IIC2523
Sistemas Distribuidos

Hernán F. Valdivieso López
(2025 - 2 / Clase 21)

Autorización y Autenticación
¿Qué implicancias tiene este proceso?

Temas de la clase

1. Asegurar identidad y sus desafíos
2. Soluciones a los desafíos de Autenticación
3. Políticas y Mecanismos de Control de Acceso
4. Delegación: Introducción a OAuth 2.1
5. Canal seguro

Autenticación:

Proceso de verificar la identidad declarada de una entidad (usuario, software, dispositivo)

Fuente: OpenAI. (2025). [Definición de Autenticación] [Imagen generada por inteligencia artificial con Sora]. https://openai.com/sora

Asegurar identidad y sus desafíos

Asegurar identidad y sus desafíos

La clase pasada vimos la noción de "autenticarse" mediante 2 formas:

- Firmas digitales mediante llave asimétrica.
  - Encriptar mensaje con llave privada.
  - Solo con la llave pública se puede desencriptar.
  - Garantiza que el mensaje fue creado por su dueño.

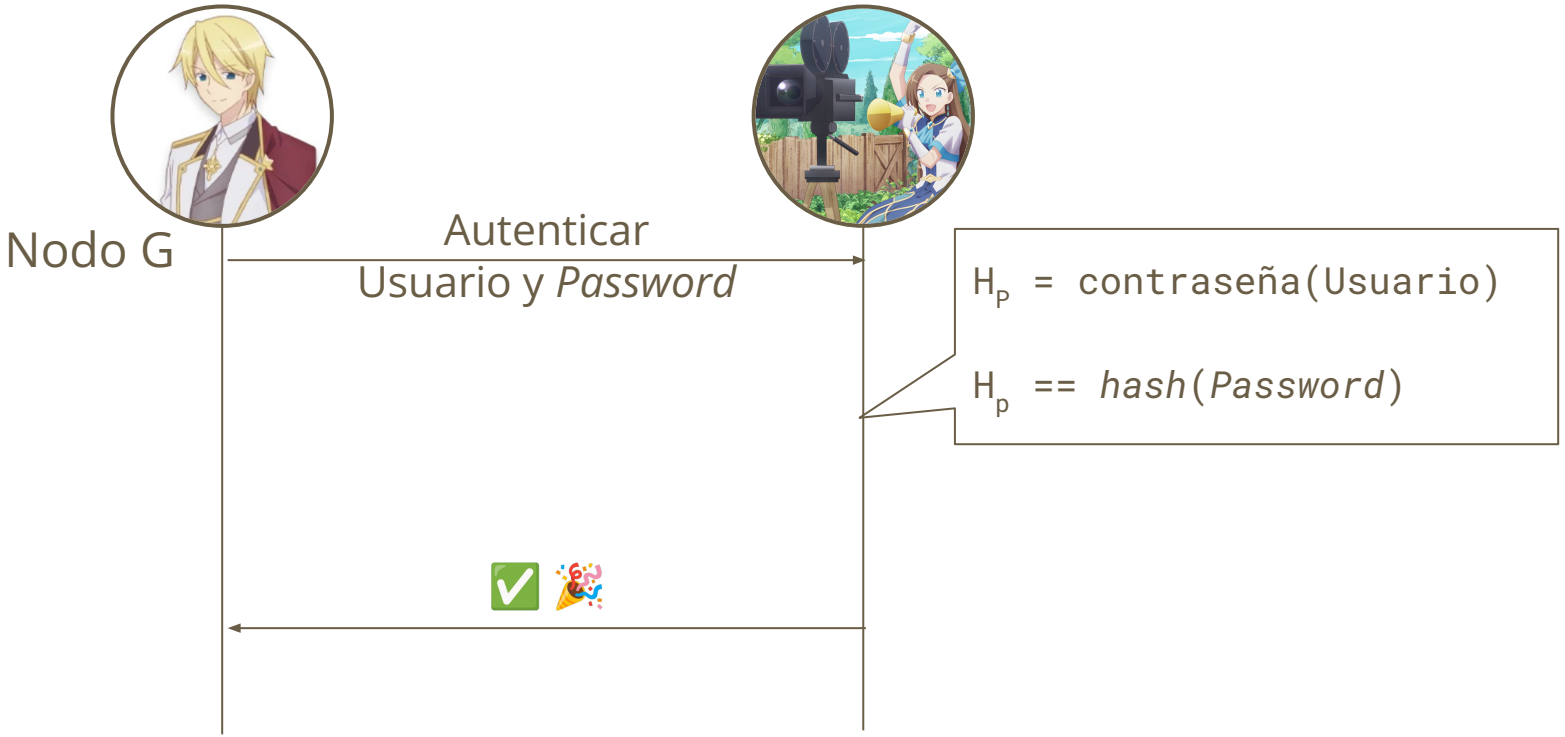
Asegurar identidad y sus desafíos

La clase pasada vimos la noción de "autenticarse" mediante 2 formas:

- Firmas digitales mediante llave asimétrica.
  - Encriptar mensaje con llave privada.
  - Solo con la llave pública se puede desencriptar.
  - Garantiza que el mensaje fue creado por su dueño.
- Firmas digitales mediante función *hash*
  - Aplicar *hash* al documento/mensaje.
  - Encriptar el *hash* con llave privada.
  - Solo con la llave pública se puede desencriptar el *hash*.
  - Verificar el *hash* desencriptado con el *hash* del documento para verificar la integridad y autenticidad del mensaje.

Asegurar identidad y sus desafíos

Existe otras formas → Autenticación simple mediante comparación de *hash*.



Asegurar identidad y sus desafíos

Existe otras formas → *Challenge-response protocol*

Asegurar identidad y sus desafíos

Existe otras formas → *Challenge-response protocol*

- Familia de protocolos que permiten la autenticación de entidades.

1. Una parte (verificador) presenta un desafío.
2. La parte que se quiere autenticar recibe el desafío y elabora una respuesta que envía al verificador.
3. El verificador recibe la respuesta y evalúa si la respuesta responde correctamente a la cuestión, y por tanto la entidad que envió la respuesta queda autenticado.

