Overview of Identity and Access Management(IAM) Architecture

2019. 5. 7

Contents

- Terminology
- Single Domain Network Access Control Models
  - RADIUS and of its ilk
  - Kerberos
- Multiple Domains Network Access Control Model
  - Single Sign-On (SSO)
  - Federated Identity Management (FIM)

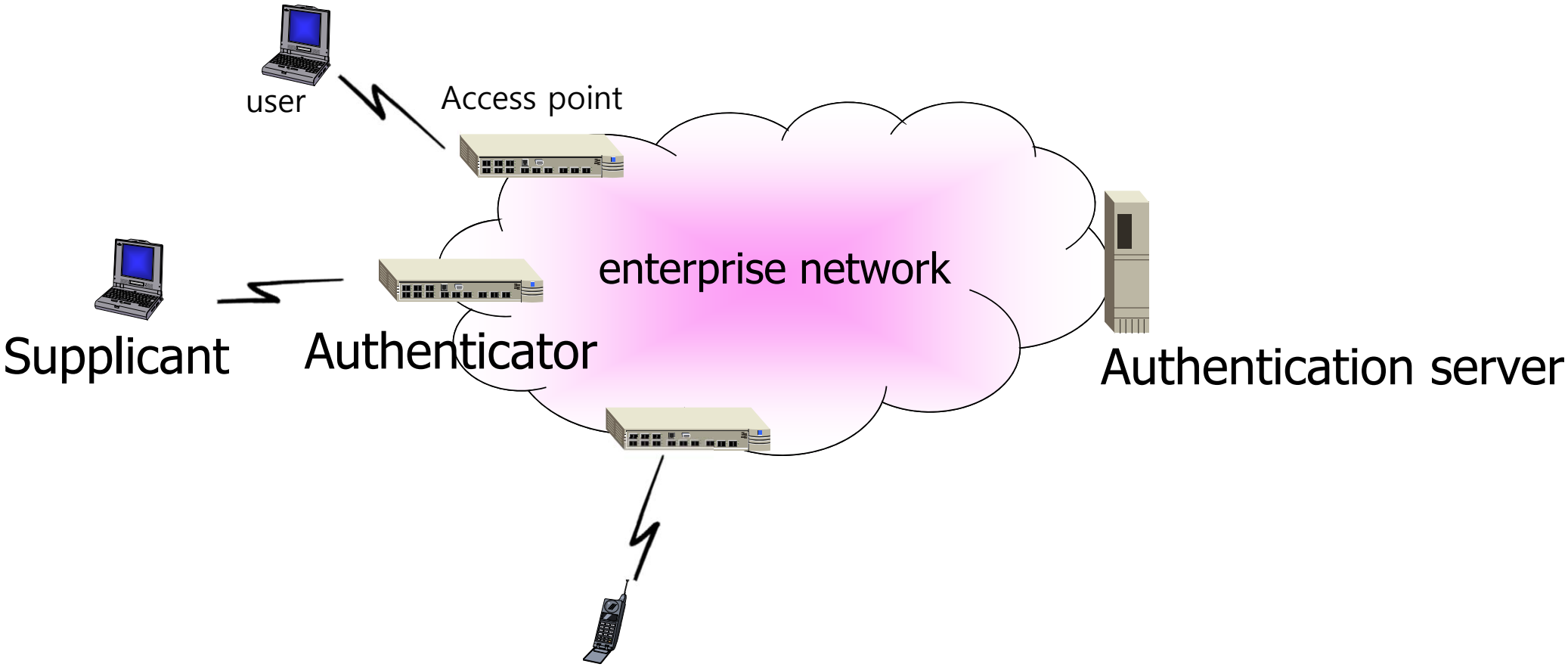
Terminology

- Access Control
  - Authentication
    - Verify whether a user is a real legitimate user, i.e., the user to be claimed by a identity shown
    - Credential ::= (identity, authentication information)
    - Authentication information ::= (password or crypto key or others)
  - Authorization
    - Verify which resources an authenticated user can have rights to access
    - Decide the level of resources which a user have a permission to access
    - Credential ::= (identity, authentication information, attributes)
    - Attribute ::= (resource object, user role, …)

- Authentication/authorization mechanism
  - Which credentials are used to authenticate and in which way
  - Which attributes are used for authorization and in which way
  - Protocols, technologies, mostly vendor dependent
- Authentication/authorization architecture (model)
  - Which components are involved in authentication/authorization
  - what is the procedure for authentication/authorization

- In the following slides, the focus is on authentication for the sake of simplicity of explanation. It can be also expanded to authorization.
- We also target an authentication architecture and procedures between its components.

Single Domain Remote Access Control model



Motivation

- Users are geographically dispersed and want to access a cooperated network.
- Local access points(AP) delegate authentication decision to a central authentication server which has all the credentials of users, managing a single database of users, instead of each of APs maintaining the authentication database.
- Components
  - Supplicant
  - Authenticator
  - Authentication server(AS)

Related Protocols

- Remote authentication protocols
  - RADIUS(Remote Authentication Dial In User Service)
  - DIAMETER
  - TACACS/TACACS+
- Protocols of conveying authentication information, i.e., encapsulating authentication messages such as TLS messages
  - EAP
  - PEAP
  - PANA

