From AAA to …

ISEP Jean-Philippe Lelièvre

jpl@hearandknow.eu

2023/12

- Basic concepts and definitions
- Strong authentication
- AAA protocols

Basic concepts and definitions

Basic security concepts

- Confidentiality
- Integrity
- Availability
- Non repudiation
- Audit
- Identification/authentication
- Trusted Third Party

Basic concepts : confidentiality

Confidentiality is keeping secret informations with access only to the authorised entities.

- example : medical file, personal files
- Threats
  - COMINT
  - Hacking (interception and illicit reuse of information)

Basic concepts: integrity

Integrity guarantees that information is modified only by one volontarist and authorised action.

- example : bank account
- Threat : logical sabotage

Basic concepts: availability

Availability is system capability to fulfill one function within predefined conditions (time schedule, or performance).

- Example : air traffic management
- Threats
  - Accidents
    - floodings,fire
    - pannes
  - Sabotage/tampering
    - physical
    - logical

Basic concepts : audit

- Procedure allows to record and to fill a story book from the access and operations done on the work position.
- The audit permits to detect anomalities (errors, tampering...) or to keep track from the user activity to analyse it for statistics and decision help.

Basic concepts: non repudiation

- **Non repudiation** means to ensure that a transferred message has been sent and received by the parties claiming to have sent and received the message.
- Non repudiation is a way to guarantee that :
  - the sender of a message cannot later deny having sent the message
  - that the recipient cannot deny having received the message.

Basic concepts: Trusted Third Party

- **Third Party** means (not Alice, not Bob)
- **Trusted** entity which facilitates interactions between two parties who both trust the third party (cf. Wikipedia)

Basic concepts : identification and authentication

- **Identify** one user, one computer = ask its identity.
- **Authenticate** one user, one computer = ask to proof this identity.

Identity proof

- What the entity :
  - Knows : secret
    - Secret (Password, PIN code…)
  - Owns : token
    - Identity documents (Passport, ID card, driving licence…)
    - Social security card
  - Is : biometry
    - DNA
    - Fingerprint
    - Face recognition
    - Voice recognition
    - Eyes recognition
    - …
  - Ideally, verification is a function of the three methods V= F(Token, Secret, Biometry)

Basic concepts: AAA

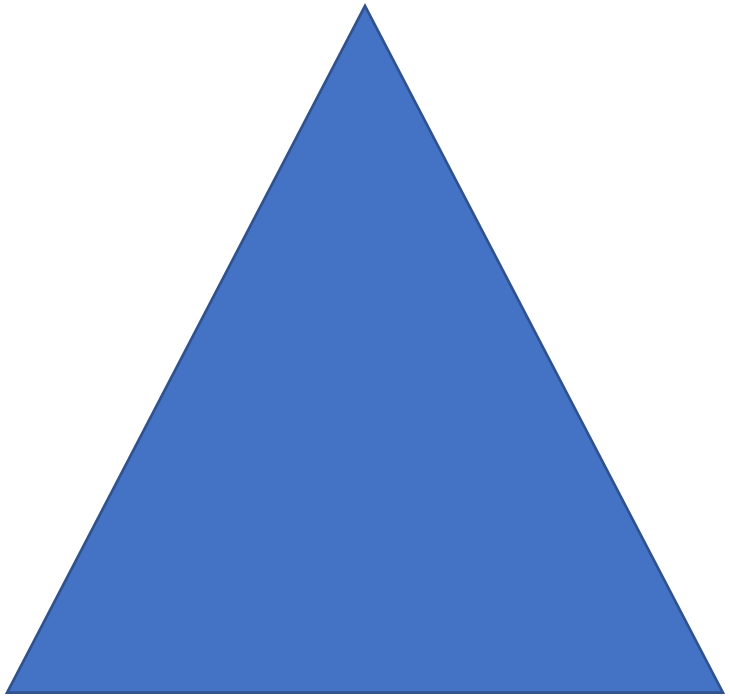
- **Authentication : Identication + proof**
- **Authorization :** specifying access rights/privileges to resources. Define an access policy
- **Accounting :** cf. audit

- Examples : Radius, TACACS, Diameter

AAA Framework

Who is the user
Examples : Login, Password

Authentication



Grants privileges to the user specifying access rights to resources.

- Defines :
  - an access policy
  - Tasks that can be performed
  - How long they can be performed

Authorization

Keeps tracks/logs

- What did the user use?
- Resources, how much, how long

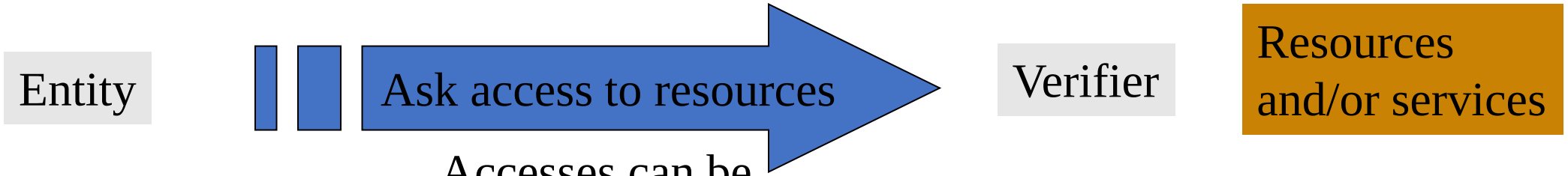
Accounting

Access control

Entity may be:
- User
- Group of users
- Computer
- Network address

...

Resources can be :
- Files
- Network objects
- Computers
- Tasks
- Data bases
- ...



Accesses can be
- R/W
- Execute
- Delete
- Control
- Update
- ...

Access control : step by step

1) Identification

2) Authentication

Authenticating one entity = asks for the proof
authentication = identification + verification

Entity

3) Autorisation

Verifier

Resource and/or services

Validation or denial of the resource access by the verifier

4) Action

5) Accounting

Control the actions from subjects onto objects

- Speaking of security means defining :
  - subjects : … identified…authentic
  - objects : …data …software … hardware
  - actions : …access …Read… Write…Delete
- From the « controlled universe », and the property that this control will provide:
  - authenticity ... confidentiality ... integrity … availability
  - That will be obtained by the security policy and the implementation of the appropriate security functions.

Strong authentication

Strong as opposed to weak

- Examples
- Pin codes
- Passwords (weaknesses (dictionary attacks…)
- Recovery questions
- …

Strong authentication

- Goal : remove static authentication with password (replayable and vulnerable to interception) and replace it by dynamic password
- Authentication solutions :
  - Synchronous ex : SecurID (time-synched)
  - Challenge/response ex : SKey (Lamport)
  - Certificates (cf PKI)
  - One Time Password
  - …

Authentication with time-synched clock

The code sent by Alice has a short time validity to avoid replay attacks

Alice token is a special timer
(original reference, not based on
hours, minutes, seconds)

The server (Vera)
knows « Alice's time »

Alice

Vera

1) Alice identifies herself

2) Vera asks Alice's « time»

3) Alice types her PIN code
on her token and sends the « time »
displayed

4) Vera verifies that the time received
corresponds to Alice's.

