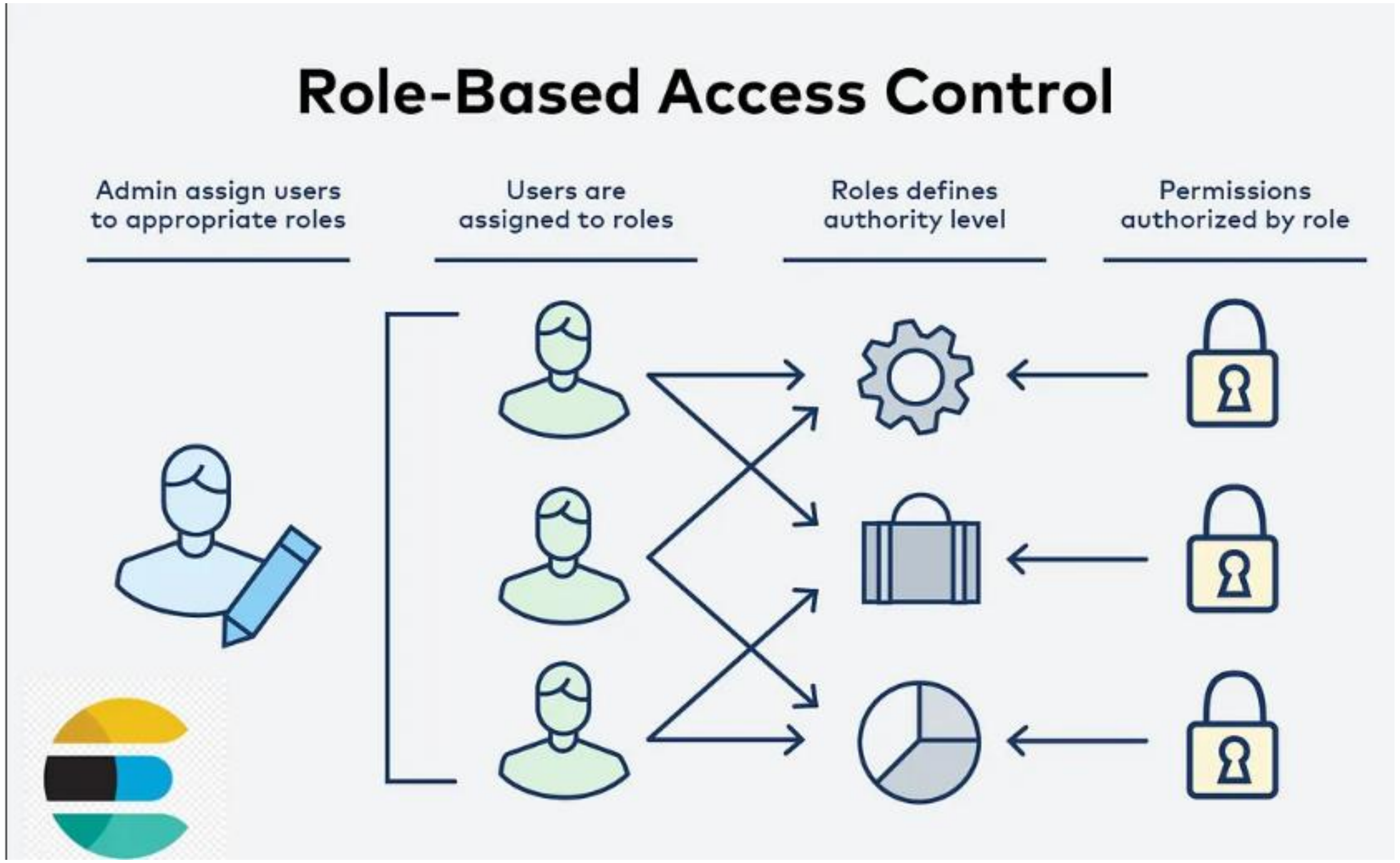

Mandatory Access Control
Top Secret
Secret
Confidential
Public
Users
Privilege Levels
Object/
Resources

# Role-Based Access Control (RBAC)

RBAC manages user permissions by assigning them to specific roles instead of individual users, making access control more efficient.

