METODOS DE AUTENTICACION

API Keys + Digest Authentication

UNIVERSIDAD TECNOLOGICA DE HONDURAS

TRABAJO:

API Keys + Digest Authentication

ESTUDIANTES:

ALEX NAIN MEJIA

2006-305-200-01

RANDULFO ALONZO MEJIA

2020-100-105-94

DENNIS ELIAS MARADIAGA

2023-100-102-19

ASIGNATURA:

PROGRAMACION MÓVIL - I

CATEDRÁTICO:

RICARDO ENRIQUE LAGOS MENDOZA

FECHA:

5 JULIO 2025

API KEY Y DIGEST AUTHENTICATION.

Una API Key (clave de acceso) es una cadena única (generalmente alfanumérica y secreta) que se utiliza como token de autenticación para identificar al consumidor de un API.

COMO FUNCIONA:

- El cliente solicita la API Key al registrarse o autenticarse en un sistema (por ejemplo, un backend en PHP).
- El servidor genera una API Key única y la entrega al cliente.
- En cada solicitud futura, el cliente debe enviar la API Key.
- El servidor valida la clave:
  - Si es válida → Permite acceso.
  - Si es inválida o está vencida → Rechaza la solicitud.

SEGURIDAD:

Riesgo	Solución
Fácil de interceptar (en redes inseguras)	Siempre usar HTTPS
No expira automáticamente	Implementar expiración o revocación manual

No incluye mecanismos internos de integridad o anti-falsificación.	Se puede combinar con tokens JWT u HMAC

Digest Authentication – Funcionamiento Detallado

Digest Authentication es un método más seguro que Basic Auth, que no transmite la contraseña directamente, sino un hash MD5 de ciertos valores (incluyendo la contraseña, nonce, etc.).

- El cliente intenta acceder a un recurso protegido (por ejemplo, desde tu app Android al backend PHP).
- El servidor responde con un desafío (challenge) tipo 401 Unauthorized que incluye:
  - **WWW-Authenticate: Digest realm="api", nonce="abc123", opaque="xyz"**

El cliente genera un hash MD5 con:

- Usuario
- Contraseña
- Nonce recibido
- URI del recurso
- Método HTTP (GET, POST)

Y construye una cabecera **Authorization** como:

```
Authorization: Digest username="user",
realm="api",
nonce="abc123",
uri="/datos",
```

response="HASH_MD5",

opaque="xyz"

COMPARACION

CARACTERISTICAS	API KEY	Digest Authentication
Facilidad de implementación	Muy fácil (una clave)	Más complejo (hashes, nonce, etc.)
Seguridad básica	Baja si no se usa HTTPS	Mejor, no envía la contraseña directamente
Protección contra replay	No	Usa nonce
Expiración automática	No (debes implementarla)	Puede usarse con nonce caducable
Compatibilidad	Usado en APIs modernas (REST)	Menos usado hoy, no funciona bien con JSON y móviles
Uso en Android Studio	Fácil de implementar con headers	Difícil (necesitas librerías o lógica personalizada)

API Key Authentication

VENTAJAS	DESVENTAJAS
Simplicidad de implementación	**Protección limitada contra replay attacks**
○ Generar y validar una cadena alfanumérica es trivial tanto en el servidor (PHP) como en el cliente (Android/Java). ○ No requiere librerías criptográficas avanzadas: basta con comparaciones de strings y consultas a base de datos	○ La misma clave puede usarse indefinidamente hasta que expire o sea revocada.
Flexibilidad en permisos	**Transporte seguro obligatorio**
○ Puedes asociar cada clave a un usuario, proyecto o aplicación y gestionar scopes (lectura, escritura, admin). ○ Fácil de revocar, rotar o caducar individualmente.	○ Si se envía sin cifrar (HTTP), cualquiera en la red puede capturarla.
Compatibilidad amplia	**Falta de hashing dinámico**

○ Funciona con cualquier cliente HTTP capaz de poner headers o parámetros URL. ○ Muy usada en APIs REST, webhook callbacks, SDK públicas.	○ No hay nonce, timestamp o firma por petición, así que no detecta modificaciones de payload ni intercepciones.

Digest Authentication

VENTAJAS	DESVENTAJAS
Protección más robusta que Basic	**Complejidad de implementación**
○ Nunca envía la contraseña en texto plano: utiliza un hash MD5 que mezcla usuario, realm, nonce, URI y método HTTP.	○ Hay que gestionar nonces, contadores (nc), cnonces, calcular múltiples hashes y validar parámetros.
Resistencia a replay attacks	**No cifra datos**
○ Cada petición incorpora el nonce del servidor, un contador (nc) y un cnonce del cliente, impidiendo reutilizar viejos mensajes.	○ Las cabeceras siguen viajando en claro (aunque no revelan la contraseña), por lo que HTTPS sigue siendo obligatorio.
Integrado en el estándar HTTP	

○ No requiere rutas o endpoints adicionales: sigue el flujo 401 → challenge → respuesta dentro del mismo recurso.	
No requiere almacenamiento de estado en servidor	
○ El servidor solo debe verificar el hash, no lleva sesiones ni tokens persistentes.	