5) Vera grants the access rights

Authentication with S/Key token (Lamport's protocol RFC 1760)

F is a one way function
$F(X_0)=X_1$,
$F^2(X_0)=F(X_1)=X_2$ …
Alice
knows the function F() and X_0

The server (Vera)
knows the function F() and one iteration
$F^k(X_0)=X_k$ but notX_0

Alice

During the next session

Vera

2)Vera sends k
4) Vera verifies that F(sent)=Xk and if OK, stores X_{k-1}
5) Vera grants the access rights
7) Vera sends k-2
9) Vera verifies that F(sent)=X_k-$_1$ and if yes, stores X_{k-2}

1) Alice identifies herself

3) Alice calculates and sends X_{k-1}

6) Alice identifies herself

8) Alice calculates and sends Xk-$_2$

Basics of symmetrical encryption

- When Alice encrypts one message with its private key, only the recipient(s) holding this key (Bob) is able to decrypt.
- It implies that beforehand Alice and Bob own this key as common secret

Drawbacks of symmetrical encryption

- Heavy key management imply that entities need to know each other before hand in order to have exchanged keys
- Not adapted to
  - exchanges on open networks (e-commerce, B to C)
  - Document signature

Challenge/response authentication with a symmetrical algorithm

F is a symmetrical cryptographic function

Alice
knows the function F() and
one encryption key Kc

The server (Vera)
knows the function F() and the encryption Key
Kc= Kd decryption Key

Alice

Vera

And so on

2)Vera sends a random number

1) Alice identifies herself

4) Vera verifies that F(Y, Kd)= random

3) Alice calculates and sends
Y=F(random,Kc)

5) Vera grants the access rights

Basics from asymmetrical encryption

- Algorithm uses two keys, one private key and one public key with the following property : the public key known it is impossible to calculate the private key.
- When Alice encrypts one message with her private key, everybody possessing Alice public key can decrypt it
- When someone encrypt one message with Alice public key , only Alice private key can decrypt it

Challenge/response authentication with an asymmetrical algorithm

F is a cryptographic function

Alice
knows the function F() and
an encryption key Kc (Alice Private key)

The server (Vera)
knows the function F() and can access the decryption key Kd (Alice's public key)

Alice

2)Vera sends a random number

4) Vera recovers Kd (Alice public key)

5) Vera verifies that
F(message sent, Kd)=random

6) Vera grants the access rights

And so on

Vera

1) Alice identifies herself

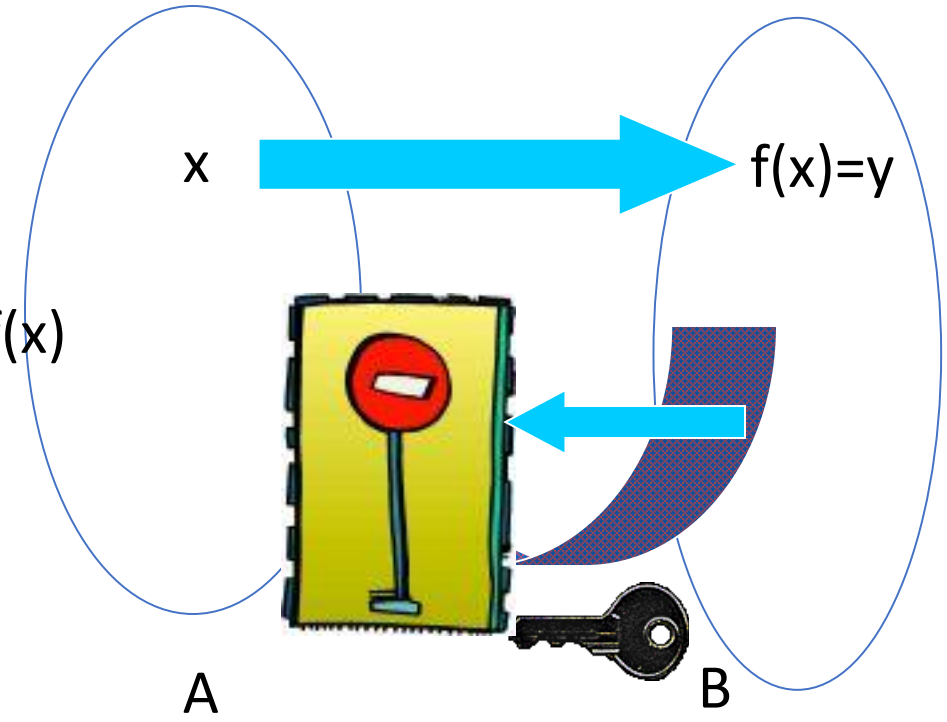
3) Alice calculates and sends F(random, Kc)

One-way functions

function f : from set A to set B

Knowing x included in A,
it is easy to calculate the image y=f(x)

On the contrary, knowing y,
It is difficult to find x



F is said with trap if : knowing y + one information supplementary (trap key) it is possible to determine x

« functions » in everyday's life : mailbox, breaking one egg, mixing colours…

Hash coding

- Use of hashing functions
- Transformation from 1 document whatever its size into one chain of characters (dozen of bytes) representing the original document: the digest.

Characteristics from one hashing function

- One way function (nor predictable nor reversible)

  Different inputs can generate one unique digest

  The same digest is always generated by the same inputs

  One change even a small one in input changes very much the digest

Hash coding

Simple functions similar to hash coding :

- « key », « social security » number
- Checksum
- Modulo

Hash coding examples :

- MD2/MD4/MD5
- SHA

Local authentication with one way function

F is a one way function

Alice **trusts Vera**,
owns a bank card and
knows her password X

The server (Vera)
knows the function F() and is capable of reading Y= F(x) on Alice's card

Alice

Same during each connection

Vera

1) Alice identifies herself

2) Alice sends her password

3) Vera applies F to password

4) Vera compares F(X) to
Y stored in Alice's card

5) Vera grants the access rights

Mutual authentication with challenge/response symmetrical algorithm

Alice knows the cryptographic function F() and
one encryption Kc

The server (Vera) knows the function F() and one encryption key Kc= Kd

Alice

Vera

1) Alice identifies herself

2)Vera sends a random number

3) Alice calculates and sends Y=F(random,Kc)

4) Vera verifies that F(Y, Kd)=random

5) Vera grants the access rights

6) Alice wants to authenticate Vera and sends a random2

7) Vera calculates and sends Y2=F(random2,Kc)

8) Alice verifies that F(Y2, Kd)=random2

Mutual authentication by challenge/response with asymmetrical algorithm

F is a cryptographic function

Alice
knows the function F() and
Alice private key : KprivA

The server (Vera)
knows the function F() and can access
the decryption key KpubA (Alice public key).

Alice

Vera

1) Alice identifies herself

2)Vera sends a random number

3) Alice calculates and sends F(random,KprivA)

4) Vera recovers KpubA

5) Vera verifies that
F(KpubA, received message)=random

6) Vera grants the access rights

7) Alice sends a random number random2

8) Alice recovers Vera's public key KpubV

9) Vera calculates and sends F(random2,KprivV)