Asegurar identidad y sus desafíos

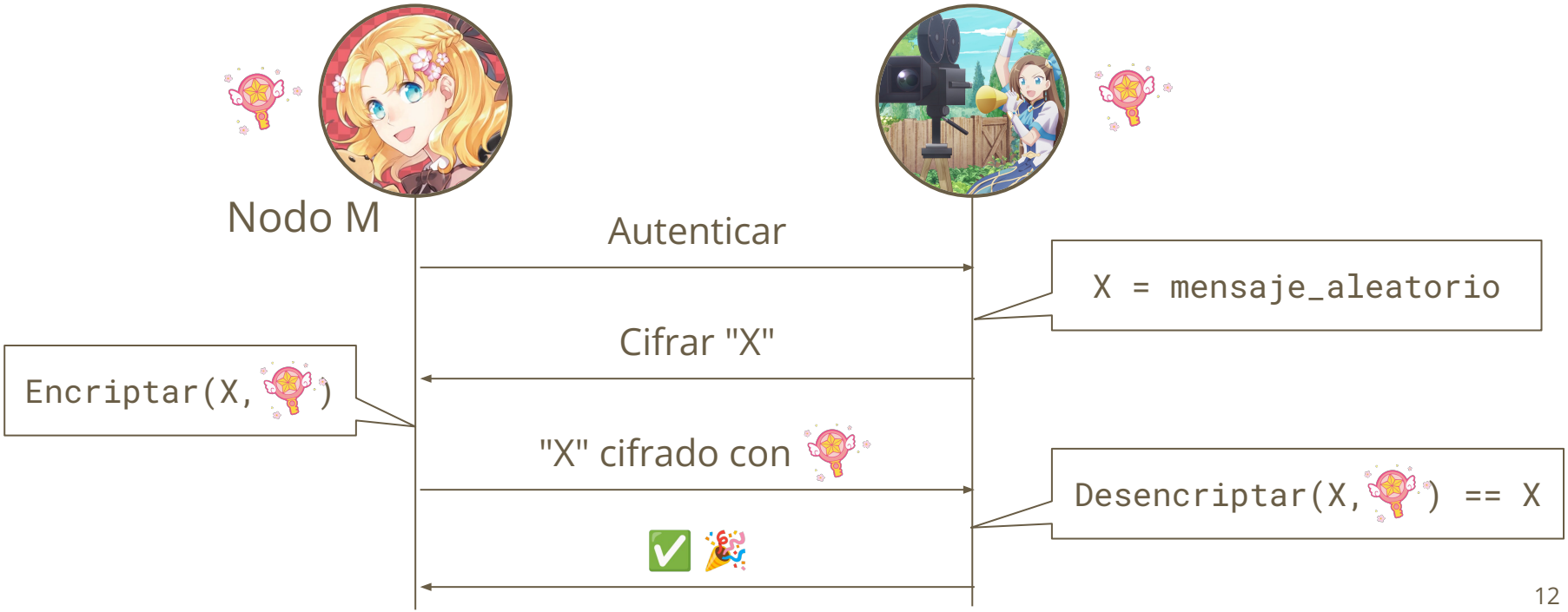
Existe otras formas → *Challenge-response protocol*

- Familia de protocolos que permiten la autenticación de entidades.
  1. Una parte (verificador) presenta un desafío.
  2. La parte que se quiere autenticar recibe el desafío y elabora una respuesta que envía al verificador.
  3. El verificador recibe la respuesta y evalúa si la respuesta responde correctamente a la cuestión, y por tanto la entidad que envió la respuesta queda autenticado.
- Se puede realizar aplicando llave simétrica o asimétrica.
- Otro ejemplo es la contraseña de un solo uso para verificación.

Asegurar identidad y sus desafíos

Existe otras formas → *Challenge-response protocol*

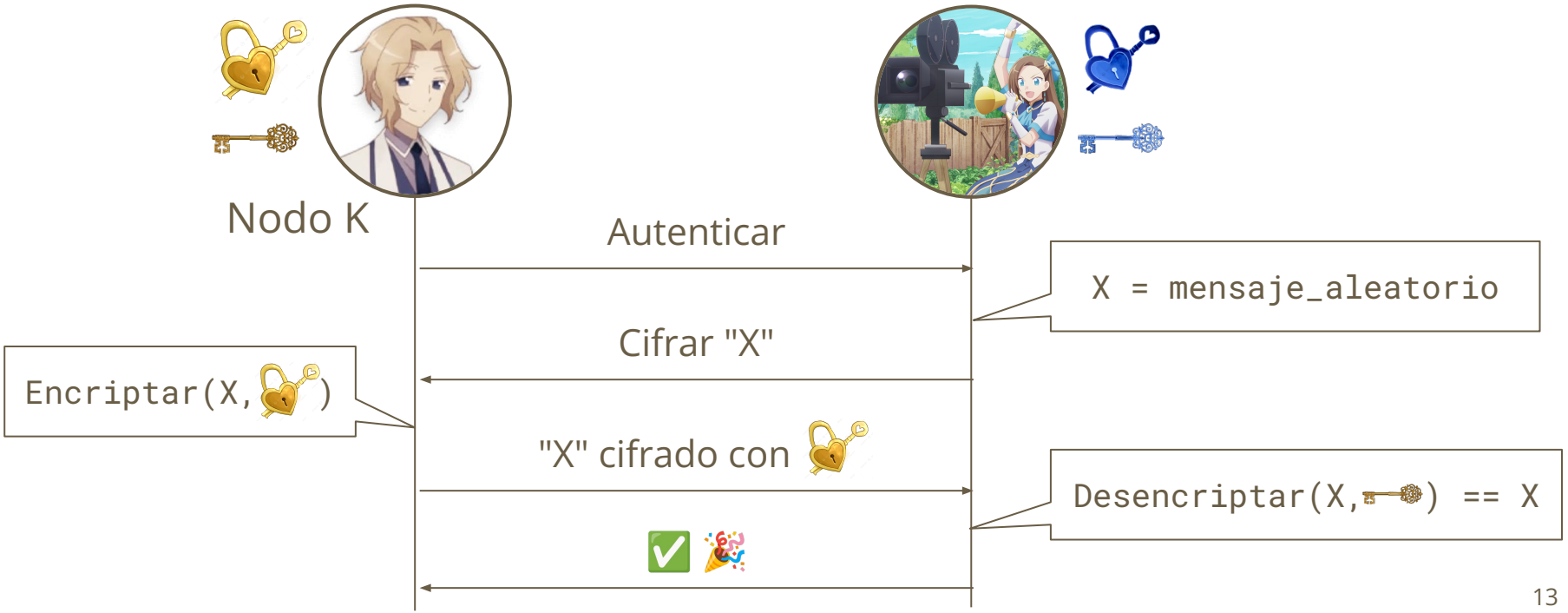
- Se puede realizar aplicando **llave simétrica** o asimétrica.