CASOS DE USO:

1. **APIs legacy:**

   Entornos donde ya se usa Digest y no conviene migrar rápidamente.

2. **Intranets o redes cerradas:**

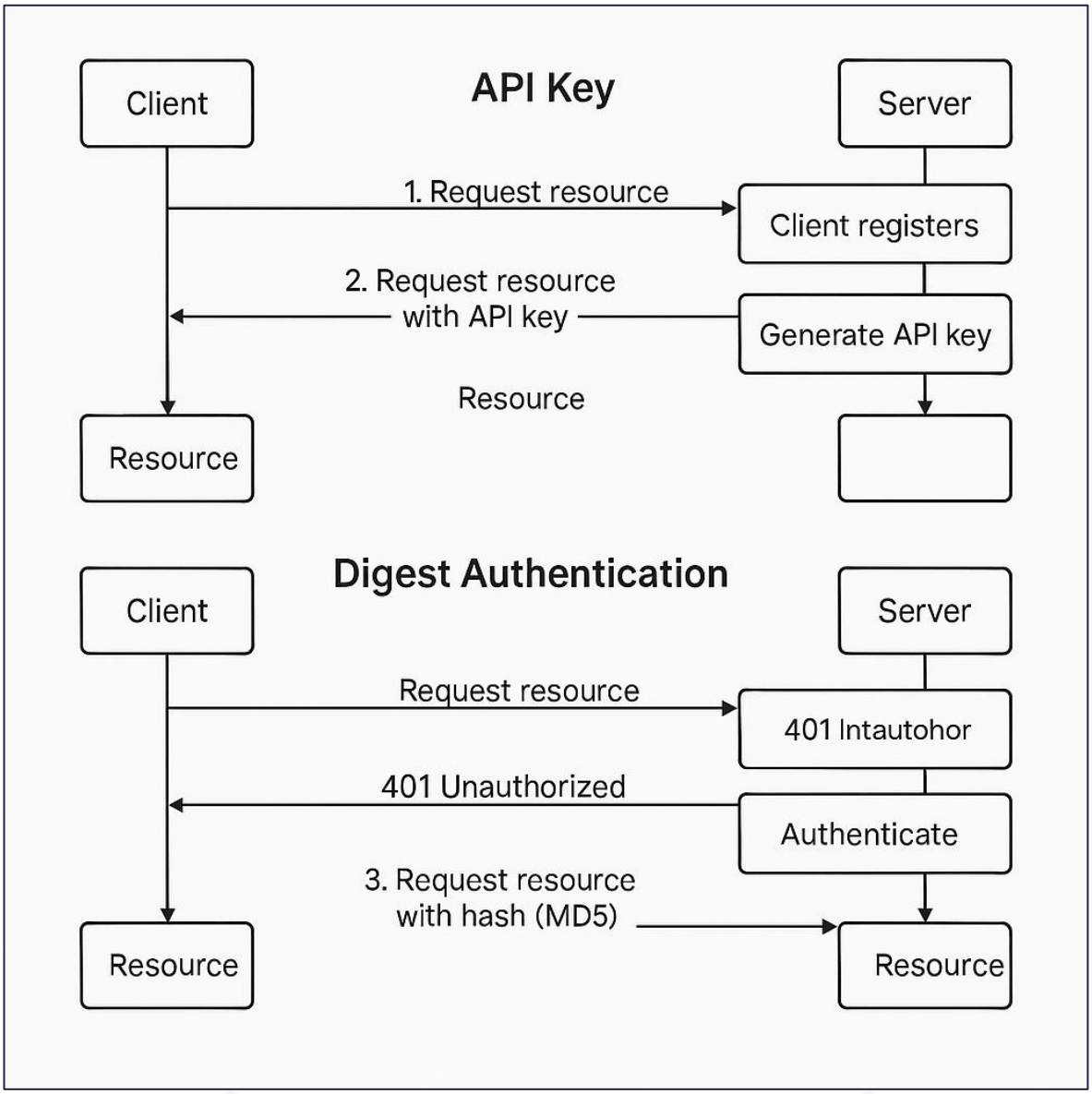
   Donde no hay ataques a gran escala, pero se quiere mejorar sobre Basic.

