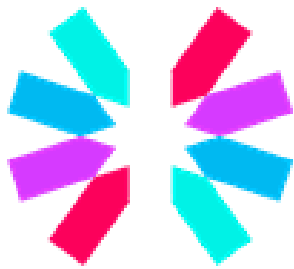
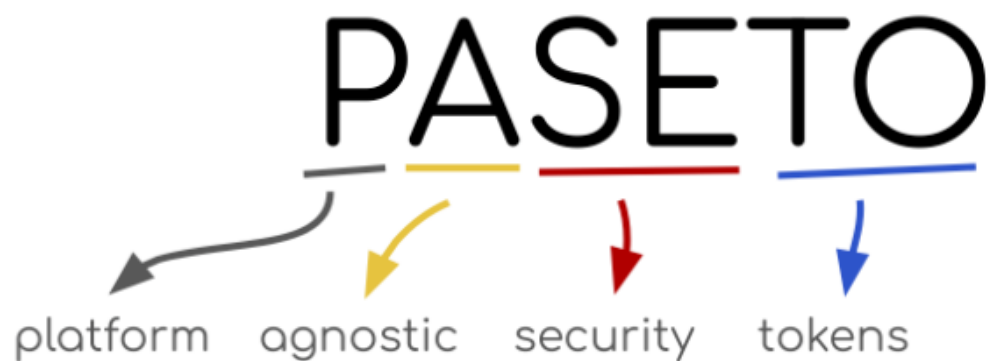


Comparativa entre JWT y PASETO



JWT

Integrantes

Cravero, Pablo

Pérez Fontela, Simón

Roba Martinez, Manuel

Soletti, Sol

Año: 2023

Comisión: 303

Profesores: Meca, Adrián

Tabacman, Ricardo

Índice	
INTRODUCCIÓN	2
DEFINICIONES.....	3
JWT.....	5
PASETO.....	8
CONCLUSIÓN.....	
BIBLIOGRAFÍA.....	

Introducción

En el mundo de la tecnología y la autenticación, tenemos diversos instrumentos a la hora de programar. Siendo JWT y PASETO dos de estos instrumentos más utilizados, compararemos las medidas de seguridad y facilidad de uso entre ambas tecnologías, para identificar y decidir cuál será la que mejor nos acompañe en el proyecto.

Definiciones

Objeto JSON:

```
{  
  "nombre": "Juan",  
  "edad": 30,  
  "ciudad": "Ejemploville"  
}
```

Un objeto JSON (JavaScript Object Notation) es una estructura de datos utilizada para representar información en un formato legible por humanos y fácil de interpretar por las máquinas. Los objetos JSON se utilizan comúnmente para intercambiar datos entre un servidor y un cliente, o entre diferentes componentes de software.

RFC 7519: RFC (Request for Comments) 7519 es un documento que especifica el estándar para JSON Web Tokens (JWT). Son una serie de documentos que describen varios aspectos de internet y tecnologías relacionadas, y en este caso, el RFC 7519 establece las reglas y estándares para la estructura y el uso de los JWT.

HMAC (Hash-based Message Authentication Code): HMAC es un algoritmo de autenticación basado en hash. Se utiliza para verificar la integridad y autenticidad de los datos transmitidos. En el contexto de JWT, HMAC se usa para firmar tokens. Cuando se firma un JWT con HMAC, se utiliza una clave secreta compartida entre las partes para generar una firma que puede ser verificada posteriormente para garantizar que el token no ha sido alterado.

RSA (Rivest-Shamir-Adleman): RSA es un algoritmo de criptografía asimétrica que se utiliza para firmar y cifrar datos. En el contexto de JWT, RSA se usa para firmar tokens cuando se emplea un par de claves pública/privada. La parte que firma el token utiliza su clave privada para generar la firma, y otros pueden verificar esa firma utilizando la clave pública correspondiente. Esto garantiza la autenticidad del token.

ECDSA (Elliptic Curve Digital Signature Algorithm): ECDSA es otro algoritmo de criptografía asimétrica que se utiliza para firmar datos, similar a RSA. La principal diferencia es que ECDSA se basa en curvas elípticas en lugar de factorización entera, lo que lo hace más eficiente en términos de rendimiento y uso de recursos. Al igual que RSA, ECDSA se utiliza en JWT para firmar tokens cuando se emplea un par de claves pública/privada.

Desarrollo

La comparativa entre ambas se realizará usando las siguientes bibliotecas y entornos de desarrollo: Visual Studio Code, Angular, MongoDB.