Asegurar identidad y sus desafíos

Existe otras formas → *Challenge-response protocol*

- Se puede realizar aplicando llave simétrica o **asimétrica**.



Asegurar identidad y sus desafíos

- **Autenticación en ambos sentidos:** no solo el cliente debe autenticarse ante el servidor, sino que el cliente también debe asegurarse de la identidad del servidor.
- **Escalabilidad:** la administración de claves crece a medida que hay más nodos en una red distribuida.
- **Delegación de credenciales:** a veces el usuario desea delegar su identidad o permisos de forma controlada a otro agente sin exponer sus credenciales originales.
- **Expiración**: es necesario tener un control sobre cada cuánto verificar la identidad de una de las partes.

Soluciones a los desafíos de Autenticación

Infraestructura de Clave Pública (PKI)

Kerbero

Infraestructura de Clave Pública (PKI)

- Cada entidad (usuario, servidor o servicio) genera una clave pública y privada.
- Se crea un certificado que valida la identidad de la entidad.
- Existe una Autoridad Certificadora (CA) que firma digitalmente un certificado que vincula una clave pública con la identidad de su dueño.

Infraestructura de Clave Pública (PKI) - Crear

Crear llave privada y pública

Servidor (Juego)

CA

Infraestructura de Clave Pública (PKI) - Crear

Crear y enviar *Certificate Signing Request* (*CSR*) a la Autoridad Certificadora (CA)

CSR

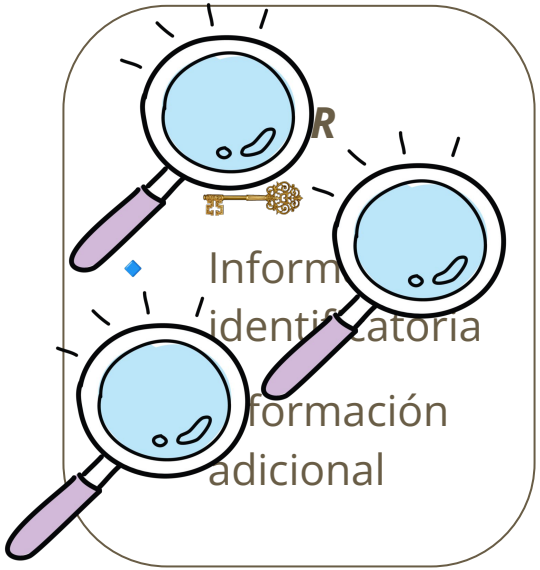
- 
- Información identificatoria
- Información adicional

Infraestructura de Clave Pública (PKI) - Crear

Verificar *Certificate Signing Request* (CSR)

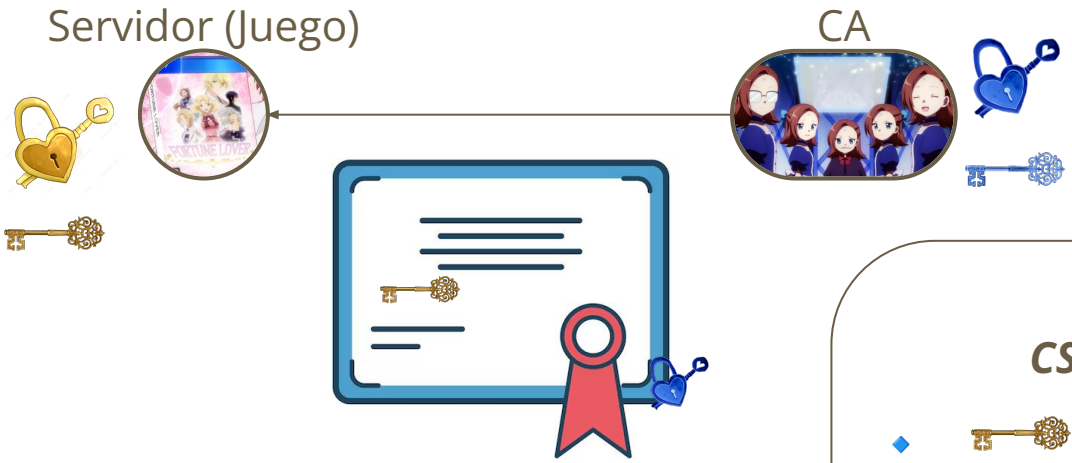
Servidor (Juego)

CA



Infraestructura de Clave Pública (PKI) - Crear

Se firma el contenido del CSR con clave privada y se emite el certificado.

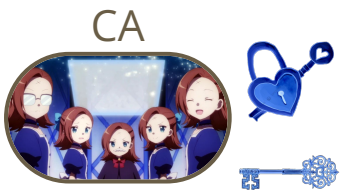


CSR

-
- Información identificatoria
- Información adicional

Infraestructura de Clave Pública (PKI) - Verificar

Cliente recibe certificado

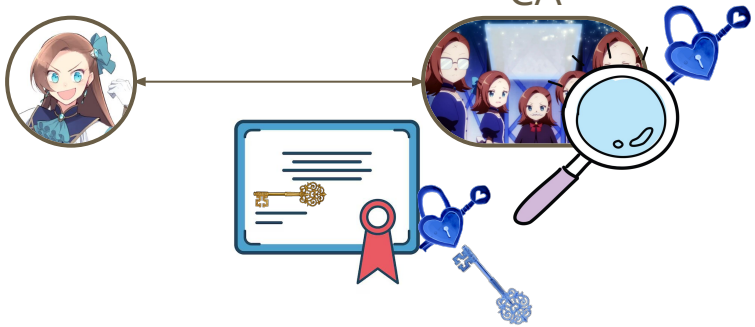


Infraestructura de Clave Pública (PKI) - Verificar

Verifica que certificado sea firmado por CA y que el CA sea confiable

Servidor (Juego)

CA

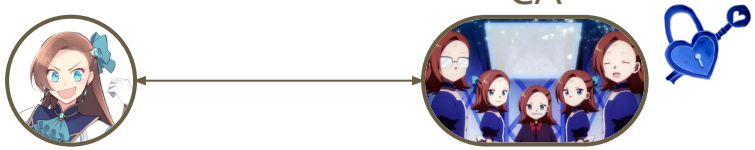


Infraestructura de Clave Pública (PKI) - Verificar

Verifica contenido del certificado (Validez temporal y coincidencia de información)