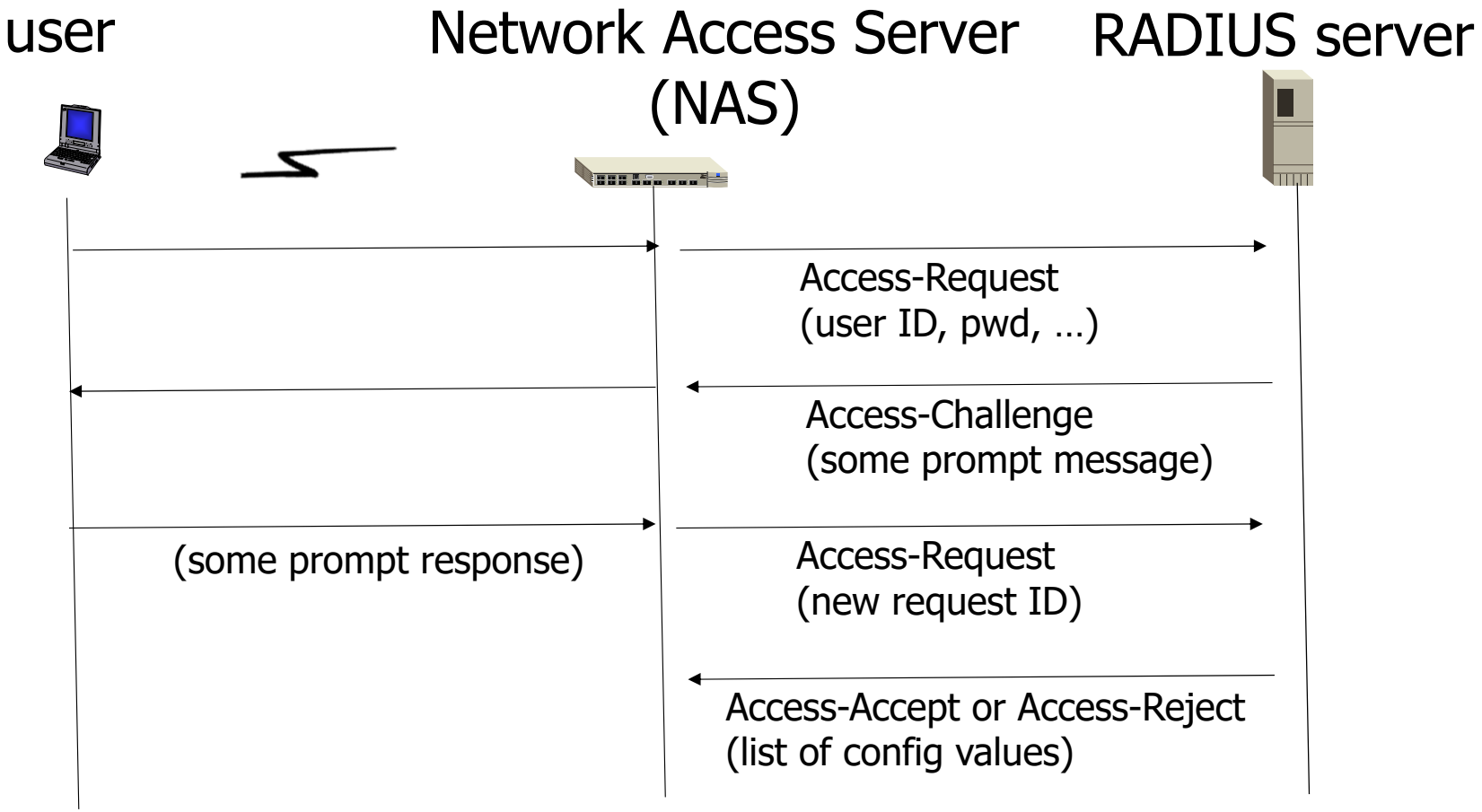
RADIUS

- Motivation
  - When users at serial lines dispersed geographically want to access a cooperated network, managing of users requires careful attention to security, authorization and accounting.
  - This can be best achieved by managing a single database of users.