Comenzaremos describiendo y analizando brevemente ambas tecnologías.

Json Web Token

(JWT para los amigos)

JSON Web Token (JWT) es un estándar abierto (RFC 7519) que define una forma compacta y autosuficiente para transmitir información de manera segura entre partes como un objeto JSON. Esta información puede ser verificada y confiable porque está firmada digitalmente. Los JWT pueden ser firmados utilizando un secreto (con el algoritmo HMAC) o un par de claves pública/privada utilizando RSA o ECDSA.

Autorización: Este es el escenario más común para el uso de JWT. Una vez que el usuario ha iniciado sesión, cada solicitud subsiguiente incluirá el JWT, permitiendo al usuario acceder a rutas, servicios y recursos que están permitidos con ese token. El Inicio de Sesión Único (Single Sign On) es una característica que utiliza ampliamente JWT en la actualidad, debido a su pequeña sobrecarga y su capacidad para utilizarse fácilmente en diferentes dominios.

ESTRUCTURA

JWT consiste de tres partes separadas por puntos (.) las cuales son:

Header

El encabezado típicamente consta de dos partes: el tipo de token, que es JWT, y el algoritmo de firma que se está utilizando, como HMAC SHA256 o RSA.

```
{  
  "alg": "HS256",  
  "typ": "JWT"  
}
```

Payload

La segunda parte del token es la carga útil (payload), que contiene las afirmaciones (claims). Las afirmaciones son declaraciones sobre una entidad (típicamente, el usuario) y datos adicionales. Hay tres tipos de afirmaciones: afirmaciones registradas, públicas y privadas.

Afirmaciones registradas: Estas son un conjunto de afirmaciones predefinidas que no son obligatorias, pero se recomienda usar para proporcionar un conjunto de afirmaciones útiles e interoperables. Algunas de ellas son: iss (emisor), exp (tiempo de vencimiento), sub (sujeto), aud (audiencia) y otras.

Es importante notar que los nombres de las afirmaciones tienen solamente tres caracteres, ya que JWT está diseñado para ser compacto.

Afirmaciones públicas: Estas pueden ser definidas a voluntad por quienes utilizan JWT. Sin embargo, para evitar colisiones, se deben definir en el Registro JSON Web Token de la IANA o como un URI que contenga un espacio de nombres resistente a colisiones.

Afirmaciones privadas: Estas son afirmaciones personalizadas creadas para compartir información entre partes que acuerdan utilizarlas y que no son ni afirmaciones registradas ni públicas.

```
{  
  "sub": "1234567890",  
  "name": "John Doe",  
  "admin": true  
}
```

Signature

Para crear la parte de la firma (signature) de un JWT, debes tomar el encabezado codificado, la carga útil (payload) codificada, un secreto (o clave), el algoritmo especificado en el encabezado y firmar esos datos.

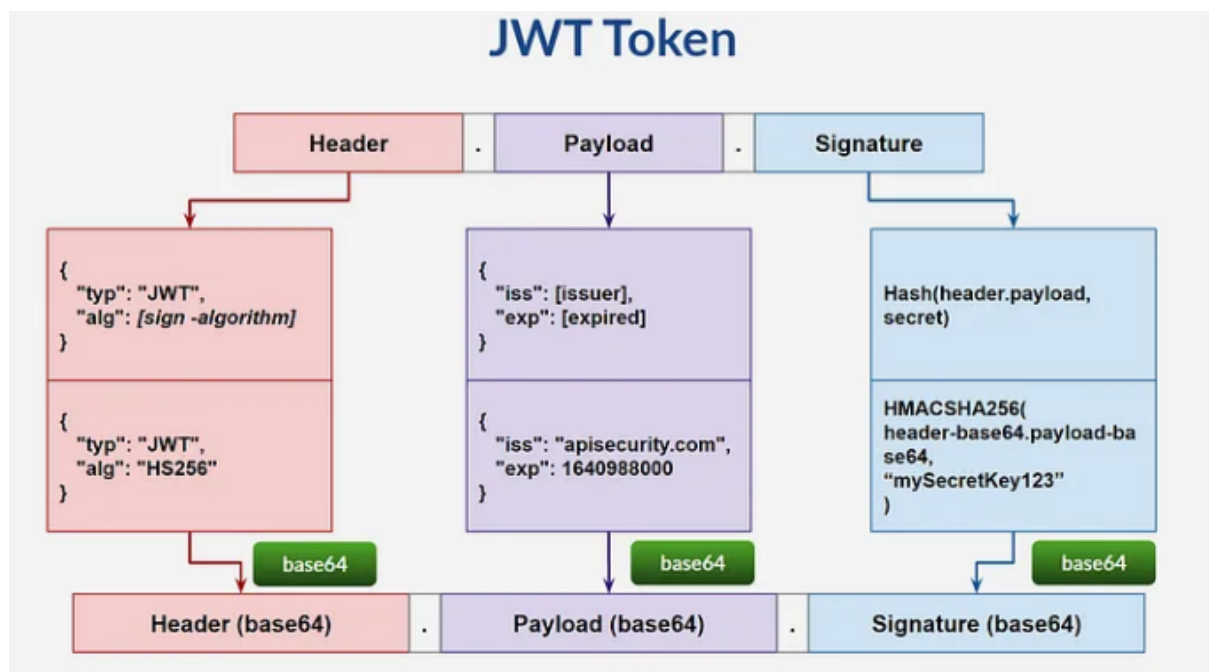
```

HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  secret)