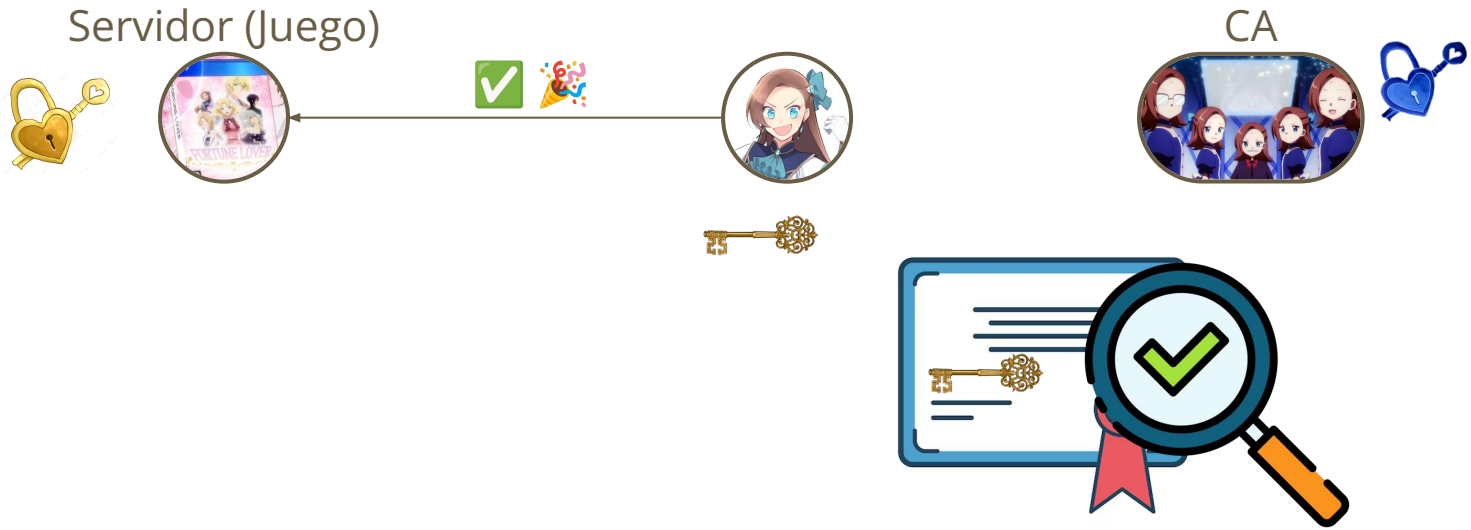
Servidor (Juego)

CA



Infraestructura de Clave Pública (PKI) - Verificar

Extrae llave pública y reconoce la autenticidad del servidor



Infraestructura de Clave Pública (PKI)

Ventaja clave

- Escala eficientemente a muchos nodos: no requiere que cada par de entidades comparta una clave secreta diferente.
- Permite autenticación del servidor (y eventualmente del cliente) sin contacto previo, usando únicamente el certificado firmado por una CA de confianza.

Kerberos – Autenticación centralizada

- Protocolo de autenticación desarrollado en el M.I.T.
- Recurre a criptografía simétrica.
- Requiere un servidor central de confianza (*Key Distribution Center*, KDC).
- Emite tickets temporales para autenticarse ante servicios.

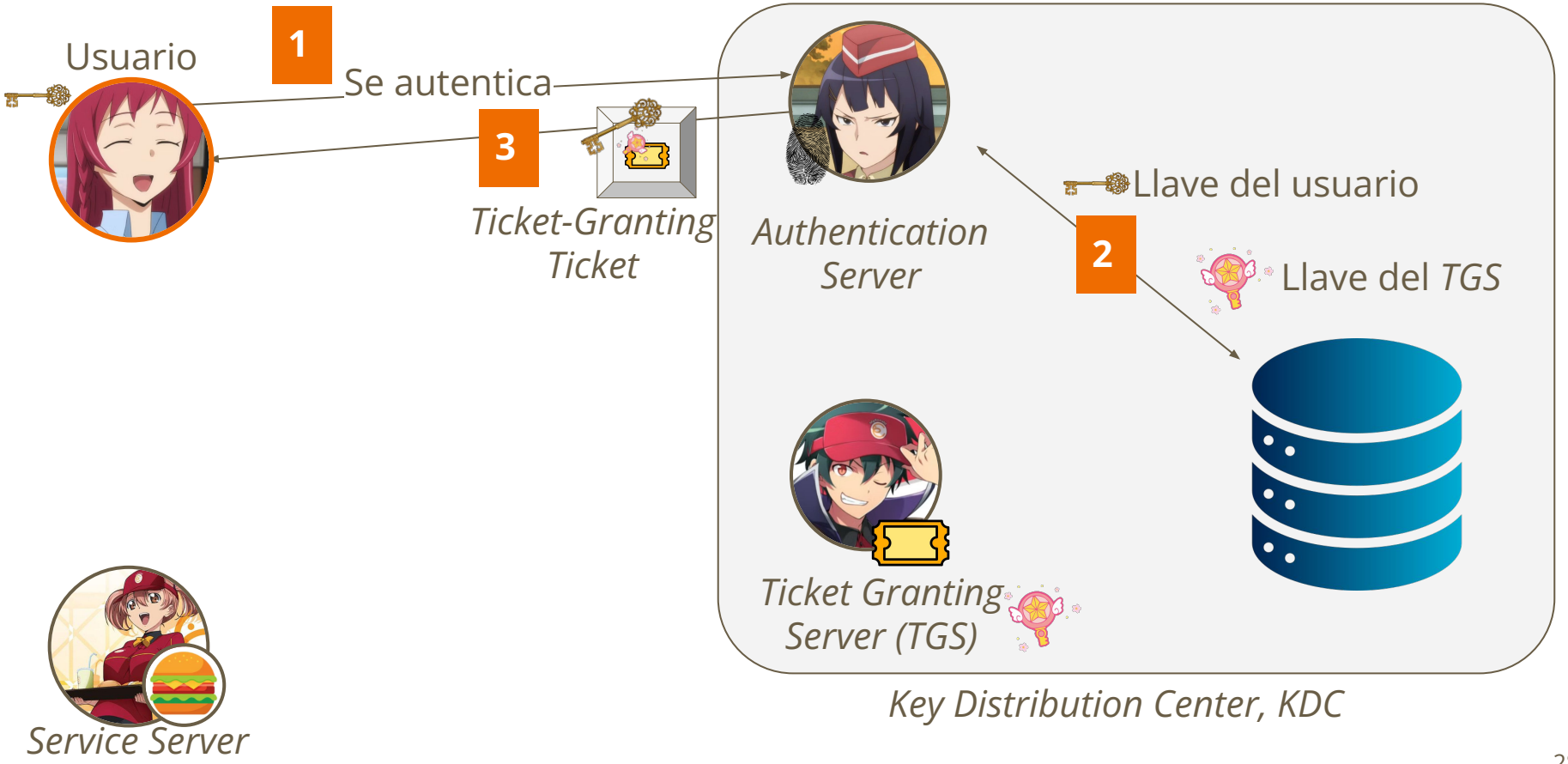
Kerberos – Autenticación centralizada

- Protocolo de autenticación desarrollado en el M.I.T.
- Recurre a criptografía simétrica.
- Requiere un servidor central de confianza (*Key Distribution Center*, KDC)
- Emite tickets temporales para autenticarse ante servicios.