- **Core Characteristics**
  - **Role-Focused:** Users receive roles such as Admin, Editor, or Viewer, each with a set of predefined access rights.
  - **Highly Scalable:** Ideal for organizations with complex hierarchies and large user bases.
  - **Duty Segregation:** Roles can be structured to avoid conflicts of interest, for example, ensuring one person cannot both approve and audit the same transaction.
- **Benefits**
  - Simplifies administration in large enterprises.
  - Enhances the ease of auditing and meeting compliance requirements.
- **Limitations**
  - Setting up roles initially can be intricate and time-consuming.
  - Lacks flexibility to adapt to contextual or real-time conditions.
- **Practical Example**
  - Within a company, an "HR Manager" role might be granted access to employee information, whereas a "Finance Manager" role would have permissions related to payroll data.

## RBAC Diagram:

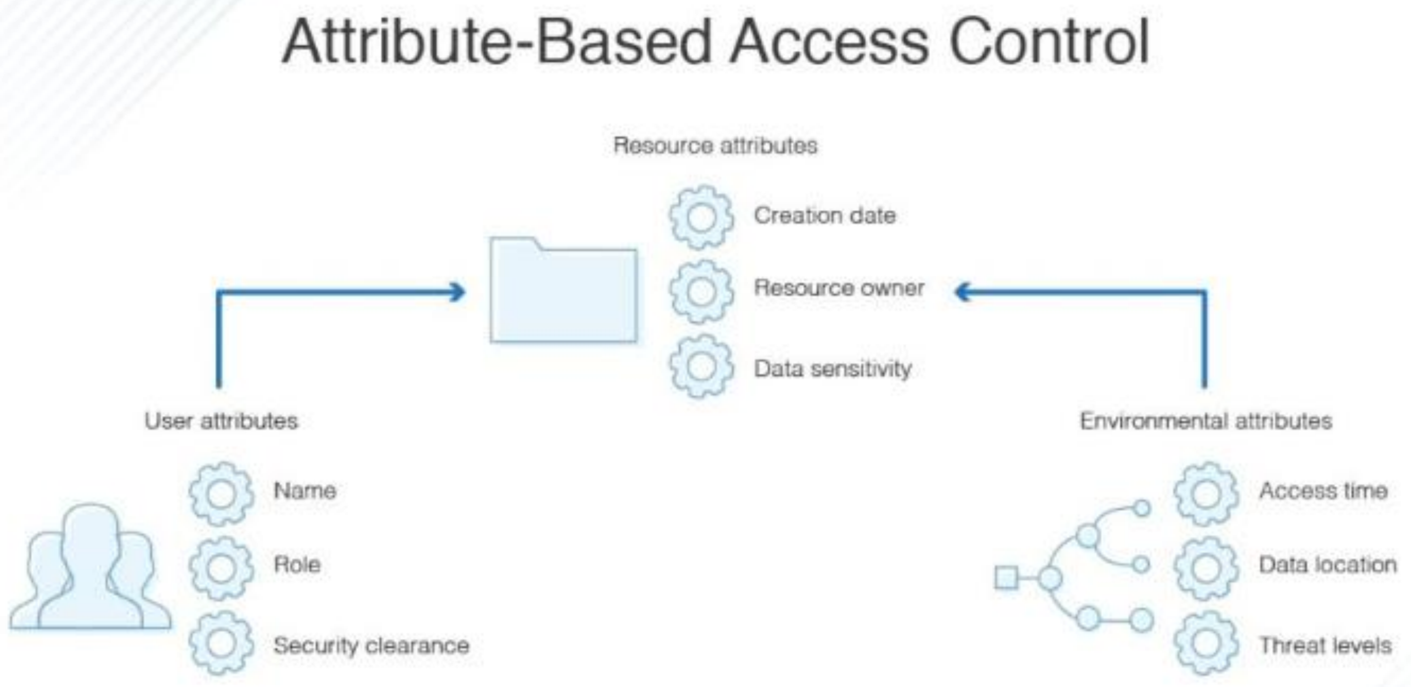

Role-Based Access Control
Admin assign users to appropriate roles
Users are assigned to roles
Roles defines authority level
Permissions authorized by role

# Attribute-Based Access Control (ABAC)

ABAC determines access rights by evaluating various attributes related to the user, the resource, and the surrounding environment, enabling real-time, adaptive access management.