```

La firma se utiliza para verificar que el mensaje no haya sido alterado en el camino y, en el caso de los tokens firmados con una clave privada, también puede verificar que el emisor del JWT es quien dice ser.

A menos que alguien tenga la clave secreta, no pueden descifrar esta información, por lo que aquí es donde JWT se vuelve seguro.



Poniendo todo junto, obtenemos el JWT, que se ve algo así:

```

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4
gRG9lIiwiaXNTb2NpYWwiOiJ1bnRydWV9.
4pcPyMD09olPSyXnrXCjTwXyr4BsezdI1AVTmud2fU4

```


Puede ser decodificado con el debugger de jwt.io

Paseto

Definición

Paseto es una especificación que permite crear tokens seguros y sin estado que pueden compartirse de forma segura mediante la web. Permite tomar datos JSON y condensarlos en un token que puede compartirse y es difícil de manipular.

En términos simples Paseto es un blob JSON el cual queremos transmitir seguramente a través del internet por más que la ruta sea insegura.

```
{  
  "nombre": "Juan",  
  "edad": 30,  
  "ciudad": "Ejemploville"  
}
```

Para qué se usan

Paseto se usa para probar que la información que estamos mandando mediante un JWT es confiable.

Estructura

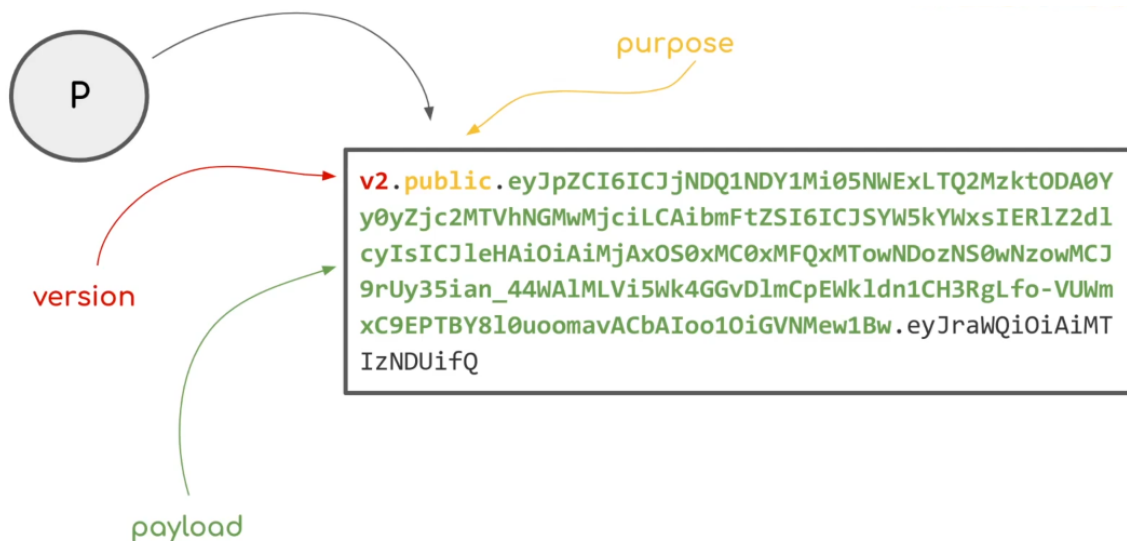
Los Paseto son, entonces, cadenas codificadas en formato binario.

El Paseto se divide en tres o cuatro secciones delimitadas por períodos.