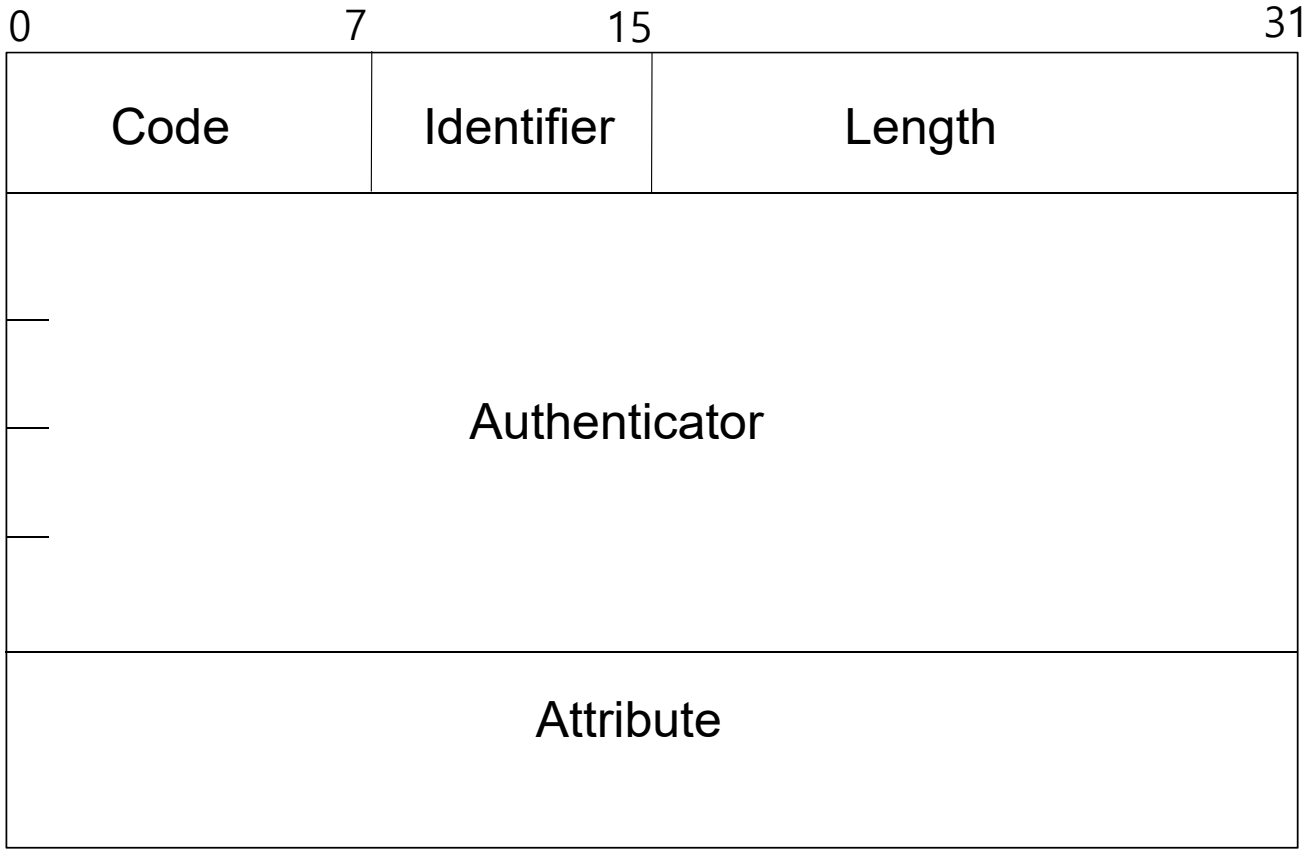
RADIUS model

- Client-sever model
  - Network Access Server (NAS) – Access Point
    - Passing user information to RADIUS servers, and then acting on the response which is returned.
  - RADIUS servers
    - Authentication server (AS)

RADIUS procedure



RADIUS packet format



Code

1 Access-Request
2 Access-Accept
3 Access-Reject
4 Accounting-Request
5 Accounting-Response
11 Access-Challenge
12 Status-Server
13 Status-Client
255 Reserved

Identifier: different for each message

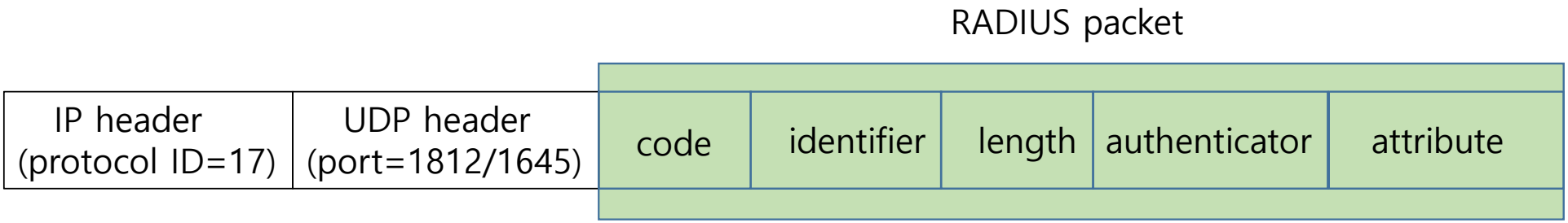
Authenticator:

request packet – nonce
response (Accept, Reject, Challenge) – MD5 hash value

Attribute : TLV

type 1: user name, 79: EAP message 80: EAP message authenticator

RADIUS packet format



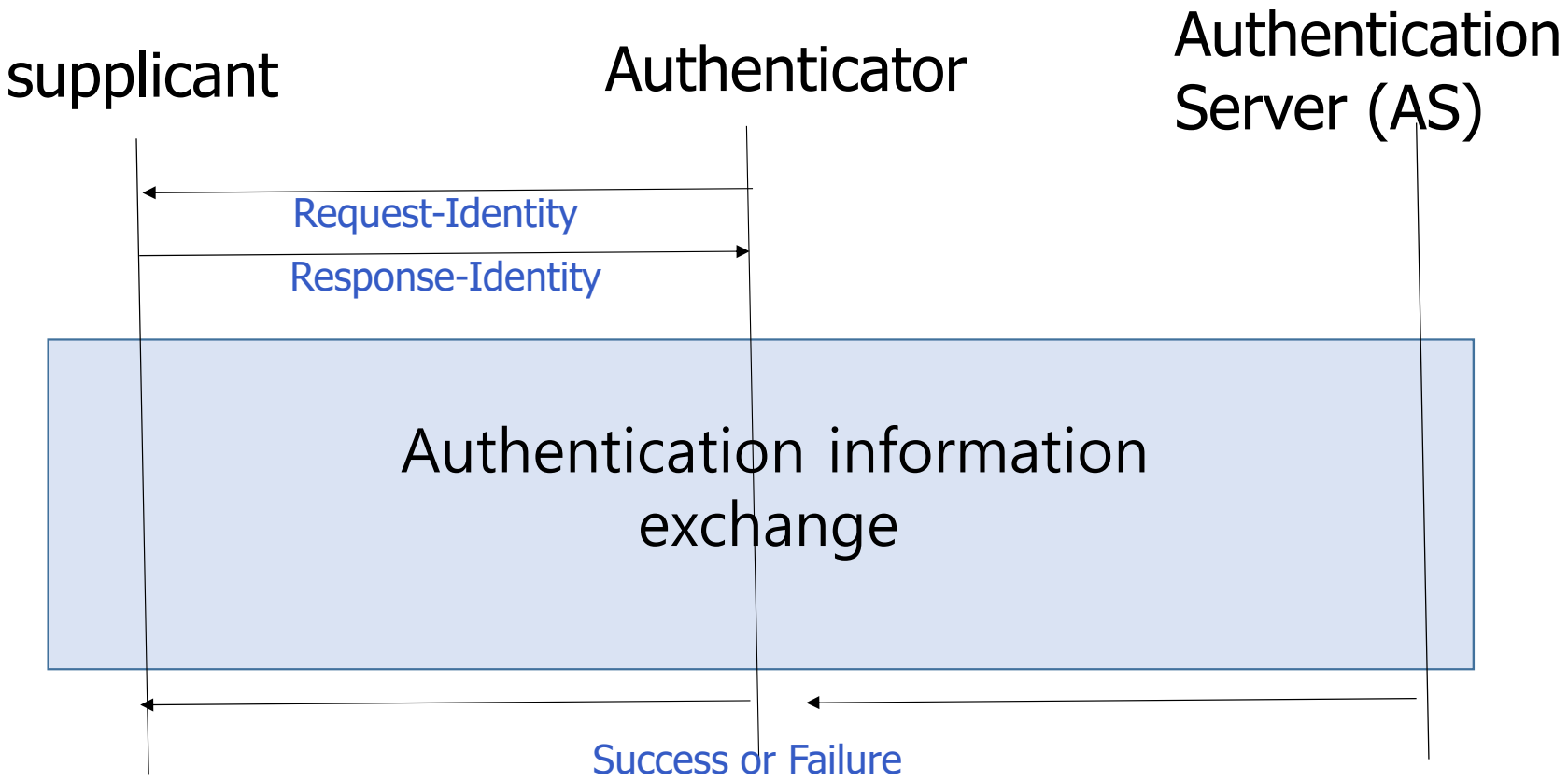
EAP

- EAP is not an authentication method or protocol itself.
- EAP is a framework to support multiple authentication mechanisms over multiple link layer networks.
  - It defines EAP packets which convey data related to a certain authentication method.
  - It defines the procedure to exchange EAP packets for the authentication process.
  - Authenticator do not have to understand each auth method and may act as a pass-through agent for AS.
  - It is independent of any specific link layer technology.

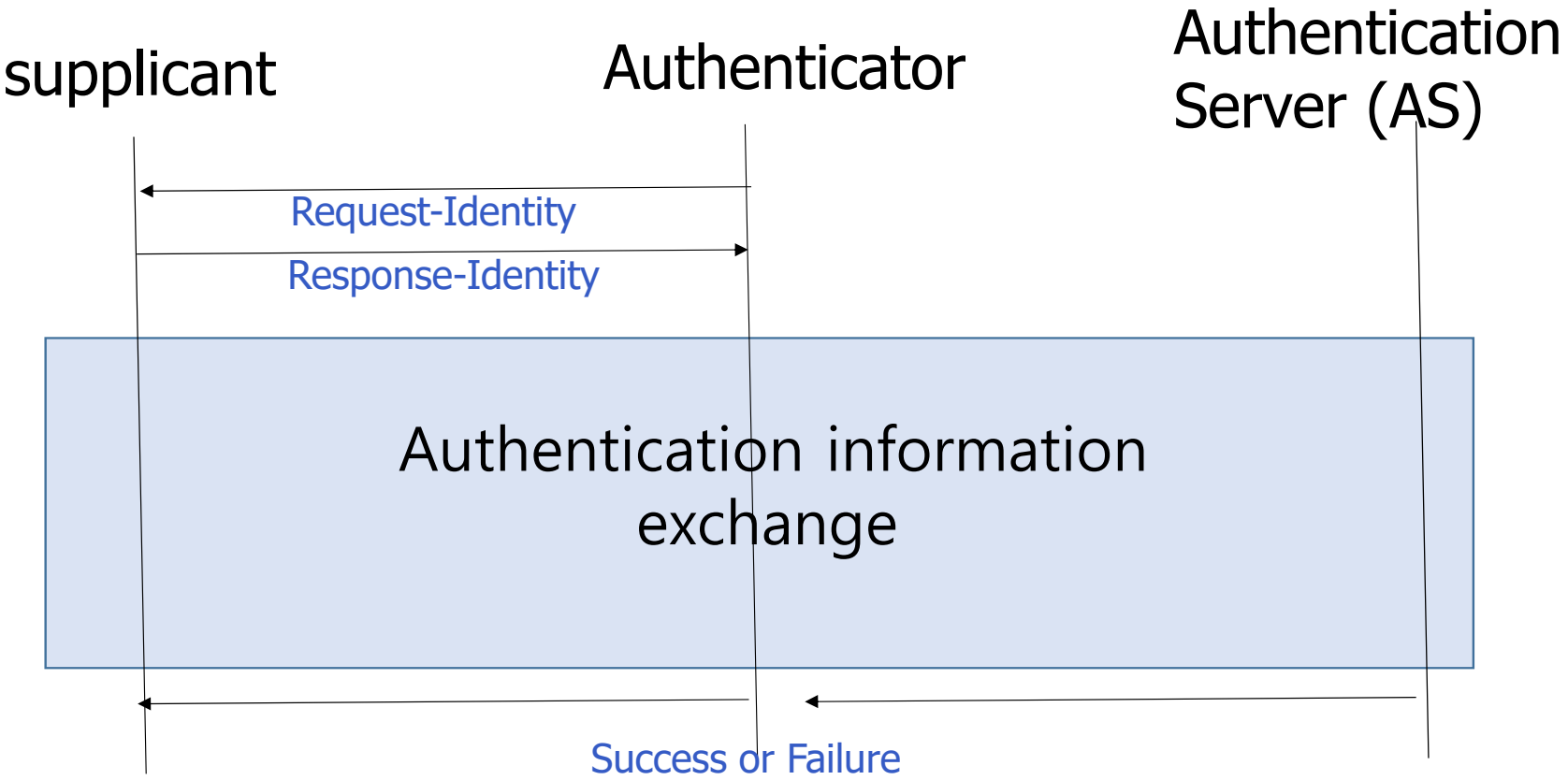
Lower layer under EAP

- EAP assumes that the lower layer is unreliable.
  - EAP defines its own retransmission scheme. The authenticator retransmits Request that have not yet received Responses.
- EAP assumes that the lower layer do error detection.
  - EAP itself does not provide error detection scheme.
- EAP MTU size is 1020 bytes or greater.
- EAP is reliant on lower layer ordering guarantee.