3. **Sistemas embebidos con HTTP nativo:**

   Equipos que solo soportan HTTP nativo y no pueden gestionar tokens complejos.

4. **Aplicaciones con requisitos de cumplimiento:**

   Algún estándar interno que exija autenticación HTTP estándar sin tokens externos.


API Key
Client
Server
1. Request resource
Client registers
2. Request resource
with API key
Generate API key
Resource
Resource
Digest Authentication
Client
Server
Request resource
401 Intautohor
401 Unauthorized
Authenticate
3. Request resource
with hash (MD5)
Resource
Resource

API Key Authentication

En la parte superior del diagrama se muestra el flujo de autenticación mediante **API Key**. Este método inicia cuando el **cliente se registra** en el servidor, el cual **genera una clave de acceso (API Key)** única y se la entrega al cliente. A partir de ese momento, cada vez que el cliente desea acceder a un recurso, envía una solicitud incluyendo dicha clave.

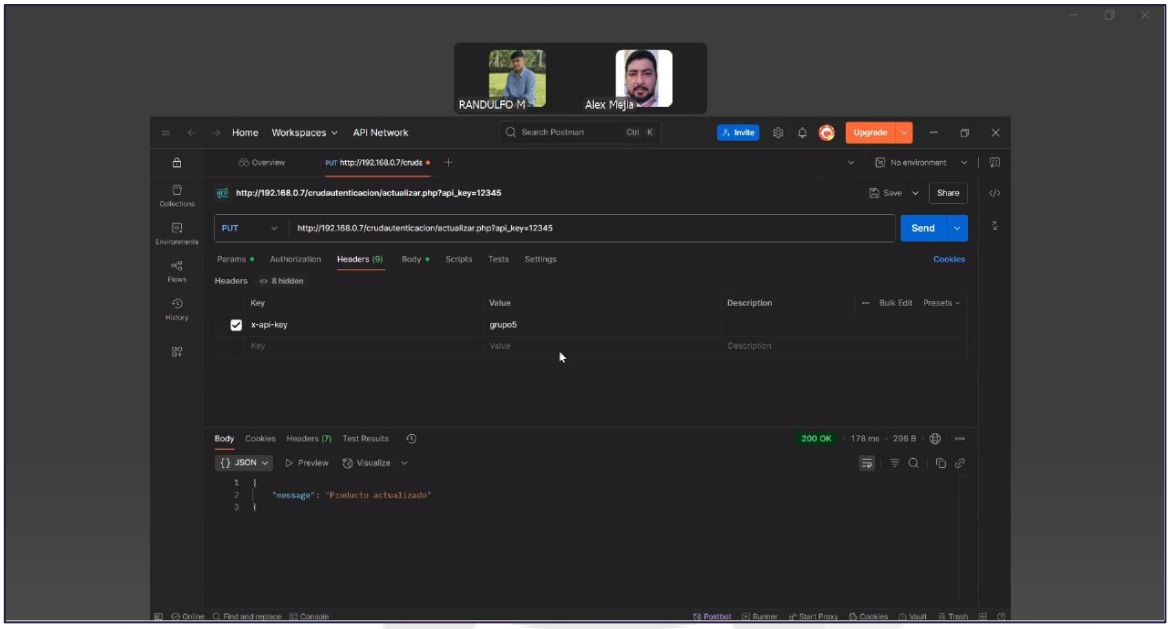
El servidor **verifica la validez de la API Key**. Si es correcta, permite el acceso al recurso. Este proceso es simple, directo y eficiente, ideal para aplicaciones móviles o integraciones entre sistemas.

Digest Authentication

En la parte inferior se representa el flujo más complejo de **Digest Authentication**. Este método comienza cuando el cliente solicita un recurso sin proporcionar aún credenciales. El servidor responde con un código 401 Unauthorized, que incluye un **desafío (challenge)** con parámetros como un nonce, realm y opaque.

El cliente responde realizando un cálculo **hash (MD5)** usando su nombre de usuario, contraseña, el nonce, el método HTTP, y otros parámetros, generando así una cabecera de autorización segura. El servidor valida este hash y, si coincide con el esperado, concede el acceso al recurso.

Este método ofrece una seguridad superior al evitar el envío directo de contraseñas, y añade medidas contra ataques de repetición, aunque implica mayor complejidad en su implementación.



CONCLUSION:

Tanto la autenticación mediante API Key como la Digest Authentication son mecanismos válidos para proteger el acceso a APIs.

La API Key destaca por su facilidad de uso e implementación, mientras que Digest ofrece mayor seguridad al evitar el envío directo de contraseñas.

La elección entre ambos dependerá del nivel de seguridad requerido y la complejidad que se esté dispuesto a manejar.