10) Alice verifies that
F(KpubV, received message)=random2

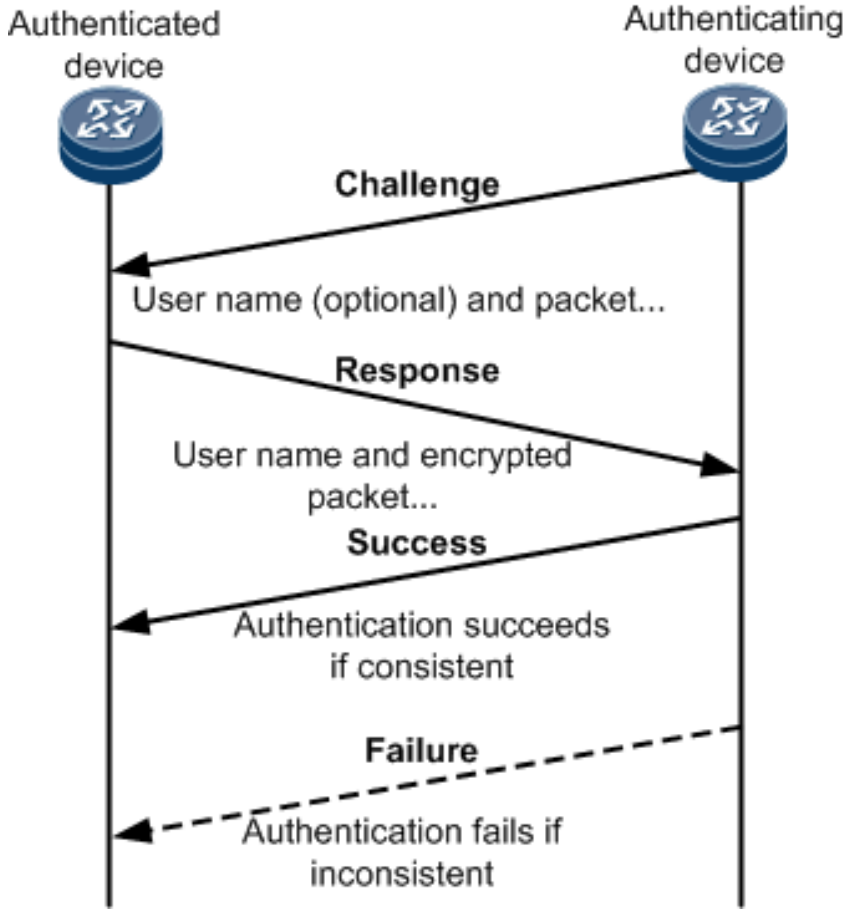
And so on

Password Authentication Protocol (PAP) RFC 1334

- PAP
  - is a password-based authentication protocol used by Point to Point Protocol (PPP) to validate users.
  - authentication is only done at the time of the initial link establishment, and verifies the identity of the client using a two-way handshake:
    1. Client sends username and password. This is sent repeatedly until a response is received from the server.
    2. Server sends authentication-ack (if credentials are OK) or authentication-nak (otherwise)
  - is considered a weak authentication scheme.

Challenge-Handshake Authentication Protocol (CHAP) RFC 1994

- CHAP
  - **authenticates** a user or network host to an authenticating entity.
  - provides protection against **replay attacks** by the peer through the use of an incrementally changing identifier and of a variable challenge-value.
  - requires that both the client and server know the plaintext of the **secret**, although it is **never sent** over the network.



Extensible Authentication Protocol (EAP)

- EAP
  - is an authentication framework frequently used in network and internet connections.
  - is not a wire protocol; instead it only defines the information from the interface and the formats. Each protocol that uses EAP defines a way to encapsulate by the user EAP messages within that protocol's messages.
  - provides some common functions and negotiation of authentication methods called EAP methods.
  - has currently about 40 different methods defined in IETF RFCs : EAP-MD5, EAP-POTP, EAP-GTC, EAP-TLS, EAP-IKEv2, EAP-SIM, EAP-AKA, and EAP-AKA'.
  - [RFC3748]

Protected Extensible Authentication Protocol (PEAP)

- PEAP is a protocol that encapsulates EAP within a potentially encrypted and authenticated Transport Layer Security (TLS) tunnel.

Identity management

- SSO
- Kerberos (protection also against internal attack)
- LDAP
- …

Single Sign-On (SSO)

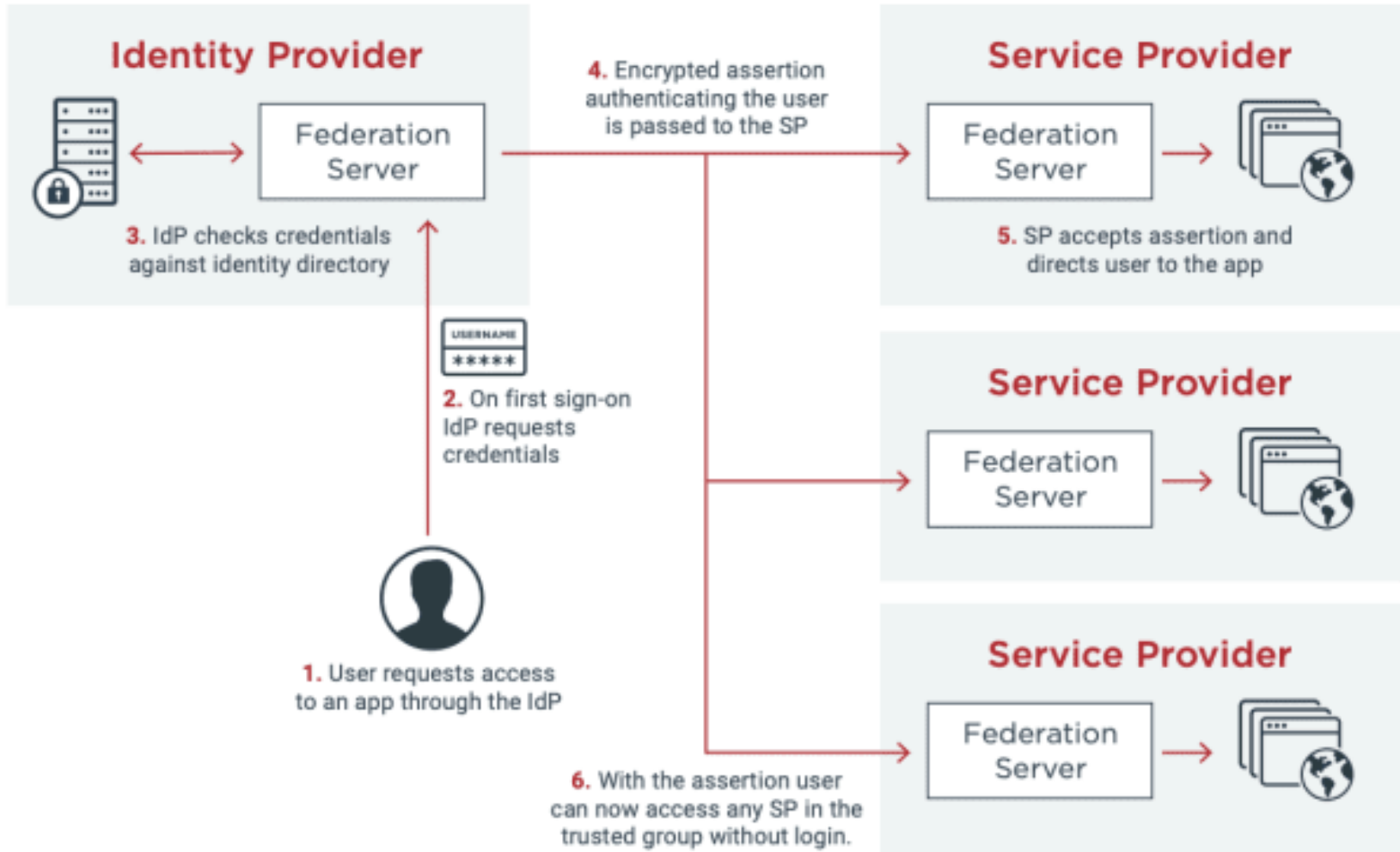
An authentication scheme that allows a user to log in with a single ID and password to any of several related, yet independent, software systems.

SSO shares centralized authentication servers that all other applications and systems use for authentication purposes and combines this with techniques to ensure that users do not have to actively enter their credentials more than once.

Mitigate risk for access to 3rd-party sites ("federated authentication") because user passwords are not stored or managed externally

Examples : Google, Microsoft (pack Office), French administration, Social networks…

IdP-initiated Federated SSO



The six-step sequence illustrates a typical federated SSO use case.

