

PROYECTO — JULIO 2023

Criptografía

20

23

Case 1

XOR

Tenemos un sistema que usa claves de 16 bytes. Por razones de seguridad vamos a proteger la clave de tal forma que ninguna persona tenga acceso directamente a la clave. Por ello, vamos a realizar un proceso de disociación de la misma, en el cuál tendremos, una clave fija en código, la cual, sólo el desarrollador tendrá acceso, y otra parte en un fichero de propiedades que rellenará el Key Manager. La clave final se generará por código, realizando un XOR entre la que se encuentra en el properties y en el código.

La clave fija en código es B1EF2ACFE2BAEEFF, mientras que en desarrollo sabemos que la clave final (en memoria) es 91BA13BA21AABB12.

¿Qué valor ha puesto el Key Manager en properties para forzar dicha clave final?

20553975c31055ed

La clave fija, recordemos es B1EF2ACFE2BAEEFF, mientras que en producción sabemos que la parte dinámica que se modifica en los ficheros de propiedades es B98A15BA31AE3B3F.

¿Qué clave será con la que se trabaje en memoria?

08653f75d31455c0

Case 2

AES-CBC

Dada la clave con etiqueta “cifrado-sim-aes-256” que contiene el keystore. El iv estará compuesto por el hexadecimal correspondiente a ceros binarios (“00”). Se requiere obtener el dato en claro correspondiente al siguiente dato cifrado:

**TQ9SOMKc6aFS9SlxhfK9wT18UXpPCd505Xf5J/5nLI7Of/o0QKIWXg3nu1RRz4QWElezdrLA
D5LO4USt3aB/i50nvvJbBiG+le1ZhpR84ol=**

1) Se ha usado un AES/CBC/PKCS7. Si lo desciframos, ¿Qué obtenemos?

**Esto es un cifrado en bloque típico. Recuerda, vas por el buen camino.
Ánimo.**

2) ¿Qué ocurre si decidimos cambiar el padding a x923 en el descifrado?

El texto en claro se mantiene, seguimos obteniendo el mismo resultado

3) ¿Cuánto padding se ha añadido en el cifrado?

1 byte de padding

Case 3

ChaCha20

Se requiere cifrar el texto “KeepCoding te enseña a codificar y a cifrar”. La clave para ello, tiene la etiqueta en el Keystore “cifrado-sim-chacha-256”. El **nonce** “9Yccn/f5nJJhAt2S”. El algoritmo que se debe usar es un ChaCha20.

1. ¿Cómo podríamos mejorar de forma sencilla el sistema, de tal forma, que no sólo **garanticemos la confidencialidad sino, además, la integridad del mismo?** Se requiere obtener el dato cifrado, demuestra tu propuesta por código, así como añadir los datos necesarios para evaluar tu propuesta de mejora.

TslZlcqLdX4jNmBcfbq49NQLW00iDmaql490DT5ZsM1w4yFyQpkcwUC7Hho=

Case 4

JWT

Tenemos el siguiente jwt, cuya clave es “Con KeepCoding aprendemos”.

eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c3VhcmIvIjoiriRG9uIFB1cGl0byBkZSBsb3MgcGFsb3RlcylsInJvbCI6ImIzTm9ybWFSliwiaWF0IjoxNjY3OTMzNTMzfQ.gfhw0dDxp6oixMLXXRP97W4TDTrv0y7B5YjD0U8ixrE

¿Qué algoritmo de firma hemos realizado?