La primera sección del Paseto es la versión del protocolo, que indica qué versión del estándar Paseto se está usando.

La segunda sección es la finalidad, que puede ser local o pública.

La tercera sección define el contenido real del token y se conoce como carga útil. Esta especifica cuándo se crea el token, y, si se tiene la clave correcta, puede descifrarla y ver los datos JSON originales.



La última sección es el pie de página, que puede utilizarse para almacenar metadatos adicionales sin cifrar. Esta sección es opcional.

Tipos de PASETO

Local

Se crean y cifran utilizando una clave secreta. Una biblioteca para desarrolladores de PASETO tomará los datos JSON que desee transmitir y los cifrará usando la clave. Este tipo de PASETO se puede descifrar luego utilizando la misma clave secreta que se utilizó para crearlo.

Para crearlos, una función genera una cadena de bytes aleatoria, que es utilizada para crear un nonce (un número arbitrario que puede ser usado sólo una vez). Luego, el encabezado se combina con el nonce y el pie de página para crear una cadena de autenticación previa. Después, la carga útil del token se cifra, y finalmente se crea una cadena de token.

Una vez que se crea la cadena, se puede compartir el token de forma segura a través de cualquier medio. Como el token está cifrado, los atacantes no pueden ver los datos de su carga útil sin su clave secreta ni modificarlos.

Estos tipos de PASETO se deben utilizar en escenarios en los que se pueda almacenar de forma segura una clave secreta compartida.

Público

Se utilizan para entornos en donde no se puede compartir de forma segura una clave secreta con todas las partes involucradas en una transacción. No están cifrados, pero están firmados digitalmente, razón por la cual si un atacante consigue el PASETO, podrá ver todos los datos que contiene, pero no modificarlos sin una alerta.

Para crear un PASETO público, se utilizan herramientas para crear claves públicas y privadas. Luego, el encabezado se combina con el contenido de la carga útil y el pie de página en una cadena de autenticación previa. Esta cadena está firmada digitalmente por la clave privada. Todo se empaqueta en el formato de cadena PASETO.

Una vez que se tenga la cadena del token, podrá compartirse de forma segura con terceros sin preocuparse de si los atacantes pueden modificarla.

Con el token creado, puede compartirse también su clave pública. Cualquiera que la tenga podrá verificar la validez del token.

Su uso ideal es transmitir datos JSON de forma que otras personas puedan saber quién los creó.

Reclamos

Los reclamos son claves JSON. El estándar PASETO define reclamos reservados que no puede utilizar para otro propósito que no sea el oficial. Los reclamos tienen propósitos especiales y las implementaciones de PASETO les permiten utilizarlos para lograr hazañas útiles.

Key	Name	Type	Example
iss	Issuer	string	{"iss":"paragonie.com"}
sub	Subject	string	{"sub":"test"}
aud	Audience	string	{"aud":"pie-hosted.com"}
exp	Expiration	DtTime	{"exp":"2039-01-01T00:00:00+00:00"}
nbf	Not Before	DtTime	{"nbf":"2038-04-01T00:00:00+00:00"}
iat	Issued At	DtTime	{"iat":"2038-03-17T00:00:00+00:00"}
jti	Token ID	string	{"jti":"87IFSGFgPNtQNNuw0AtuLttP"}
kid	Key-ID	string	{"kid":"stored-in-the-footer"}

Desventaja

La principal desventaja de los PASETO es que no están diseñados para ser tokens reutilizables, pues no tienen mecanismos para prevenir ataques de repetición. Si un atacante puede obtener un PASETO válido y usarlo varias veces, hay un error en el uso de los PASETO.

Comparativas generales

DOCUMENTACIÓN

La documentación de PASETO es mucho más simple y se enfoca especialmente en la seguridad, reduciendo la complejidad para la validación de tokens. Además, se basa en un formato binario de los tokens y explica cómo personalizarlos.

La documentación de JWT, en cambio, es más amplia y se basa en estándares. Admite una gran variedad de algoritmos de firma y es más fácil de implementar. Los tokens son menos compactos que en PASETO.

COMUNIDAD

Dentro de la comunidad, hay distintas discusiones acerca de cuál tecnología es mejor y por qué. Sin embargo, no hay una marcada diferencia entre la cantidad de usuarios que prefieran una por sobre la otra.



Publicado por [u/Left-Independent9874](#) hace 1 año 🇺🇸

1


Do you guys use JWT or Paseto?



I was just wondering what you guys use and WHY




P.S.. I'm fairly new to programming.