- **Core Characteristics**
    - **Policy-Driven:** Access permissions are governed by rules that assess specific attributes.
    - **Context-Sensitive and Dynamic:** Factors such as the user's role, current time, location, and device type influence access decisions.
    - **Granular Access:** Supports detailed and conditional policies for precise control.
- **Benefits**
    - Offers exceptional flexibility, making it suitable for environments that change frequently.
    - Capable of handling intricate policies that involve multiple attribute combinations.
- **Challenges**
    - Can impose significant processing demands.
    - Crafting and maintaining policies can be complex and resource-intensive.
- **Illustrative Example**
    - Access might be permitted only if the individual is an employee (attribute: role), accessing during business hours (attribute: time), and using a company-approved device (attribute: device type).

## ABAC Diagram:



| Name | Restrictions | Description |
|---|---|---|
| Mandatory Access Control (MAC) | End user cannot set controls | Most restrictive model |
| Discretionary Access Control (DAC) | Subject has total control over objects | Least restrictive model |
| Role Based Access Control (RBAC) | Assigns permissions to particular roles in the organization and then users are assigned to roles | Considered a more “real-world” approach |
| Rule Based Access Control (RBAC) | Dynamically assigns roles to subjects based on a set of rules defined by a custodian | Used for managing user access to one or more systems |

## Comparison

These models cater to different use cases and security requirements, and many organizations employ a hybrid approach to leverage their .strengths

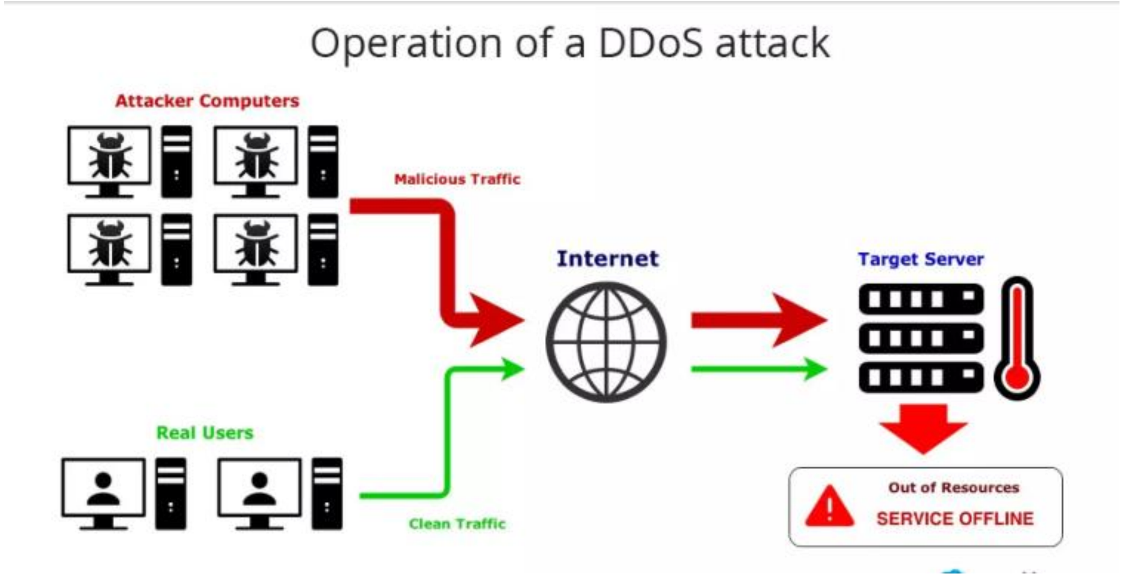
| Feature | DAC | MAC | RBAC | ABAC |
|---|---|---|---|---|
| Control Authority | Resource Owner | Centralized Administrator | Role-Based | Attribute-Based |
| Flexibility | High | Low | Moderate | Very High |
| Scalability | Low | Low | High | High |
| Security | Moderate | Very High | High | Very High |
| Example Usage | Personal Files | Government Systems | Corporate IT Systems | Cloud Services |