Kerberos

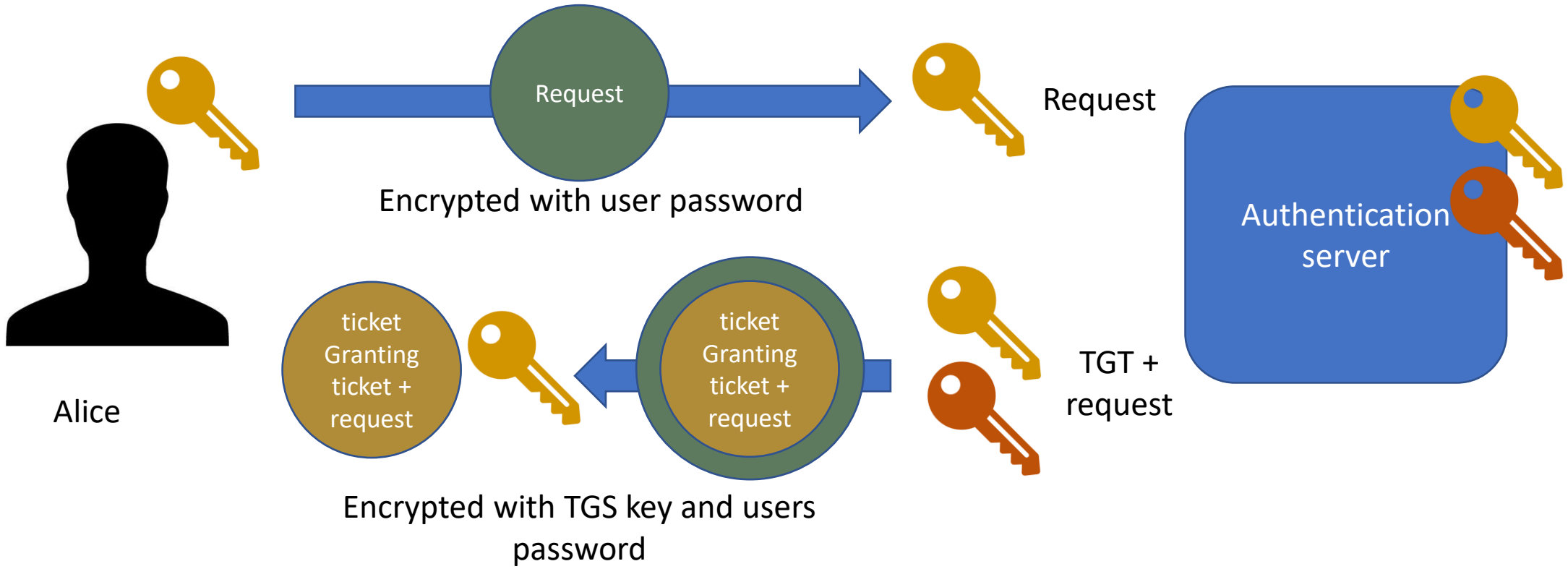
An authentication protocol which works on the basis of tickets between two or more different nodes in an insecure network.

Kerberos builds on symmetric cryptography and requires a trusted third party server known as Kerberos Distribution Center (KDC).

- Authentication Authorization Accounting protocol > identification
- Created by MIT
- Refers to Kerberos : the 3 headed dog guarding hell/inferno,
- Born with Unix OS
- Goal : avoid transmitting password in clear

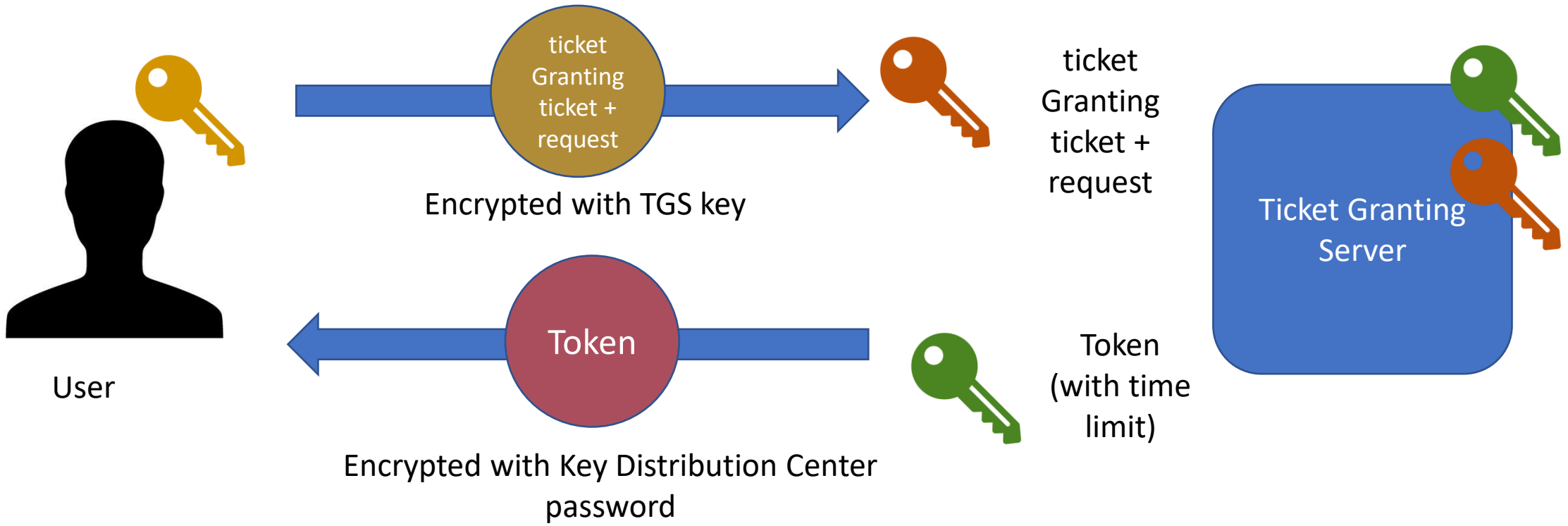
[Kerberos: The network Authentication Protocol (mit.edu)]

Step 1 : From client to authentication server



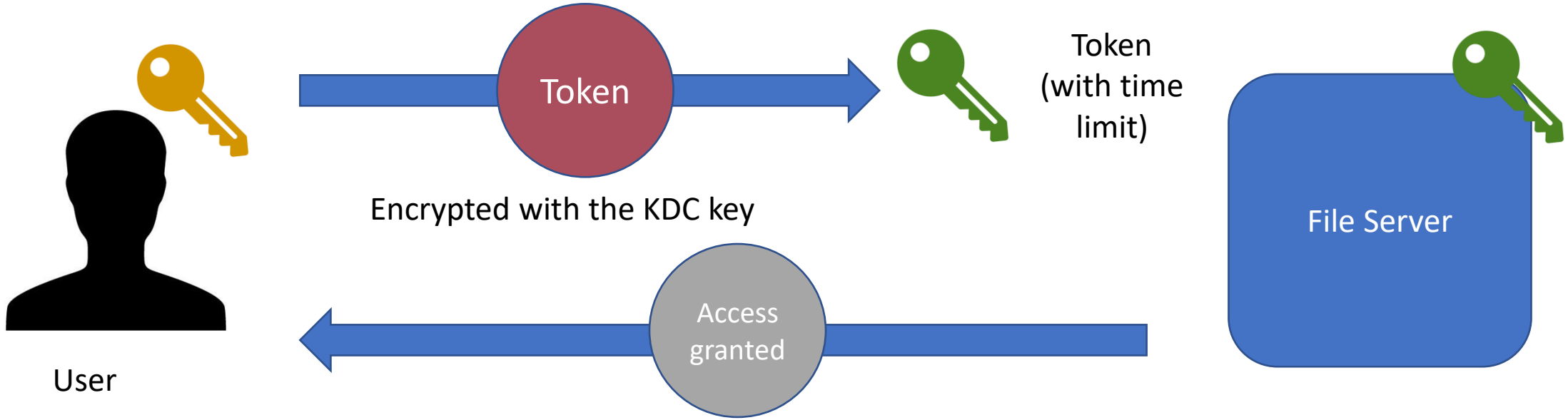
Application Server (AS) : Application Server authenticates user and basis of authentication generates ticket granting ticket (TGT)