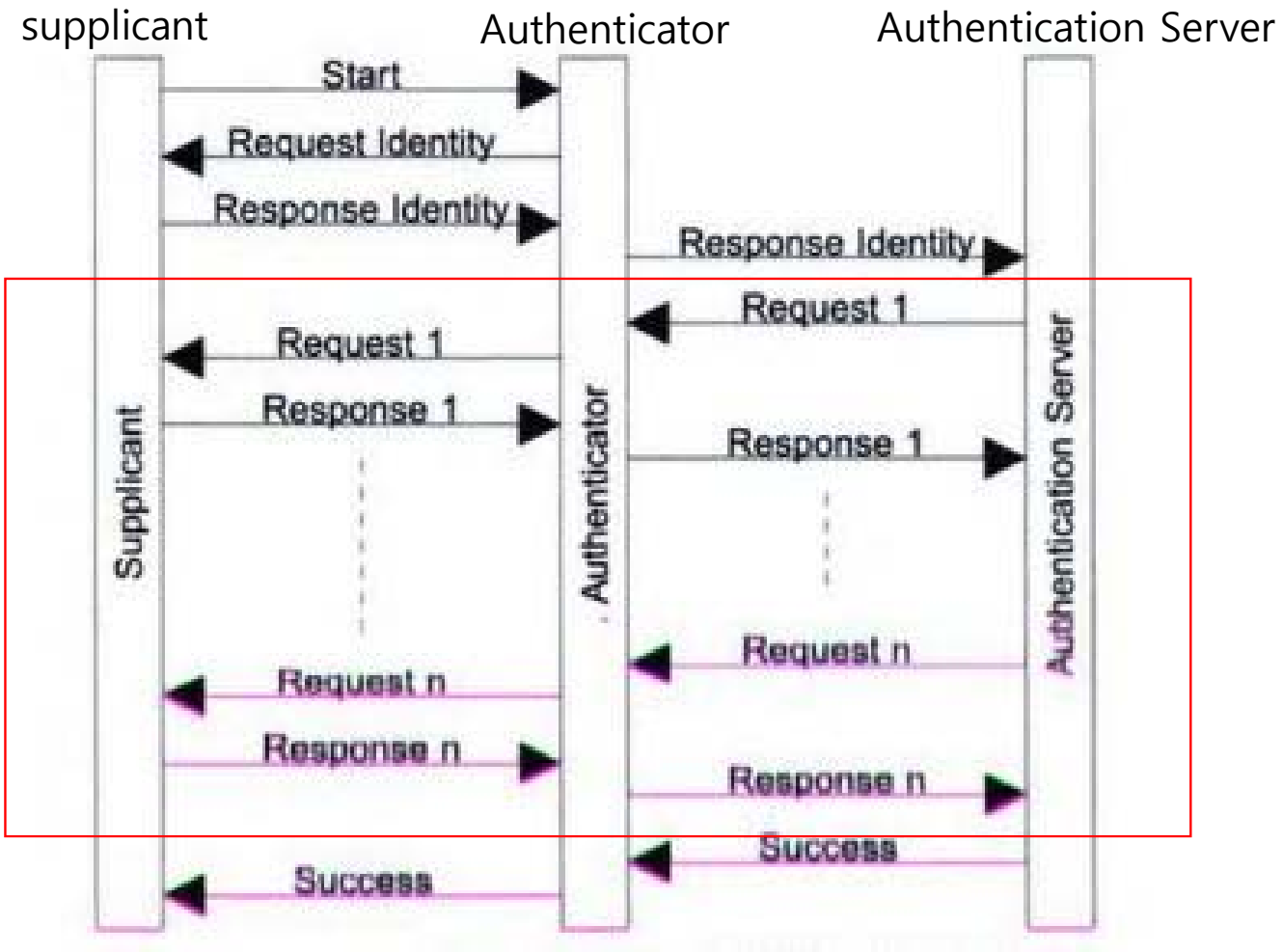
Authentication Process



Authentication Process



EAP message exchange



Kerberos

- In the previous lecture, we introduced Kerberos as an example of the key establishment protocol.
- Kerberos is a protocol that performs two functions: the authentication and session key establishment.
- Before we revisit the Kerberos protocol, let's consider the following case we may experience in our every day life.

When you eat at a food court,

A

B

C

D

E

3. Service request

4. service

Ticket counter

1. Authentication (payment for B)

2. ticket to B

What about doing this way,

First, ask a Ticket for Granting Ticket (TGT) of which validation is for 1 month.

A

B

C

D

E

Ticket counter

1. Authentication
ask TGT

2. TGT 

What about doing this way,

And then, she ask a ticket of a food whenever she is hungry for 1 month.

A

B

C

D

E

3. Service request

4. service

1. Ask Ticket to D with TGT (no further authenticat

Ticket counter

2. TGT 

What about doing this way,

And then, she ask a ticket of a food whenever she is hungry for 1 month.