## DDoS Attack

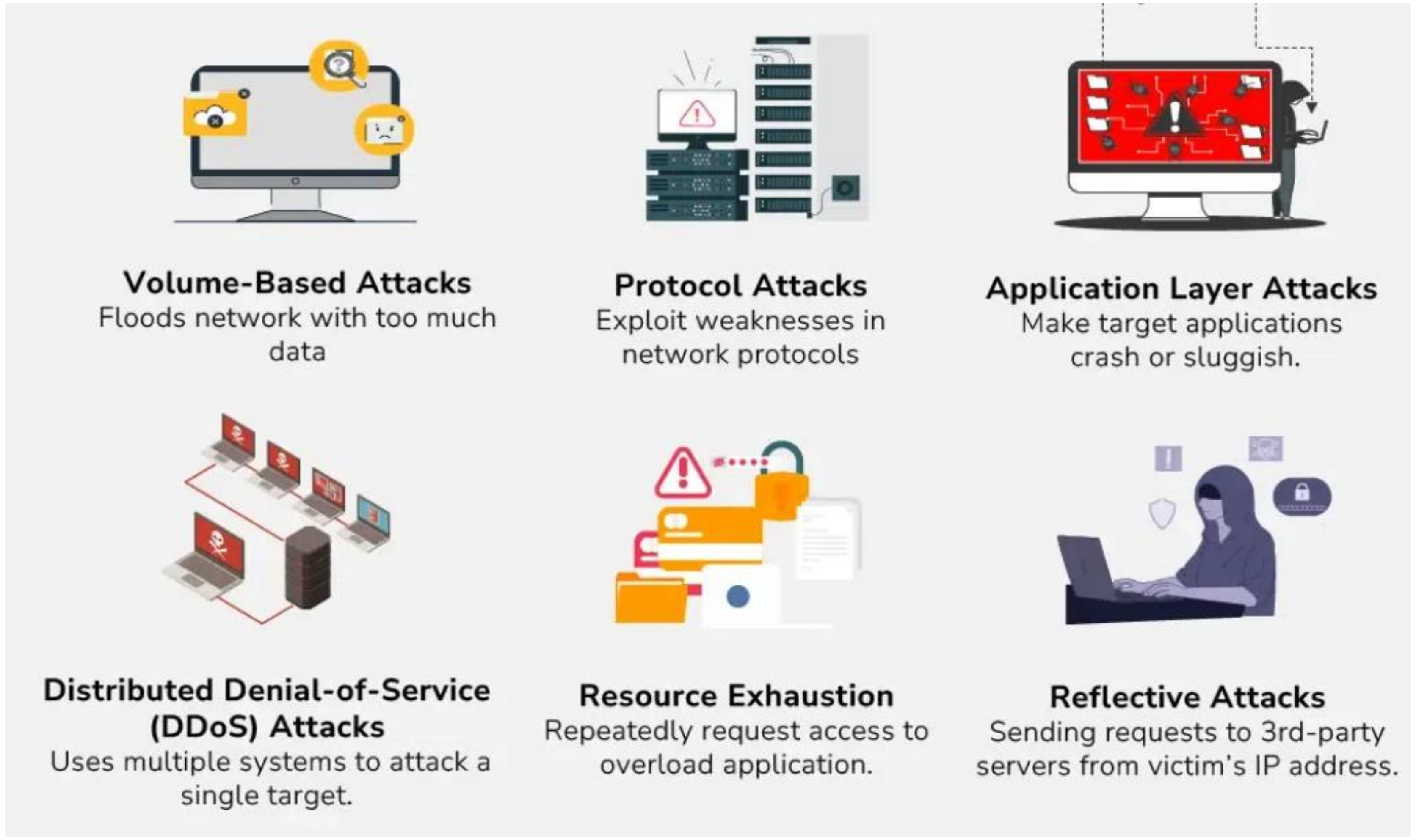
## Intro:

Denial of service (DOS) is a network security attack, in which, the hacker makes the system or data unavailable to someone who needs it. Hacker tries to make a network, system, or machine unavailable by flooding it with fake requests or traffic. This prevents real users from accessing it, causing anything from slowdowns to complete shutdowns.