Step 2 : From client to ticket granting server

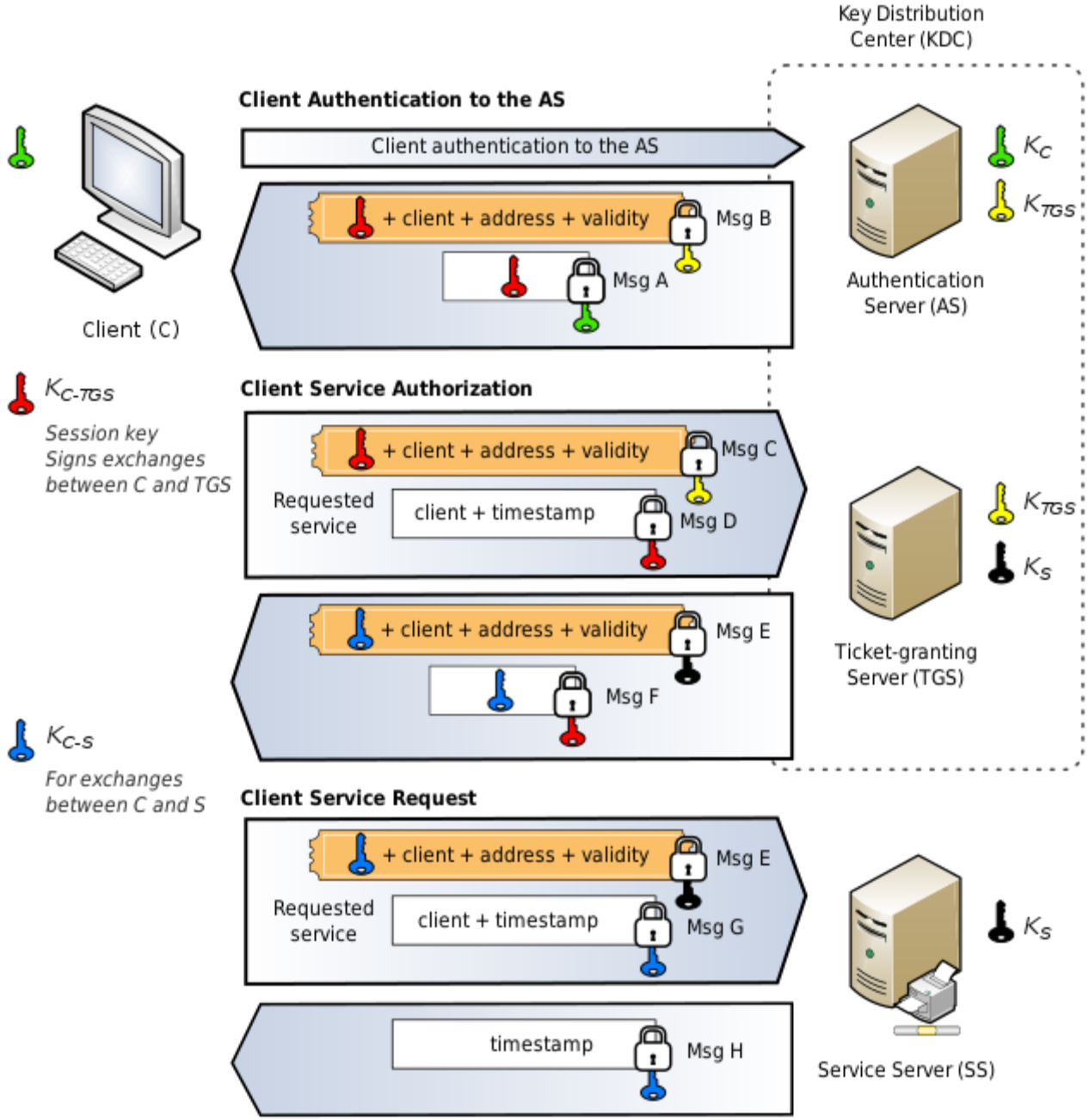


Ticket granting server (TGS) : TGS issues the service ticket for the server.

Step 3 : From client to file server

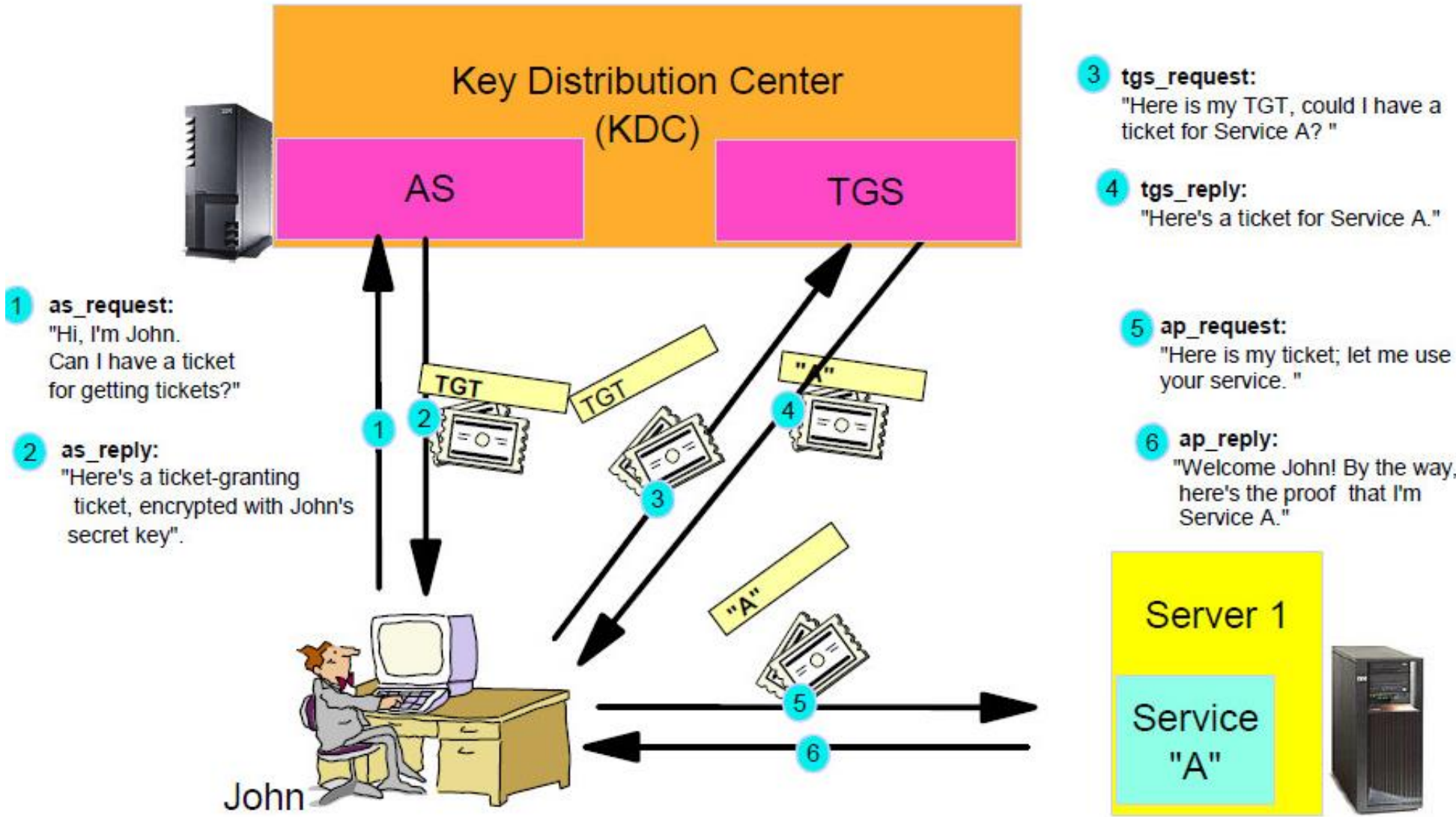


Database : Application Server verify the rights of users(principal) from the database.