🎫 Tickets

- Permiten que el usuario se autentique múltiples veces sin re-enviar su contraseña.
- Los servicios no necesitan almacenar una llave secreta por usuario, sino que usan las llaves que el sistema entrega.

Kerberos – Componentes y flujo

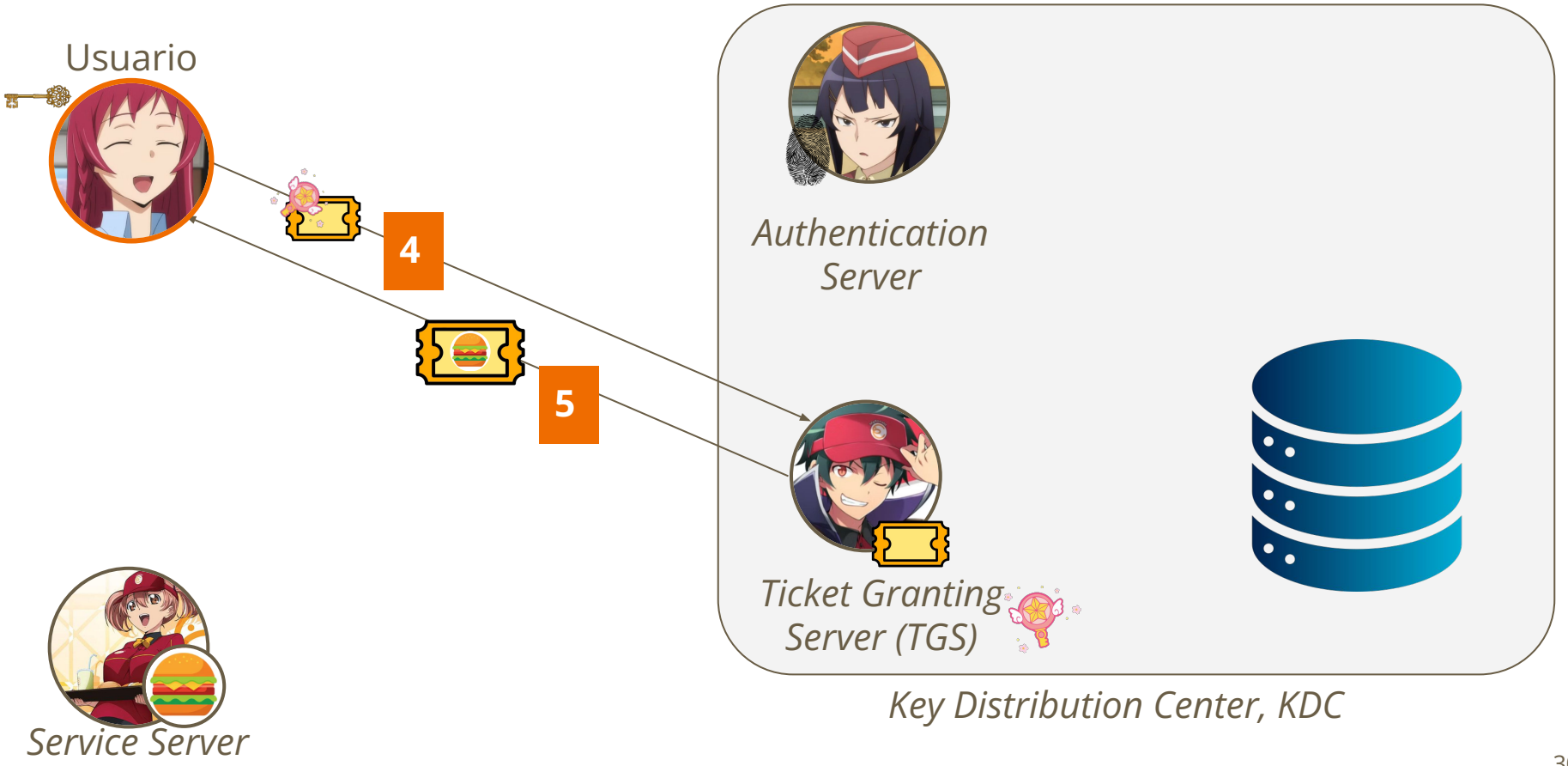


Kerberos – Componentes y flujo (Resumen)

Fase 1: *Authentication Server (AS)*

1. Usuario se autentica ante el *AS*
2. *AS* verifica. Luego busca la llave secreta del usuario y del *Ticket Granting Server (TGS)*.
3. Envía *Ticket Granting Ticket (TGT)* encriptado con la llave del *TGS* y luego con la del usuario.