A

B

C

D

E

3. Service request

4. service

Ticket counter

1. Ask Ticket to C with TGT

2. TGT

Kerberos Server(AS and KDC)

- Kerberos Authentication Server (AS) and Key Distribution Center(KDC)
  - AS and KDC act as the TTP(Trusted Third Party)
  - TTP should be trusted, so it must not be compromised
- AS shares symmetric key K_A with Alice, key K_B with Bob
- And a master key K_{KDC} known ***only*** to KDC
- AS and KDC enables authentication as well as establish session keys
  - Session key for confidentiality and integrity

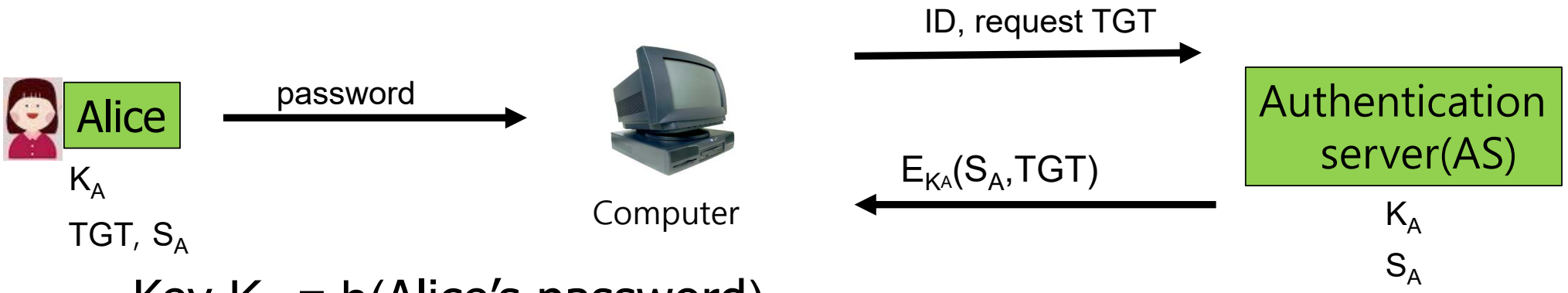
Kerberos Tickets

- AS issues Ticket-Granting Tickets (TGTs) that are used to obtain tickets after it authenticates a client
- Each TGT contains
  - Secret key (not session key)
  - User's ID
  - Expiration time
- Every TGT is encrypted with K_{KDC}
  - So, only TGT can be read by the KDC
- On request, KDC issue tickets containing info needed to access network resources

Login and Get TGT

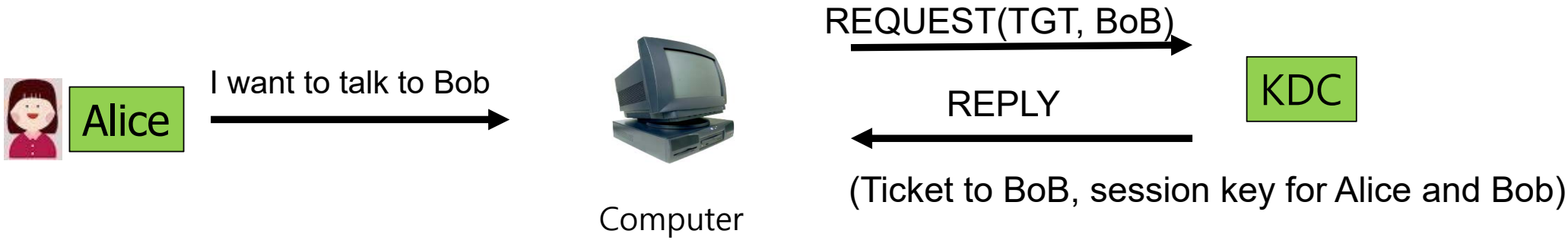
- Alice enters her password
- Then Alice's computer does following:
  - Derives K_A from Alice's password
  - Uses K_A to get TGT for Alice from Authentication Server (AS)
- Alice then uses her TGT (credentials) to securely access network resources
- Getting TGT means that AS authenticates Alice.
- After that, Alice can use TGT to access any service in this domain

Login and Get TGT



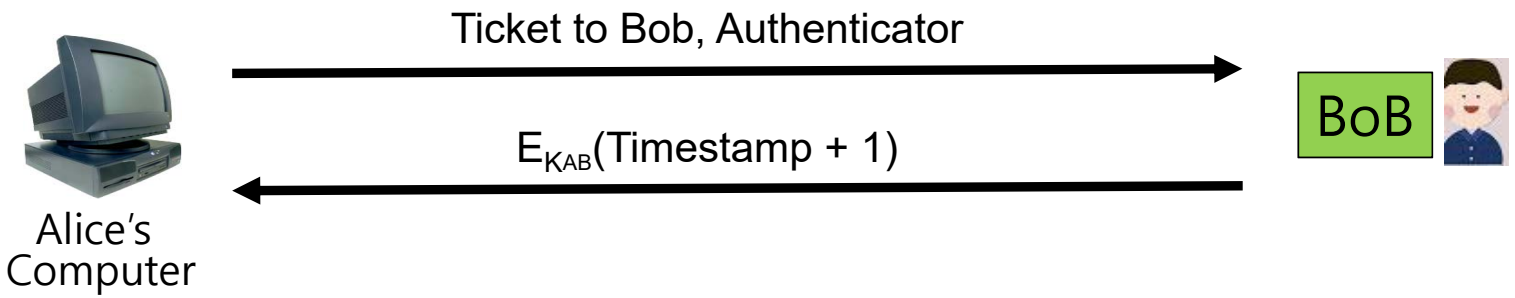
- Key K_A = h(Alice's password)
- AS generates a secret key S_A
- AS generates TGT = $E_{K^{KDC}}$("Alice", S_A)
  - Then it forgets S_A
- Alice's computer decrypts S_A and TGT
  - Then it forgets K_A

Alice Requests "Ticket to Bob"



- REQUEST = (TGT, Authenticator)
  - authenticator = E_{SA}(Timestamp)
- KDC gets S_A from TGT to verify timestamp
- REPLY = E_{SA} ("Bob", K_{AB}, Ticket to Bob)
  - Ticket to Bob = E_{KB}("Alice", K_{AB})
  - K_{AB} : session key between Alice and Bob
  - Then it forgets S_A

Alice Uses Ticket to Bob



- Ticket to Bob = E_{KB}("Alice", K_{AB})
- Authenticator = E_{KAB}(Timestamp)
- Bob decrypts "Ticket to Bob" to get K_{AB} which he then uses to verify timestamp
- Of course Bob trusts Alice. Why?