Source Wikipedia https://en.wikipedia.org/wiki/Kerberos_(protocol)

Global view


Key Distribution Center
(KDC)
AS
TGS
1 as_request:
"Hi, I'm John.
Can I have a ticket
for getting tickets?"
2 as_reply:
"Here's a ticket-granting
ticket, encrypted with John's
secret key".
3 tgs_request:
"Here is my TGT, could I have a
ticket for Service A? "
4 tgs_reply:
"Here's a ticket for Service A."
5 ap_request:
"Here is my ticket; let me use
your service. "
6 ap_reply:
"Welcome John! By the way,
here's the proof that I'm
Service A."
TGT
TGT
"A"
"A"
1
2
3
4
5
6
John
Server 1
Service
"A"

Advantages

- From Unix to other OS
- Password are never sent in clear
- Token have a limited validity
- SSO

Lightweight Directory Access Protocol (LDAP)

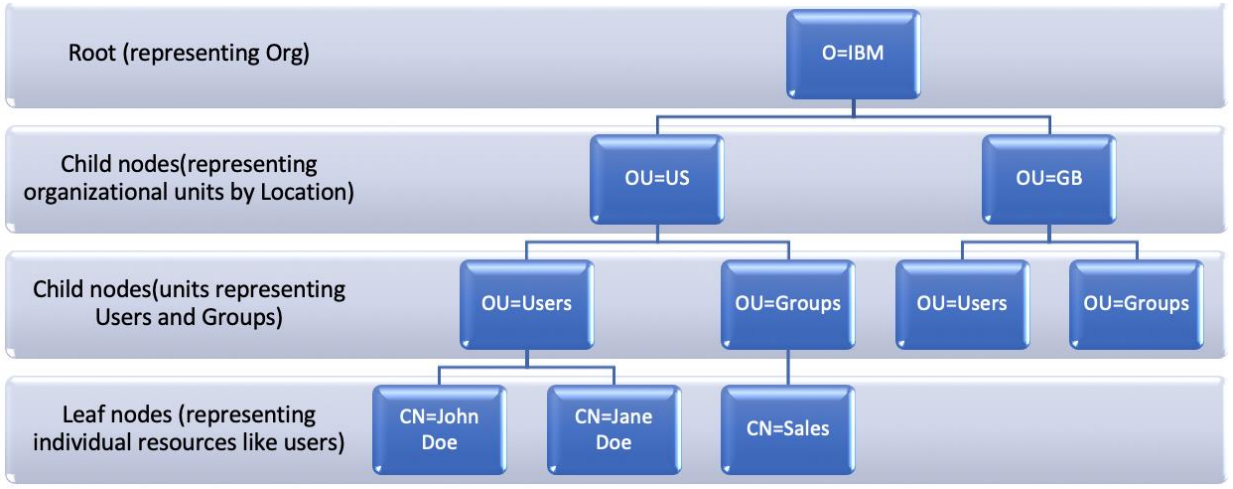
This protocol is suitable for storing data that does not often change and is used to store user information.

AAA* can use LDAP to provide authentication and authorization services for users.

LDAP uses directories to maintain the organization information, personnel information, and resource information. The directories are organized in a tree structure and include entries.

An entry is a sand of attributes with distinguished names (DNs). The attributes are used to store information such as usernames, passwords, emails, computer names, and phone numbers.

LDAP exchanges information in LDAP Data Interchange Format (LDIF).

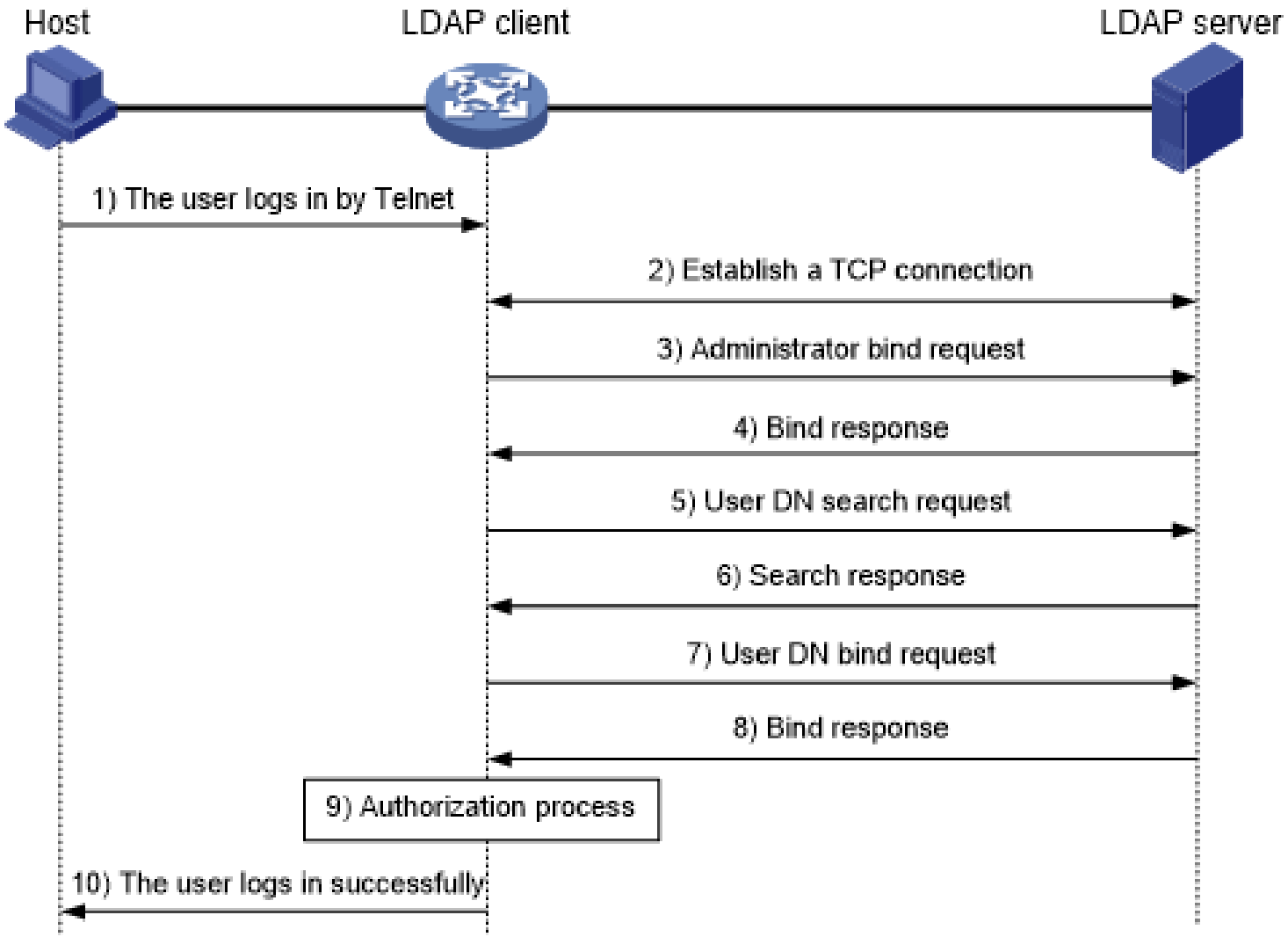


AAA* : Authentication, Authorization and Accounting

Basic LDAP authentication process

1. A Telnet user initiates a connection request and sends the username and password to the LDAP client.
2. After receiving the request, the LDAP client establishes a TCP connection with the LDAP server.
3. To obtain the right to search, the LDAP client uses the administrator DN and password to send an administrator bind request to the LDAP server.
4. The LDAP server processes the request. If the bind operation is successful, the LDAP server sends an acknowledgment to the LDAP client.
5. The LDAP client sends a user DN search request with the username of the Telnet user to the LDAP server.
6. After receiving the request, the LDAP server searches for the user DN by the base DN, search scope, and filtering conditions. If a match is found, the LDAP server sends a response to notify the LDAP client of the successful search. There might be one or more user DNs found.
7. The LDAP client uses the obtained user DN and the entered user password as parameters to send a user DN bind request to the LDAP server. The server will check whether the user password is correct.
8. The LDAP server processes the request, and sends a response to notify the LDAP client of the bind operation result. If the bind operation fails, the LDAP client uses another obtained user DN as the parameter to send a user DN bind request to the LDAP server. This process continues until a DN is bound successfully or all DNs fail to be bound. If all user DNs fail to be bound, the LDAP client notifies the user of the login failure and denies the user's access request.
9. The LDAP client saves the user DN that has been bound and exchanges authorization packets with the authorization server.
    1. If LDAP authorization is used, see the authorization process shown in next slide
    2. If another method is expected for authorization, the authorization process of that method applies.
10. After successful authorization, the LDAP client notifies the user of the successful login.