DDoS attacks use multiple systems, often compromised computers (botnets), to attack a single target. Examples are amplification attacks and botnet-based attacks. In an amplification attack, attackers use services like DNS to send a small query that generates a large response, flooding the victim with data. Botnets coordinate many infected computers to send attack traffic from multiple sources, making it hard to defend against..



# Types of DoS Attack



**Volume-Based Attacks**
Floods network with too much data

**Protocol Attacks**
Exploit weaknesses in network protocols

**Application Layer Attacks**
Make target applications crash or sluggish.

**Distributed Denial-of-Service (DDoS) Attacks**
Uses multiple systems to attack a single target.

**Resource Exhaustion**
Repeatedly request access to overload application.

**Reflective Attacks**
Sending requests to 3rd-party servers from victim's IP address.

## Wireless security

# Why Wireless Security is Critical

Key Risk Factors:

1. Channel – Broadcast nature → prone to eavesdropping, jamming
2. Mobility → Portability increases risk of loss/theft/unauthorized access
3. Resources – Limited memory/processing in mobile devices → vulnerable to malware.
4. Accessibility – Devices left unattended → physical attacks

**Wireless Network Threats**

**Accidental Association**
Example: Employee's laptop connects to a neighboring company's Wi-Fi unintentionally.

**Malicious Association**
Example: Attacker sets up a fake "Free Airport Wi-Fi" AP to capture login credentials.

**Ad Hoc Networks**
Example: Two laptops connected directly without AP → no central control.

## Wireless Network Components & Attack Points

- **Wireless Client** – Phone, laptop, IoT device
- **Access Point** – Wi-Fi hotspot, cell tower
- **Transmission Medium** – Radio waves (vulnerable to interception)
- **Example:**
  A Wi-Fi-enabled laptop in a café connects to a rogue AP instead of the legitimate one.

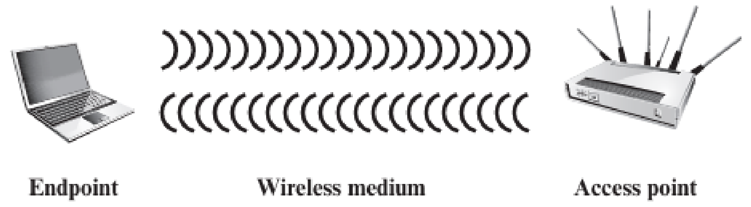


Figure 18.1 Wireless Networking Components

## Securing Wireless Transmissions

- **Goal:** Prevent eavesdropping, tampering, disruption
- **Signal Hiding Techniques**
  - Disable SSID broadcasting
  - Use cryptic SSID names
  - Reduce signal strength
  - Use directional antennas
- **Encryption**
  - Use WPA3, AES-CCMP
    *Example:* Encrypt all Wi-Fi traffic so intercepted data is unreadable.

## Securing Wireless Access Points

- **Threat:** Unauthorized network access
  **Solution:** IEEE 802.1X Port-Based Access Control
- Requires authentication before granting network access
- Prevents rogue APs
  *Example:* Employee must enter username/password/certificate to connect to corporate Wi-Fi.
  - 
    *Example:* Encrypt all Wi-Fi traffic so intercepted data is unreadable.

## Securing Wireless Networks – Best Practices

- **Use encryption** (WPA3 for router-to-router)
- **Install antivirus/firewall** on all endpoints
- **Turn off identifier broadcasting**
- **Change default router SSID**
- **Change default admin password**
- **Use MAC filtering** (with caution – can be spoofed)
- *Example:* Home Wi-Fi → Change SSID from “Linksys” to “HomeNet_secure”, disable SSID broadcast, enable WPA3.

# Firework

- A firewall is a network security system that:
  - Creates a controlled barrier between trusted internal networks and untrusted external networks (e.g., the Internet)
  - Inspects and controls traffic based on security rules
  - Provides a single point of monitoring and enforcement
- “The function of a strong position is to make the forces holding it practically unassailable.” – Carl Von Clausewitz

Example:

- A company uses a firewall to allow employees to browse the web while blocking hackers from entering the network.