💬 Comentarios: 4 ➦ Compartir 📌 Guardar ...

- 




carsonbottomtooth · hace 1 a

JWT. Because I've never had a good reason to implement OAuth myself and JWT was the only choice. Reading more on Paseto and JWT vulnerabilities, this seems unfortunate. Always scan your dependencies kids!

 3
 
 Compartir ...
- 



Atomfinger · hace 1 a

JWT here.

 2
 
 Compartir ...
- 

[deleted] · hace 6 a

Can confirm JWT is a ~~PoC~~ PoC: very poorly designed. Thanks for your efforts in bringing security to the masses.

 3
 
 Responder
 Compartir ...

PASETO

Paseto is everything you love about JOSE (JWT, JWE, JWS) without any of the [many design deficits that plague the JOSE standards](#).

PERFORMANCE

JWT se destaca por ofrecer flexibilidad a la hora de elegir un algoritmo de firma digital y la implementación de verificación, pero esto conlleva, a su vez, algunos problemas, como errores en la implementación y una consecuente vulnerabilidad en la seguridad. PASETO, por otra parte, soluciona estos problemas mediante la simplificación del proceso de implementación, tomando un enfoque más duro, y centrándose en la seguridad para reducir las potenciales vulnerabilidades. En este sentido, la performance de JWT se reduce, y la de PASETO aumenta.

Por otra parte, los tokens de PASETO se codifican y decodifican más rápido debido a su formato binario, que los hace más compactos. Esto significa que PASETO requiere menos poder de procesamiento y que la validación de tokens es más rápida que en JWT.

Otra desventaja que tiene JWT frente a PASETO se basa en los algoritmos de firma que, en el primer caso, requieren la gestión de claves públicas y privadas que agrega complejidad y disminuye el rendimiento. En PASETO, se

utiliza una clave secreta compartida, simplificando el manejo de claves y mejorando el rendimiento.

De este modo, podemos concluir que, en términos generales, la performance de PASETO supera ampliamente a la de JWT debido a su enfoque más simple y en la seguridad, mientras JWT mejora en la flexibilidad y compatibilidad.


PLUGINS Y BIBLIOTECAS

Dentro de la documentación oficial de jwt, podemos encontrar una lista extensa de librerías y plugins para el Token Signing/Verification. También podemos encontrar esta información para paseto en su documentación oficial.

paseto
v1/v2 support


Name	Language	Author	Features			
			v1.l	v1.p	v2.l	v2.p
authenticvision/libpaseto	C	Thomas Renoth	✗	✗	✓	✓
lanleec Clark/Paseto	Elixir	Ian Clark	✓	✓	✓	✓
o1egl/paseto	Go	Oleg Lobanov	✓	✓	✓	✓
go-paseto	Go	Aidan T. Woods	✗	✗	✓	✓

jwt




.NET

- Sign
- Verify
- iss check
- sub check
- sud check
- exp check
- nbf check
- iat check
- jti check
- typ check
- ES256
- ES256K
- ES384
- ES512
- EdDSA



.NET

- Sign
- Verify
- iss check
- sub check
- sud check
- exp check
- nbf check
- iat check
- jti check
- typ check
- HS256
- HS384
- HS512
- PS256
- PS384
- PS512
- RS256
- RS384
- RS512
- ES256
- ES256K
- ES384
- ES512
- EdDSA



.NET

- Sign
- Verify
- iss check
- sub check
- sud check
- exp check
- nbf check
- iat check
- jti check
- typ check
- HS256
- HS384
- HS512
- PS256
- PS384
- PS512
- RS256
- RS384
- RS512
- ES256
- ES256K
- ES384
- ES512
- EdDSA

Dentro de ambas documentaciones, podemos observar como JWT realiza una extensa explicación actualizable acerca de las bibliotecas , por versión que apoya. Lo mismo sucede con Paseto, siendo este quién menor cantidad de librerías posee, debido a la diferencia de longevidad entre ambas tecnologías y a su compatibilidad limitada debido a las restricciones de seguridad que impone.