Basic LDAP authentication process



Basic LDAP autorization process

1.A Telnet user initiates a connection request and sends the username and password to the device. The device will act as the LDAP client during authorization.

2.After receiving the request, the device exchanges authentication packets with the authentication server for the user:

1. If LDAP authentication is used, see the authentication process shown in previous slide
    1. If the device (the LDAP client) uses the same LDAP server for authentication and authorization, skip to step 6.
    2. If the device (the LDAP client) uses different LDAP servers for authentication and authorization, skip to step 4.
2. If another authentication method is used, the authentication process of that method applies. The device acts as the LDAP client. Skip to step 3.

3.The LDAP client establishes a TCP connection with the LDAP authorization server.

4.To obtain the right to search, the LDAP client uses the administrator DN and password to send an administrator bind request to the LDAP server.

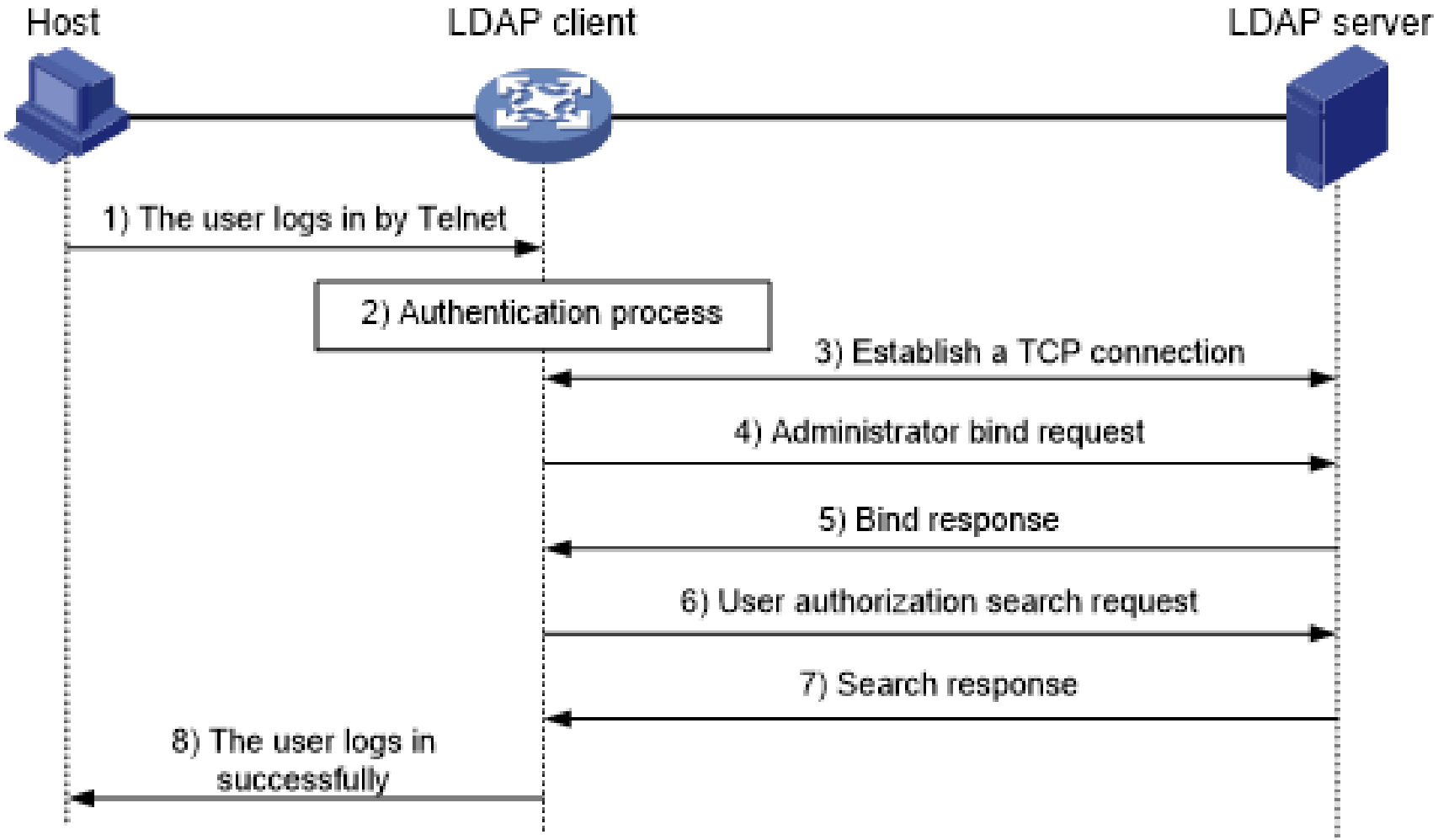
5.The LDAP server processes the request. If the bind operation is successful, the LDAP server sends an acknowledgment to the LDAP client.

6.The LDAP client sends an authorization search request with the username of the Telnet user to the LDAP server. If the user uses the same LDAP server for authentication and authorization, the client sends the request with the saved user DN of the Telnet user to the LDAP server.

7.After receiving the request, the LDAP server searches for the user information by the base DN, search scope, filtering conditions, and LDAP attributes. If a match is found, the LDAP server sends a response to notify the LDAP client of the successful search.

8.After successful authorization, the LDAP client notifies the user of the successful login.

Basic LDAP authorization process



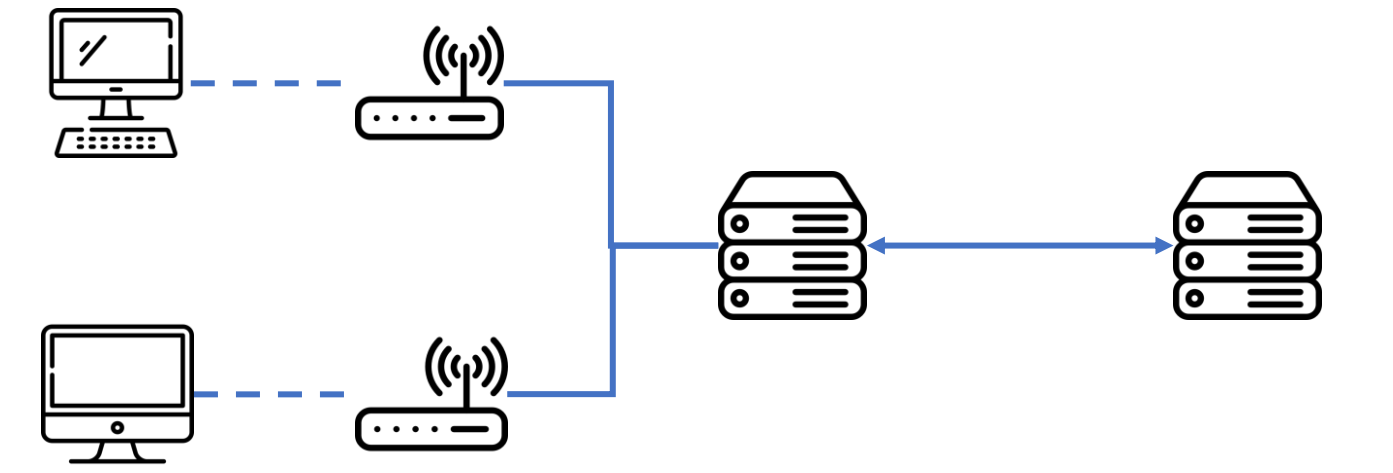
AAA Protocols

- Radius
- TACACS +
- Diameter
- …

Remote Authentication Dial-In User Service (RADIUS)