## Firewall characteristics:

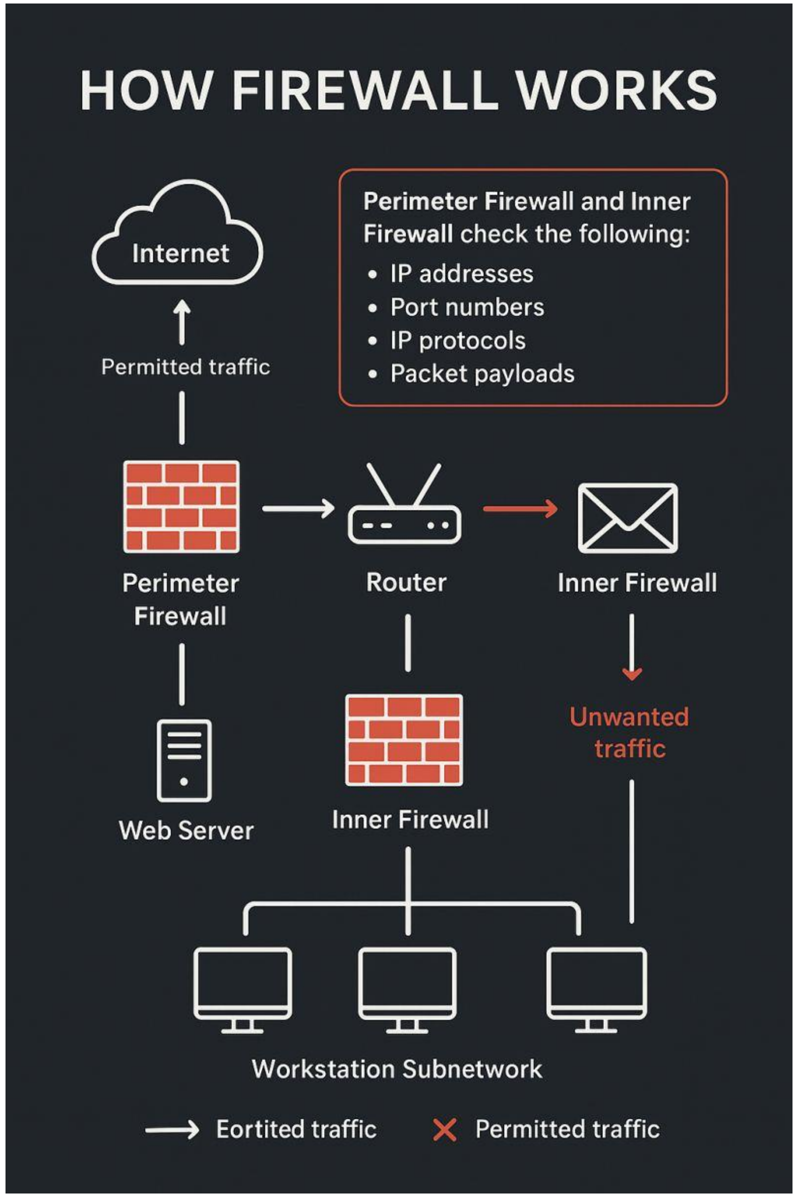
Firewalls are designed to:

- **Force all traffic through them** – no bypass allowed
- **Permit only authorized traffic** – defined by security policy
- **Be immune to attacks** – run on hardened, secure systems

**Control Techniques:**

- **Service Control** – What services (HTTP, SMTP) are allowed
- **Direction Control** – Which way traffic can flow
- **User Control** – Who can access (via authentication)
- **Behavior Control** – How services are used (e.g., block spam)