Remark:

- Detailed information exchanged at each step may be different from the actual Kerberos protocol.
- To clarify two functions, we differentiate an Authentication server and an KDC. But in reality they are working on the same server.
- Key S_A used in authentication for Alice to KDC
- Timestamps for replay protection
  - Reduce the number of messages — like a nonce that is known in advance
  - But, "time" is a security-critical parameter
- Why does KDC use a TGT?
  - KDC doesn't need to remember any information about Alice and Bob.
  - Stateless KDC is major feature of Kerberos

Multiple Domains Network Access Control Model

- Motivation
  - There are many enterprises and numerous applications and many, even millions users to want to access
  - Then every time a user request a service, should it authenticate itself to gain access to the service?
  - Can a user authenticate itself to the first server and then login to any other servers (resources) in another domain without any further authentication process? - the Single Sign-On (SSO).
  - Furthermore, can one user's identity and its associating attributes be recognized as the same user in multiple domains? – the Federated Identity Management (FIM)
  - Realizing FIM means realizing SSO. In that respect, SSO is a subset of FIM.

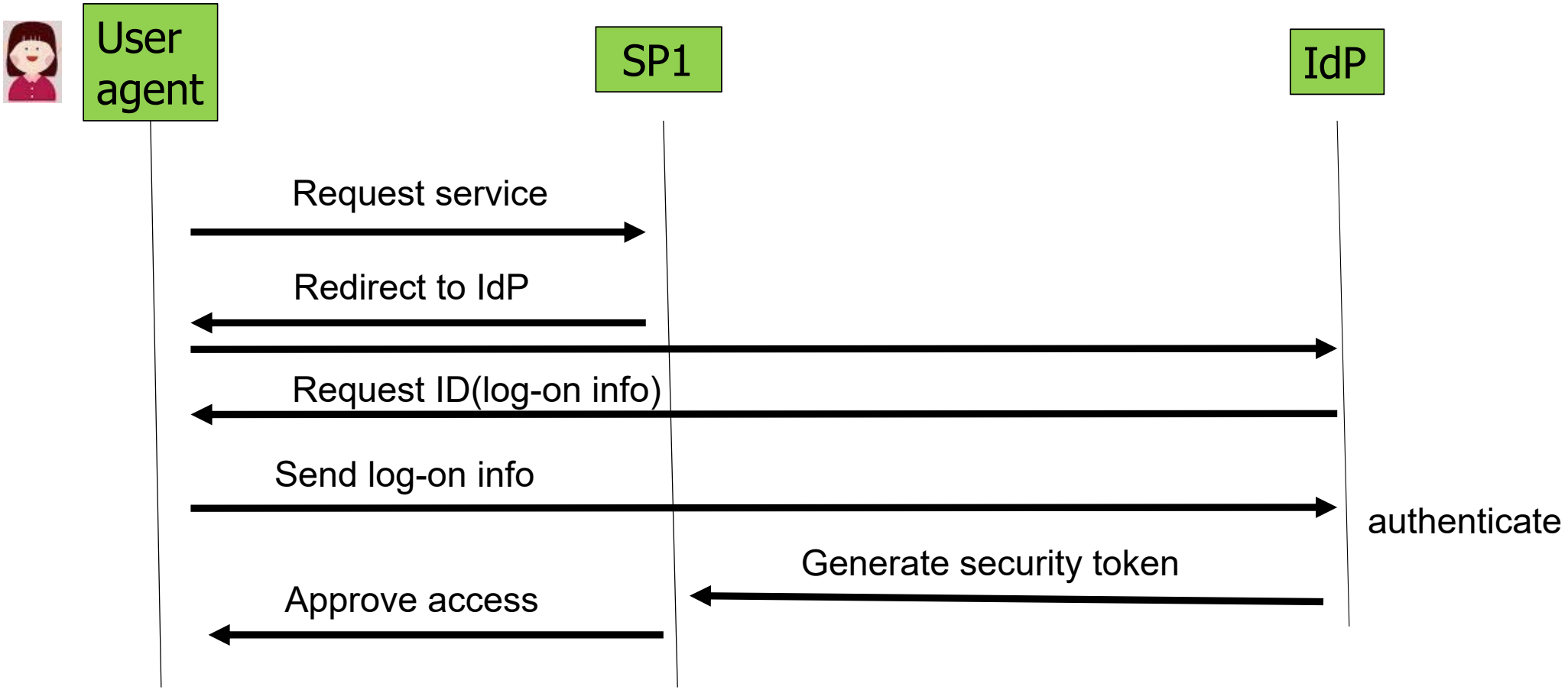
Approach

- Can we expand the ticket concept of Kerberos to multiple domains?
  - When a user wants to access a service, an authentication server(AS) decides if it is legitimate or not.
  - If approved, AS generates a security token which contains user ID, authentication info, and attributes, and others.
  - When the user want to access another service in the federated domain, it use this security token, so gains access without further authentication process.