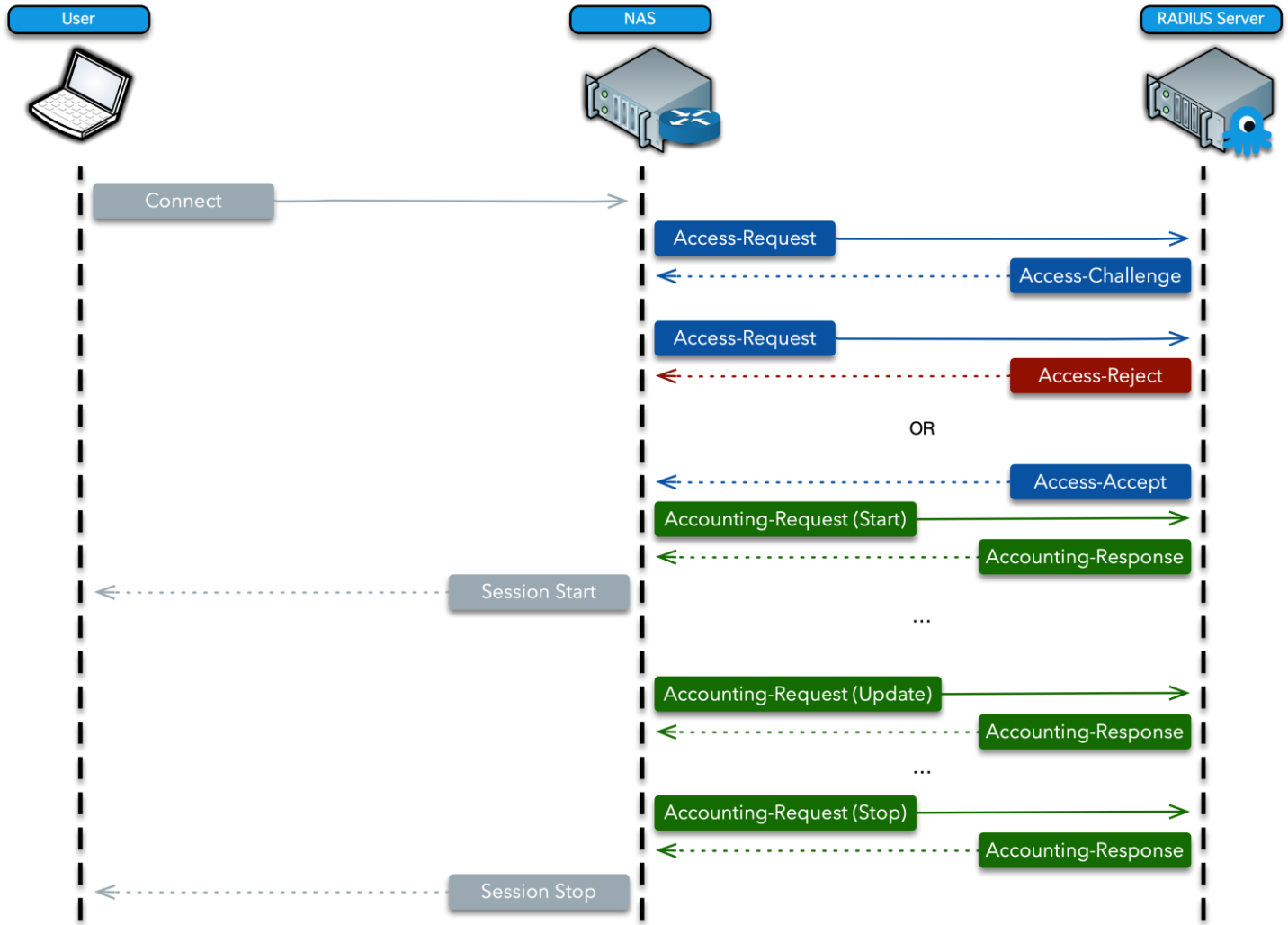
- AAA security framework
- is a client/server protocol that runs in the application layer.
- Provides centralized access to a network
- provides centralized authentication, authorization, and accounting management for users who connect and use a network service.
- Client component may be contained in network access servers (NAS), the gateways that control access to a network, to communicate with the RADIUS server.
- cannot deal effectively with remote access, IP mobility and policy control
- Encrypts only the user password
- Uses UDP
- Historically,
  - Was used for **remote** dial-in connections
  - servers checked the user's information against a locally stored flat file database.
- Modern RADIUS servers
  - can be **remote** and **local**
  - can refer to external sources—commonly SQL, Kerberos, LDAP, or Active Directory servers—to verify the user's credentials.

RADIUS



The RADIUS server only returns one of three access responses to the NAS (middleman) :

1) Reject
2) Challenge
3) Accept



Diameter

- Diameter :
  - Is an AAA **protocol**
  - belongs to the **application layer**
  - uses **TCP** and **SCTP** (RFC 4960)
  - security is provided by **IPsec** or **TLS**

- Enhancement of
  - Radius
  - SS7 for cellular networks > 3G
- Diameter Applications extend the base protocol by adding new commands and/or attributes, such as those for use with the Extensible Authentication Protocol (EAP).
- The Diameter protocol defines a policy protocol used by clients to perform policy, AAA, and resource control. This allows a single server to handle policies for many services.

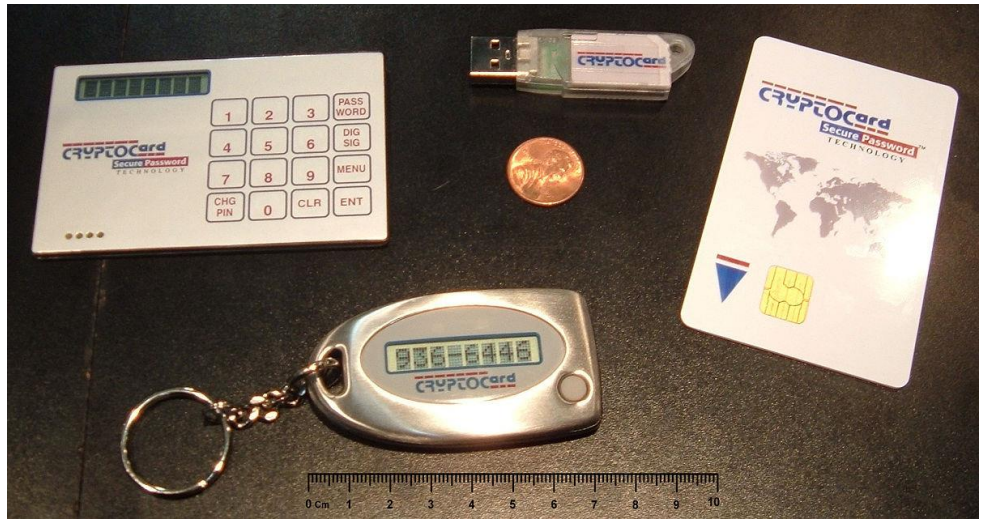
Terminal Access Controller Access-Control System Plus (TACACS+)

- TACACS+
  - AAA security framework
  - uses TCP
  - does not have to implement transmission control (since TCP is a connection oriented protocol
  - encrypts all the information (username, authorization, accounting)
- RADIUS encrypts only the users' password as it travels from the RADIUS client to RADIUS server. TACACS+ mentioned above and therefore does not have the vulnerabilities present in the RADIUS protocol.
- TACACS+ is a CISCO designed extension to TACACS that encrypts the full content of each packet.

Device authentication factors

- MDM Mobile Device Management
- BYOD Bring Your Own Device
- CYOD Choose Your Own Device
- COPE Corporate Owned Personally Enabled

Backup


CRYPTOCard
Secure Password
TECHNOLOGY
1
2
3
PASS WORD
4
5
6
DIG SIG
7
8
9
MENU
CHG PIN
0
CLR
ENT
0 Cm 1 2 3 4 5 6 7 8 9 10

Cookies

- GDPR

Capcha

- Identification between Human and machine

Smart card

VPN

SSL/TLS

Dictionnary attack

- UNIX/Linux

Multiple channels

- Confirmation by SMS