Comparativa	JWT	Paseto
Documentación	Cantidad de información: amplia y se encontraron muchos ejemplos distintos de implementaciones de la misma	Cantidad de información: gran cantidad de información acerca de cada una de sus versiones
Comunidad	No hay una tendencia a elegir una por sobre la otra.	
Curva de aprendizaje	Poco empinada gracias a su flexibilidad de implementación	Más empinada al ser una tecnología más nueva
Performance	Factores como el tamaño de tokens, el procesamiento y validación, y falla en la seguridad de los mismos disminuye la performance.	Ofrece una mejor performance debido a los tokens compactos y el enfoque en la seguridad.
Plugins	Mayor cantidad de plugins debido a su longevidad y gran uso	Menor cantidad de plugins debido a la rigidez de seguridad, pero sus plugins tienen muy buen rendimiento y son altamente implementados
Bibliotecas	Amplia gama de bibliotecas	Amplia gama de bibliotecas open-source.
Herramientas	Gran cantidad de herramientas y extensiones.	Gran cantidad de herramientas y extensiones.

Integración con otras herramientas	Gran compatibilidad y flexibilidad.	Compatibilidad limitada debido al enfoque rígido de la misma.
---	-------------------------------------	---

Comparativa de código ejemplo: seguridad y facilidad de uso

El código estará proporcionado en el mismo repositorio donde se encuentra este archivo, sin embargo haremos referencia a ellos en este informe.

JWT

```
router.post('/login', async(req, res) => {

  const { email, password } = req.body;
  const user = await User.findOne({email})
  if (!user) return res.status(401).send("Email no existe");
  const passwordMatch = await bcrypt.compare(password, user.password);
  if (!passwordMatch) return res.status(401).send("Contraseña Incorrecta");
  const token = jwt.sign({ _id: user._id, role: user.role}, 'secretKey');
  res.status(200).json({ token });
  console.log(user.role);

});
```

PASETO

```
router.post('/login', async (req, res) => {
  const { email, password } = req.body;
  const user = await User.findOne({ email });
  if (!user) return res.status(401).send("Email no existe");
  const isPasswordValid = await bcrypt.compare(password, user.password);
  if (!isPasswordValid) return res.status(401).send("Contraseña Incorrecta");
  const tokenPaseto = await v2.sign({ _id: user._id}, 'secretKey');
  res.status(200).json({ tokenPaseto });
});
```

Como conclusión y comparativa, la manera en llamar a la función es la misma, tanto en dificultad como sintaxis. El mayor inconveniente que encontramos a la hora de realizar esta comparativa, es la cantidad de fuentes de información con códigos de paseto, para poder guiarnos de errores frecuentes, y encontrar un código óptimo para la realización de este informe.

Conclusión

La elección entre PASETO y JWT como sistema de autenticación y autorización depende de una serie de factores clave. JWT es ampliamente conocido y adoptado, con una gran comunidad de usuarios, lo que facilita el acceso a recursos y soporte. PASETO, aunque más nuevo, está ganando terreno con documentación en crecimiento y un diseño simple y eficiente. La curva de aprendizaje y el rendimiento pueden variar, y la elección dependerá de las necesidades específicas de tu proyecto. Ambos tienen sus ventajas y desventajas, por lo que es esencial considerar las prioridades y restricciones de tu aplicación antes de decidir cuál usar.

Dentro de las pruebas realizadas, encontramos más comodidad a la hora de utilizar JWT, pero aun así quedamos intrigados por el potencial creciente de PASETO.

Bibliografía

- <https://github.com/paseto-standard/paseto-spec>
- <https://jwt.io>
- <https://developer.okta.com/blog/2019/10/17/a-thorough-introduction-to-paseto>
- <https://www.youtube.com/watch?v=nBGx-q52KAY>
- <https://www.youtube.com/watch?v=Oi4FHDGILuY>
- <https://news.ycombinator.com/item?id=33063305>
- <https://www.linkedin.com/pulse/paseto-jwt-killer-sande-sh-dahake>
- <https://engineering.monstar-lab.com/en/post/2023/05/16/An-overview-of-PASETO-Token-Based-Authentication/>
- <https://github.com/paragonie/paseto/tree/master/docs>
- <https://dev.to/techschoolguru/why-paseto-is-better-than-jwt-for-token-based-authentication-1b0c>
- <https://medium.com/batc/jwt-for-dummies-ok-not-100-dummies-1f08d3279a0b>
- <https://dev.to/carstenbehrens/the-basics-of-jwt-json-web-tokens-for-dummies-2iin>
- <https://www.youtube.com/watch?v=67mezK3NzpU>
- https://en.wikipedia.org/wiki/Cryptographic_nonce
- <https://www.reddit.com/search/?q=jwt%20vs%20paseto>
- <https://github.com/paseto-toolkit/jpaseto>
- <https://techwasti.com/paseto-vs-jose-jws-jwe-and-jwt>