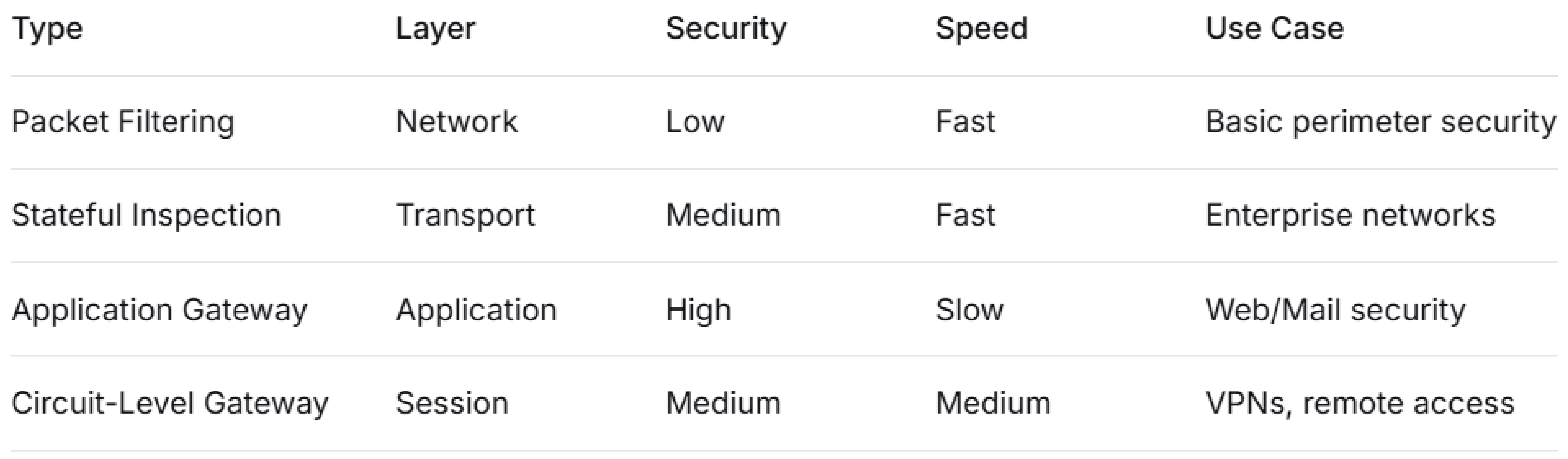
**Example:**
Only HR can access the payroll server, and only during work hours.



## Types of firewall

- **Packet Filtering Firewall**
  - Filters based on IP, port, protocol
  - Fast but minimal inspection
  - *Example:* Block all traffic from a specific IP
- **Stateful Inspection Firewall**
  - Tracks active connections in a state table
  - Only allows return traffic for established sessions
  - *Example:* Allow inbound response only if outbound request was made first
- **Application-Level Gateway (Proxy)**
  - Acts as an intermediary for app traffic (e.g., HTTP, FTP)
  - Inspects full content, enforces detailed policies
  - *Example:* Web proxy that blocks malicious sites
- **Circuit-Level Gateway**
  - Relays TCP connections without deep inspection
  - Often used with SOCKS protocol
  - *Example:* Secure relay for remote user VPN access

## Summary:



| Type | Layer | Security | Speed | Use Case |
|---|---|---|---|---|
| Packet Filtering | Network | Low | Fast | Basic perimeter security |
| Stateful Inspection | Transport | Medium | Fast | Enterprise networks |
| Application Gateway | Application | High | Slow | Web/Mail security |
| Circuit-Level Gateway | Session | Medium | Medium | VPNs, remote access |

# References:

- Stallings, W. (2017). *Cryptography and Network Security: Principles and Practice* (7th ed.). Pearson. Chapter: User Authentication.
- Stallings, W. (2017). *Cryptography and Network Security: Principles and Practice* (7th ed.). Pearson. Chapter: Access Control.
- Stallings, W. (2017). *Cryptography and Network Security: Principles and Practice* (7th ed.). Pearson. Chapter: Wireless security.
- Stallings, W. (2017). *Cryptography and Network Security: Principles and Practice* (7th ed.). Pearson. Chapter: Firewalls.
- https://sheerazali.com/how-to-kerberos-its-components-and-function/
- https://www.youtube.com/watch?v=5N242XcKAsM
- https://www.slideshare.net/slideshow/009-authentication-and-access-control-pptx/273786341#4
- https://www.apono.io/blog/rbac-vs-abac-choosing-the-right-access-control-model-for-your-organization/
- https://levelup.gitconnected.com/implementing-role-based-access-control-rbac-in-elasticsearch-for-security-65a44042aaf5?gi=0fa7d24977ab
- https://www.geeksforgeeks.org/computer-networks/difference-between-dac-and-mac/