Kerberos – Componentes y flujo



Kerberos – Componentes y flujo (Resumen)

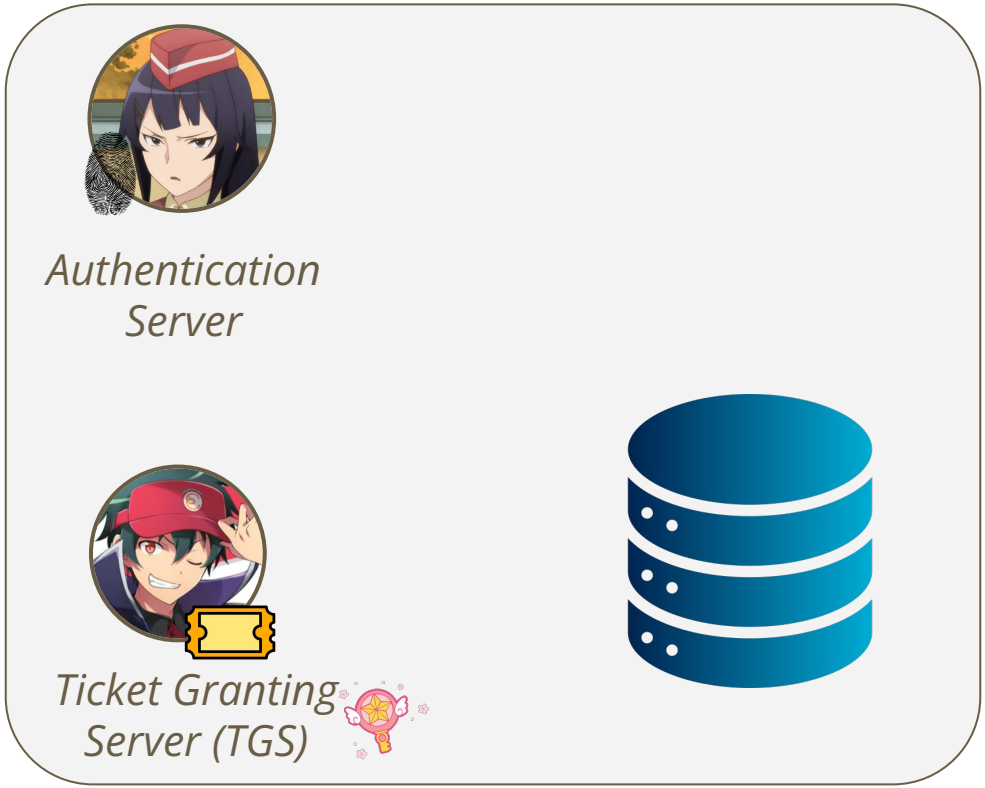
Fase 1: *Authentication Server (AS)*

1. Usuario se autentica ante el *AS*
2. *AS* verifica. Luego busca la llave secreta del usuario y del *Ticket Granting Server (TGS)*.
3. Envía *Ticket Granting Ticket (TGT)* encriptado con la llave del *TGS* y luego con la del usuario.

Fase 2: *Ticket Granting Server (TGS)*

4. Usuario desencripta *TGT* con su llave secreta y envío el resultado al *TGS* junto al servicio solicitado.
5. *TGS* desencripta *TGS* usando su llave secreta, verifica y responde con un *ticket server*.

Kerberos – Componentes y flujo



Kerberos – Componentes y flujo (Resumen)

Fase 1: *Authentication Server (AS)*

1. Usuario se autentica ante el *AS*
2. *AS* verifica. Luego busca la llave secreta del usuario y del *Ticket Granting Server (TGS)*.
3. Envía *Ticket Granting Ticket (TGT)* encriptado con la llave del *TGS* y luego con la del usuario.

Fase 2: *Ticket Granting Server (TGS)*

4. Usuario desencripta *TGT* con su llave secreta y envío el resultado al *TGS* junto al servicio solicitado.
5. *TGS* desencripta *TGS* usando su llave secreta, verifica y responde con un *ticket server*.

Fase 3: *Service Server (SS)*

6. Usuario envía el *ticket server* al servidor que provee el servicio.
7. Servidor verifica el *ticket server* y envía la respuesta correspondiente al servicio que debe entrega.

Kerberos vs PKI

Criterio	PKI	Kerberos
Criptografía	Asimétrica	Simétrica
Escalabilidad	Muy alta	Moderada (centralizada)
Requiere	Autoridad Certificadora Certificados	Key Distribution Center *Tickets*

Políticas y Mecanismos de control de acceso

Políticas

- **MAC, DAC, RBAC, ABAC**

Mecanismos

- **ACL y Capacidades**

Políticas y Mecanismos de control de acceso

- Define quién puede hacer qué sobre qué recurso.
- Existen 4 políticas principales:
  - MAC - Control de Acceso Obligatorio
  - DAC - Control de Acceso Discrecional
  - RBAC - Control Basado en Roles
  - ABAC - Control Basado en Atributos
- Estudiaremos 2 métodos típicos de implementación de dichas políticas
  - ACL - Lista de control de Acceso
  - Capacidades

Tipos de Políticas de Control de Acceso

MAC – Control de Acceso Obligatorio

- Política definida por autoridad central
- Datos con etiquetas: público, secreto, etc.
- Usuarios no pueden cambiar permisos.

DAC – Control de Acceso Discrecional

- El propietario decide permisos.
- Común en sistemas tipo UNIX.
- Cambiable por el dueño del recurso.

RBAC – Control Basado en Roles

- Permisos se asignan a roles (ej: “profesor”)
- Usuarios obtienen permisos según su rol

ABAC – Control Basado en Atributos

- Políticas basadas en atributos (rol, curso, contexto, etc.)
- Permite reglas complejas y dinámicas
- Ejemplo: "profesores del curso X pueden leer notas de sus estudiantes"

Tipos de Políticas de Control de Acceso

MAC – Control de Acceso Obligatorio

- Política definida por autoridad central
- Datos con etiquetas: público, secreto, etc.
- Usuarios no pueden cambiar permisos.

DAC – Control de Acceso Discrecional

- El propietario decide permisos.
- Común en sistemas tipo UNIX.
- Cambiable por el dueño del recurso.

RBAC – Control Basado en Roles

- Permisos se asignan a roles (ej: “profesor”)
- Usuarios obtienen permisos según su rol

ABAC – Control Basado en Atributos

- Políticas basadas en atributos (rol, curso, contexto, etc.)
- Permite reglas complejas y dinámicas
- Ejemplo: "profesores del curso X pueden leer notas de sus estudiantes"

Tipos de Políticas de Control de Acceso

MAC – Control de Acceso Obligatorio

- Política definida por autoridad central
- Datos con etiquetas: público, secreto, etc.
- Usuarios no pueden cambiar permisos.

DAC – Control de Acceso Discrecional

- El propietario decide permisos.
- Común en sistemas tipo UNIX.
- Cambiable por el dueño del recurso.

RBAC – Control Basado en Roles

- Permisos se asignan a roles (ej: “profesor”)
- Usuarios obtienen permisos según su rol

ABAC – Control Basado en Atributos

- Políticas basadas en atributos (rol, curso, contexto, etc.)
- Permite reglas complejas y dinámicas
- Ejemplo: "profesores del curso X pueden leer notas de sus estudiantes"

Tipos de Políticas de Control de Acceso

MAC – Control de Acceso Obligatorio

- Política definida por autoridad central
- Datos con etiquetas: público, secreto, etc.
- Usuarios no pueden cambiar permisos.