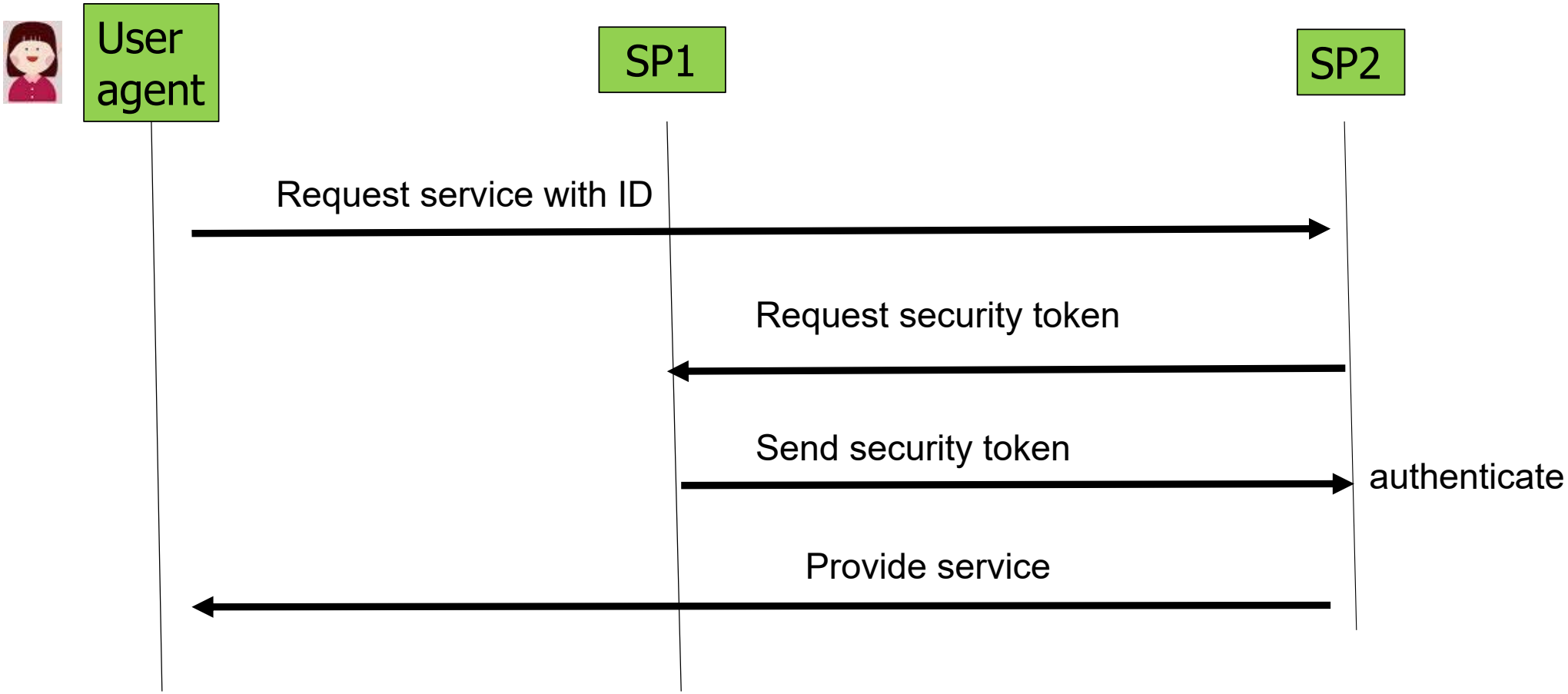
Federated Authentication

- Let's elaborate this idea.
- Components
  - Identity providers(IdP)
    - System (or server) that create and manage identity information, and also provide authentication service to other applications that agree to delegate their authentication requirements to it
  - Service provider (SP)
    - Applications who delegates their authentication requirements to an IdP
  - User agent
    - Software application acting on behalf of a user, such as a browser
  - User
    - Actors who interact with SP and IdP through a user agent

Federated Authentication Flow(1)



Federated Authentication Flow(2)



Use Case 1

- In my university, I mostly have access to three services(sites): gw.mju.ac.kr(groupware including email service), myiweb.mju.ac.kr(administrative service), and lib.mju.ac.kr (library service)
- Once I gain access to one of them, I don’t need any further authentication when accessing another site.
- How can it be possible?

Use Case 2

- Sometimes, when I access a certain site, the site asks me to authenticate me by showing the following messages:
  - Do you want to use Google account?
  - Or do you want to use Facebook account?
  - Or others
- Then, if I enter my google ID and password, can the site authenticate me? Does the site have all the Google user credentials?

User Case 3

- Suppose that I want to buy an air ticket, so I accessed one travel fare aggregator website, eg, Skyscanner.
- Among the fares offered to me, I chose the fare of MyTrip.com and clicked the link.
- So, I moved to MyTrip.com site. But it didn't request any authentication information to me. Does it already know who I am?
- Later, I wanted to rent a car. So, I clicked Herz on the Skyscanner page.
- Now I moved to the Herz site. Surprisingly, it seemed that it already knew who I am, furthermore my purchase of the air ticket, my departure date, and my destination city.
- How can it be possible?

Protocols and Standards

- The previous authentication flow is one possible example, in reality there are many other possible scenarios.
- Should IdP be involved in every authentication SPs?
- How can SPs have agreement for mutual level of trust for users?
  - Identity mapping?
  - Standardized means of representing attributes?
- How can SPs and IdPs exchange these information and security token?

- Is the security token a magic bullet to solve all authentication/authorization problems? Then, what kind of information should be contained in the security token?
- How can the information exchanged securely?
  - How can we generate and establish keys between SPs, or SPs and IdPs, or SPs and user agents?
- And more ….

Popular Protocols

- OpenID
  - A lightweight authentication protocol which uses REST based message flows using JSON web tokens (JWT)
- Security Assertion Markup Language (SAML)
  - XML-based standard for exchanging authentication and authorization data between IdPs and SPs
- OAuth 2.0
- And other vender-oriented protocols