¿Qué es una clave de firma?

En el contexto de OpenID Connect (OIDC) , una **clave de firma** (signing key), generalmente un par de claves asimétricas, se utiliza para firmar y verificar JSON Web Tokens (JWTs) . Los proveedores OpenID utilizan claves de firma para firmar tokens como ID tokens y access tokens para garantizar su integridad y autenticidad.

Aunque el concepto de claves de firma puede ser más amplio, nos centraremos en cómo se utilizan en OIDC para asegurar tokens. Otros casos de uso, como firmar correos electrónicos, documentos y paquetes de software, pueden derivarse de los mismos principios.

Ejemplo: Firmado de ID token

Cuando un usuario se autentica (authentication) con un proveedor OpenID, el proveedor emite un ID token que contiene información del usuario (claims) y es firmado por la clave de firma del proveedor. Dado que el ID token es un JWT, consta de tres partes: encabezado, carga útil y firma.

1. Encabezado

Supongamos que el encabezado es:

```
{
  "alg": "ES384",
  "typ": "JWT"
}
```

El JSON indica que el ID token está firmado utilizando el algoritmo ECDSA con la curva P-384. El campo `typ` especifica que el tipo de token es JWT.

2. Carga útil

La carga útil contiene información básica del usuario:

```
{
  "sub": "1234567890",
  "name": "Alice"
```

```
}
```

El `sub` claim es el identificador único del usuario, y `name` es su nombre para mostrar.

3. Firmar el token

De acuerdo con el formato JWT, el encabezado y la carga útil deben estar codificados en Base64URL y concatenados con un `.` para firmar:

```
{{header}}.{{payload}}
```

En este caso, el valor sería:

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkFsaWNlIn0
```

Supongamos que el proveedor OpenID utiliza la siguiente clave privada para firmar el token:

```
-----BEGIN PRIVATE KEY-----
MIG2AgEAMBAGByqGSM49AgEGBSuBBAAiBIGeMIGbAgEBBDBW9PDXInlNT2hjOtQr
g4pkVkyJsKia33dHrsbOG4Z77pfYN7SYZCHh9YdLXTTKinehZANiAAQX/FB1s6Gj
YnDSGCY08PRUAQ8CCRCt8Ph/VDHfLj1xSbrjp8wFf0NjH7jcfNebpV1fvu4XKbP3
Ro7h0G6elN1TMsVECJPv4ieDNkYOsgT4UboJypC5E/rmvrlJTMM6Y/k=
-----END PRIVATE KEY-----
```

Para firmar el token, el proveedor OpenID necesita usar la clave privada para generar una firma mediante:

```
signature = sign(header + '.' + payload, private_key)
```

Luego, la firma codificada en Base64URL es:

```
Cjy6A_FHnwQBP0hRawoGTkRy8m8o0Ncc1q4BeyxYr0fxhKYmJJinIWZPXJdaAXRO9wOFuH2-
UML2yWHjot_LnCPO6362asMvgNkEJMZ6UtqyOPlsCOJ7voTPOCT6sYu2
```

4. Ensamblar el JWT

Finalmente, el proveedor OpenID ensambla el JWT concatenando el encabezado, la carga útil y la firma con `.`:

```
{{header}}.{{payload}}.{{signature}}
```

En este caso, el ID token sería:

```
eyJhbGciOiJFUzM4NCIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkFsaWNlIn0.Cjy6A_FHn
wQBP0hRawoGTkRy8m8o0Ncc1q4BeyxYr0fxhKYmJJinIWZPXJdaAXRO9wOFuH2-
UML2yWHjot_LnCPO6362asMvgNkEJMZ6UtqyOPlsCOJ7voTPOCT6sYu2
```

El ID token está ahora listo para ser enviado al Cliente (Client) para su procesamiento posterior.

5. Verificar el token

Cuando el cliente recibe el ID token, puede verificar la firma utilizando la clave pública del proveedor OpenID. Por lo general, la clave pública está disponible a través del endpoint de Descubrimiento de OpenID Connect (OpenID Connect Discovery) (`jwks_uri`) en el formato de un Conjunto de Claves Web JSON (JWKS) .

Para este ejemplo, la clave pública es:

```
-----BEGIN PUBLIC KEY-----
MHYwEAYHKoZIzj0CAQYFK4EEACIDYgAEF/xQdbOho2Jw0hgmNPD0VAEPAgkQrfD4
f1Qx3y49cUm646fMBX9DYx-43HzXm6VdX77uFymz90aO4dBunpTdUzLFRAiT7+In
gzZGDrIE+FG6CcqQuRP65r65SUzDOmP5
-----END PUBLIC KEY-----
```

Y el valor JWK correspondiente es:

```
{
  "kty": "EC",
  "crv": "P-384",
  "x": "F_xQdbOho2Jw0hgmNPD0VAEPAgkQrfD4f1Qx3y49cUm646fMBX9DYx-43HzXm6Vd",
  "y": "X77uFymz90aO4dBunpTdUzLFRAiT7-IngzZGDrIE-FG6CcqQuRP65r65SUzDOmP5"
}
```

Ahora, el cliente puede verificar la firma utilizando la clave pública.

Elegir el algoritmo correcto

Existen varios algoritmos disponibles para firmar JWTs:

- **Algoritmos simétricos**: HMAC con la familia SHA (por ejemplo, HS256, HS384, HS512) es un algoritmo simétrico que utiliza la misma clave tanto para firmar como para verificar.

No se recomienda en la mayoría de los casos ya que la clave secreta necesita ser compartida entre las partes.

- **Algoritmos asimétricos**: RSA (por ejemplo, RS256, RS384, RS512) y ECDSA (por ejemplo, ES256, ES384, ES512) son algoritmos asimétricos que utilizan un par de claves: una clave privada para firmar y una clave pública para verificar.
  - RSA es ampliamente utilizado y compatible con muchas bibliotecas y plataformas. Sin embargo, tiene un tamaño de clave y de firma mucho mayor en comparación con ECDSA.
  - ECDSA es más eficiente y genera firmas más pequeñas, lo que lo convierte en una mejor opción para entornos restringidos. Dado que es menos común, asegúrate de que tu plataforma lo soporte.

> ECDSA debería ser la opción preferida para nuevas aplicaciones debido a sus beneficios en rendimiento y seguridad.

Otros escenarios de clave de firma

Aunque el ejemplo anterior se centra en ID tokens en OIDC, el concepto de clave de firma se utiliza ampliamente en varios escenarios, como firmar correos electrónicos, documentos y paquetes de software. Los principios clave permanecen iguales:

- Para claves simétricas, se utiliza la misma clave tanto para firmar como para verificar. Esto es adecuado para escenarios donde las partes pueden compartir la clave de manera segura, o hay una sola entidad responsable de firmar y verificar.
- Para claves asimétricas, se utiliza una clave privada para firmar y una clave pública correspondiente para verificar. Esto es adecuado para la mayoría de los escenarios donde las partes que firman y verifican son entidades diferentes.

Revisión #2
Creado 2024-11-13 13:48:38 UTC por Marluan Espiritusanto
Actualizado 2024-11-13 14:14:46 UTC por Marluan Espiritusanto