DAC – Control de Acceso Discrecional

- El propietario decide permisos.
- Común en sistemas tipo UNIX.
- Cambiable por el dueño del recurso.

RBAC – Control Basado en Roles

- Permisos se asignan a roles (ej: “profesor”)
- Usuarios obtienen permisos según su rol

ABAC – Control Basado en Atributos

- Políticas basadas en atributos (rol, curso, contexto, etc.)
- Permite reglas complejas y dinámicas
- Ejemplo: "profesores del curso X pueden leer notas de sus estudiantes"

Mecanismos de Implementación del Control de Acceso

ACL - Listas de Control de Acceso

- Cada recurso tiene un "diccionario" {usuario: [operaciones permitidas]}
- El servidor:
  - Autentica al usuario
  - Busca en la ACL del recurso
  - Permite o deniega la operación

Capacidades

- El usuario posee "claves de acceso" (ID recurso, operaciones, firma)
- Las capacidades:
  - Son infalsificables.
  - Prueba directa de autorización.
  - No necesita consultar una lista.
  - Costosas de administrar cuando se modifican los accesos del usuario.

Mecanismos de Implementación del Control de Acceso

ACL - Listas de Control de Acceso

- Cada recurso tiene un "diccionario" {usuario: [operaciones permitidas]}
- El servidor:
  - Autentica al usuario.
  - Busca en la ACL del recurso.
  - Permite o deniega la operación.

Capacidades

- El usuario posee "claves de acceso" (ID recurso, operaciones, firma)
- Las capacidades:
  - Son infalsificables.
  - Prueba directa de autorización.
  - No necesita consultar una lista.
  - Costosas de administrar cuando se modifican los accesos del usuario.

Introducción a OAuth 2.1

Introducción a OAuth 2.1

- Delegación de autenticidad consiste en una técnica que permite a una entidad(o un proceso) realizar una acción con la autoridad de otra entidad.

Introducción a OAuth 2.1

- Delegación de autenticidad consiste en una técnica que permite a una entidad(o un proceso) realizar una acción con la autoridad de otra entidad.

OAuth 2.0 (*Open Authorization*):

- Comenzó en noviembre de 2006 con la versión 1.0.
- Protocolo de delegación ampliamente utilizado que permite que las aplicaciones accedan a recursos protegidos en nombre de un usuario, sin necesidad de compartir las credenciales del usuario con la aplicación
  - Iniciar sesión con Google.
  - Iniciar sesión con Github.

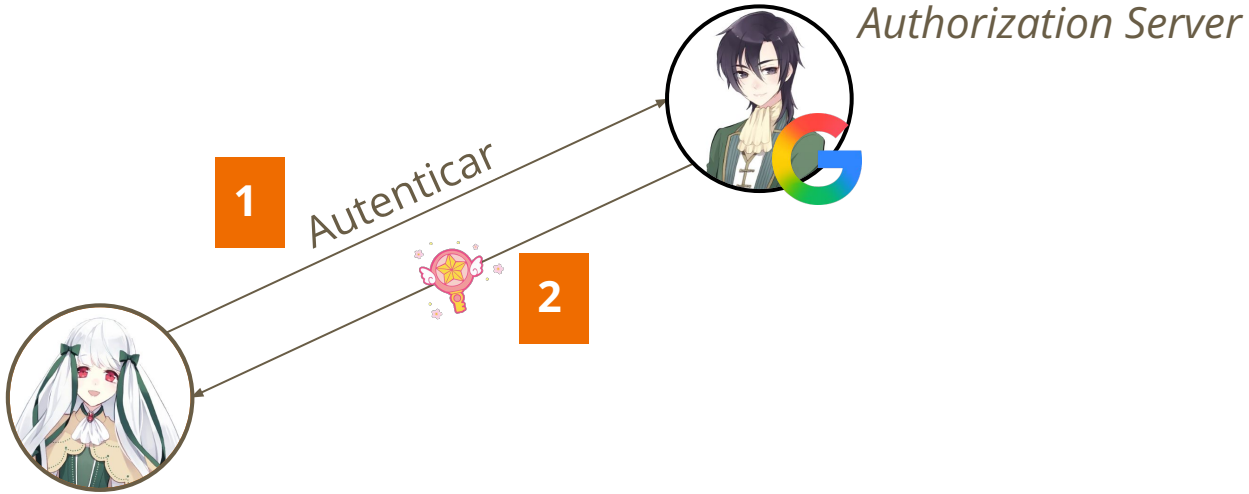
Introducción a OAuth 2.1

- Delegación de autenticidad consiste en una técnica que permite a una entidad(o un proceso) realizar una acción con la autoridad de otra entidad.

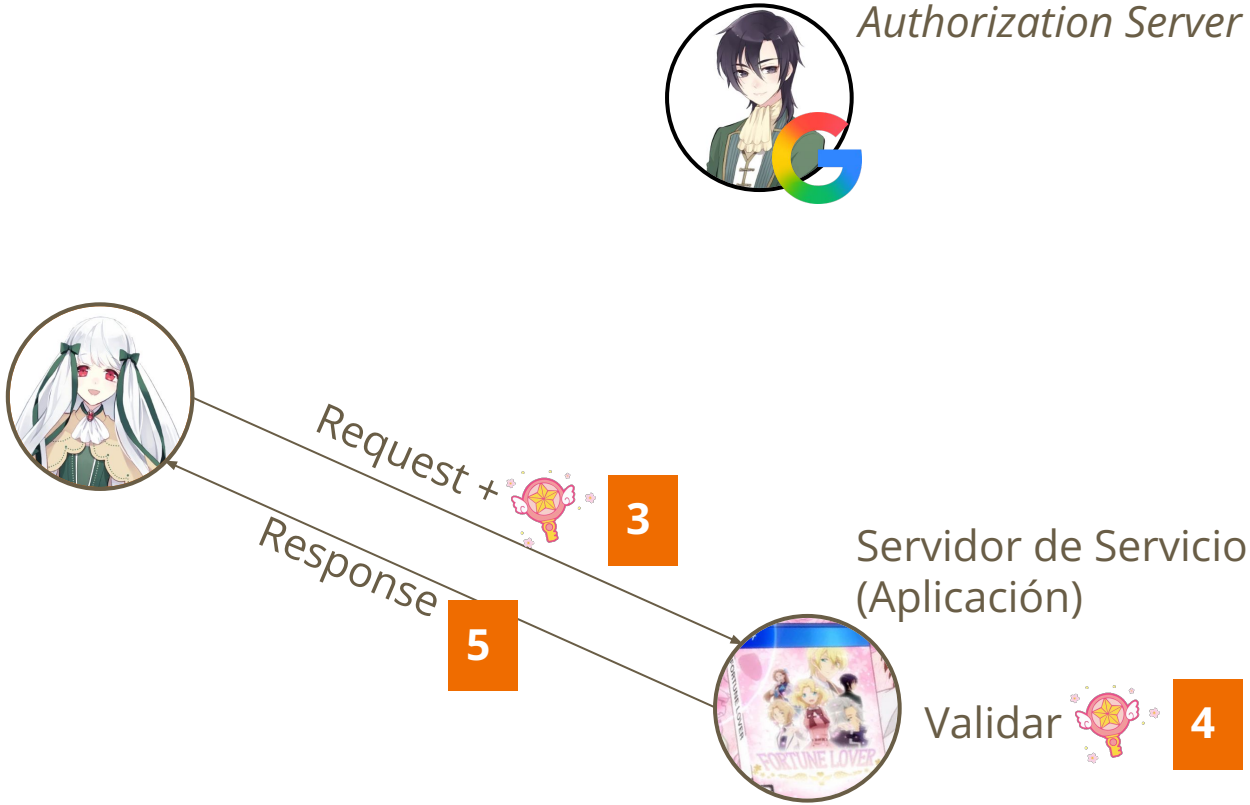
OAuth 2.0 (*Open Authorization*):

- Comenzó en noviembre de 2006 con la versión 1.0.
- Protocolo de delegación ampliamente utilizado que permite que las aplicaciones accedan a recursos protegidos en nombre de un usuario, sin necesidad de compartir las credenciales del usuario con la aplicación
  - Iniciar sesión con Google.
  - Iniciar sesión con Github.
- La versión OAuth 2.1 es la más actual y restringe el flujos de autorización, pero la más utilizada hasta ahora es la 2.0.

Introducción a OAuth 2.1 - *Pipeline*



Introducción a OAuth 2.1 - *Pipeline*



Canal seguro

Canales Seguros

- Una capa de servicio sobre un canal de comunicación que garantiza propiedades de seguridad para proteger los datos transmitidos entre dos procesos.
- Usualmente utiliza criptografía simétrica, asimétrica y funciones *hash*.

Canales Seguros

- Una capa de servicio sobre un canal de comunicación que garantiza propiedades de seguridad para proteger los datos transmitidos entre dos procesos.
- Usualmente utiliza criptografía simétrica, asimétrica y funciones *hash*.

Propiedades clave del canal

- **Autenticación**: Identidad fiable de ambas partes.
- **Privacidad**: Cifrado de extremo a extremo.
- **Integridad**: Protección frente a modificaciones.
- ***Anti-replay***: Prevención de retransmisión de mensajes.

Canales Seguros - Fases típicas

1. **Negociación (*handshake*)**

- Intercambio de claves (usualmente con criptografía asimétrica)
- Verificación de identidad (certificados, claves públicas)

2. **Establecimiento del canal**

- Derivación de clave simétrica compartida.
- Selección de algoritmos criptográficos.

3. **Comunicación segura**

- Cifrado y firma de cada mensaje.
- Inclusión de *timestamps* o números de secuencia para evitar *replay*.

Canales Seguros - Ejemplos

Protocolo / Tecnología	Usado en...
HTTPS (TLS)	Web, REST APIs
SSH	Administración remota de servidores
IPsec	VPNs a nivel de red
gRPC + TLS	Comunicación entre microservicios
SMTP + TLS	Correo seguro

Poniendo a prueba lo que hemos aprendido 👀

CineTV, una empresa de *streaming*, implementa una plataforma donde los usuarios deben iniciar sesión. Para esto, los usuarios pueden usar su cuenta de *Spotify* como medio de acceso para evitar ingresar directamente sus credenciales en el sitio.

Finalmente, los usuarios se han comenzado a quejar porque al momento de actualizar el país desde donde se están conectando, ya no pueden ver algunos videos que empezaron en su país de origen.

¿Cuál de las siguientes afirmaciones es **correcta** según lo descrito en este caso?

a. Spotify es un ejemplo de una Autoridad Certificadora para validar la identidad de CineTV.

b. La queja de los usuarios proviene de implementar una política basada en roles (RBAC).

c. CineTV permite OAuth y utiliza una política Basado en Atributos (ABAC) para la autorización.

d. Para implementar el acceso restringido a los videos, el mecanismo de capacidades es lo más adecuado y óptimo.

e. Spotify es un ejemplo de *Ticket Granting Server* porque da un *token* que permite al usuario ingresar al servicio de CineTV.

Poniendo a prueba lo que hemos aprendido 👀

CineTV, una empresa de *streaming*, implementa una plataforma donde los usuarios deben iniciar sesión. Para esto, los usuarios pueden usar su cuenta de *Spotify* como medio de acceso para evitar ingresar directamente sus credenciales en el sitio.

Finalmente, los usuarios se han comenzado a quejar porque al momento de actualizar el país desde donde se están conectando, ya no pueden ver algunos videos que empezaron en su país de origen.

¿Cuál de las siguientes afirmaciones es **correcta** según lo descrito en este caso?

a. Spotify es un ejemplo de una Autoridad Certificadora para validar la identidad de CineTV.

b. La queja de los usuarios proviene de implementar una política basada en roles (RBAC).

c. CineTV permite OAuth y utiliza una política Basado en Atributos (ABAC) para la autorización.

d. Para implementar el acceso restringido a los videos, el mecanismo de capacidades es lo más adecuado y óptimo.

e. Spotify es un ejemplo de *Ticket Granting Server* porque da un *token* que permite al usuario ingresar al servicio de CineTV.

Próximos eventos

Próxima clase

- Monitoreo de Seguridad.
- ¿Qué hacer cuando encriptar y garantizar identidad no es suficiente?

Evaluación

- Estamos con la investigación. Se entrega el lunes 24 de noviembre.

IIC2523
Sistemas Distribuidos

Hernán F. Valdivieso López
(2025 - 2 / Clase 21)

Créditos (animes utilizados)

Otome Gēmu no Hametsu Furagu Shika Nai Akuyaku Reijō ni Tensei Shiteshimatta...

Hataraku Maō-sama!