HMAC SHA256 para la encriptación del mensaje

¿Cuál es el body del jwt?

```
{  
  "usuario": "Don Pepito de los palotes",  
  "rol": "isNormal",  
  "iat": 1667933533  
}
```

The screenshot shows the JWT.io website interface. The 'Decoded' tab is active, displaying the following information:

- HEADER: ALGORITHM & TOKEN TYPE:**

```
{  
  "typ": "JWT",  
  "alg": "HS256"  
}
```
- PAYLOAD: DATA:**

```
{  
  "usuario": "Don Pepito de los palotes",  
  "rol": "isNormal",  
  "iat": 1667933533  
}
```
- VERIFY SIGNATURE:**

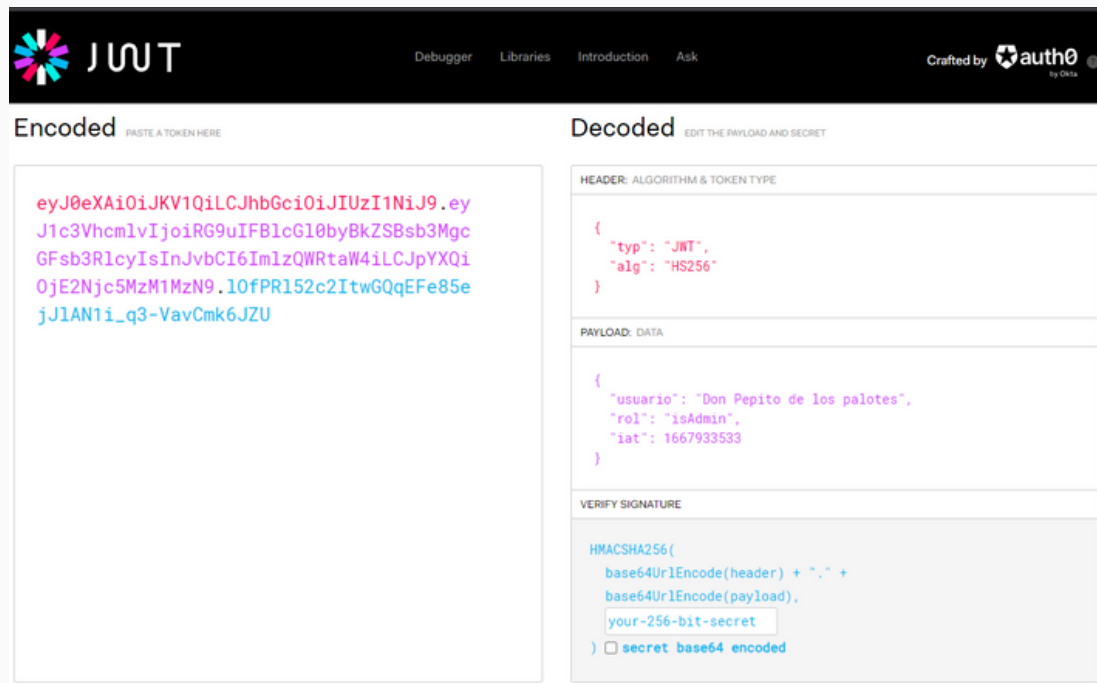
```
HMACSHA256(  
  base64UrlEncode(header) + "." +  
  base64UrlEncode(payload),  
  your-256-bit-secret  
)  
☐ secret base64 encoded
```

Un hacker está enviando a nuestro sistema el siguiente jwt:

eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c3VhcmVlIjoiaRG9uIFBlcGl0byBkZSBsb3MgcGFsb3RlcylsInJvbCI6ImIzQWRtaW4iLCJpYXQiOiE2Njc5MzM1MzN9.krgBkzCBQ5WZ8JnZHuRvmnAZdg4ZMeRNV2CIAODlHRI

¿Qué está intentando realizar?

Cambiar el rol del Pepito a 'Admin'



The image shows the JWT.io web interface. On the left, under the 'Encoded' tab, a JWT token is pasted: `eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c3VhcmVlIjoiaRG9uIFBlcGl0byBkZSBsb3MgcGFsb3RlcylsInJvbCI6ImIzQWRtaW4iLCJpYXQiOiE2Njc5MzM1MzN9.krgBkzCBQ5WZ8JnZHuRvmnAZdg4ZMeRNV2CIAODlHRI`. On the right, under the 'Decoded' tab, the token is broken down into three parts: Header, Payload, and Signature. The Header shows `{ "typ": "JWT", "alg": "HS256" }`. The Payload shows `{ "usuario": "Don Pepito de los palotes", "rol": "isAdmin", "iat": 1667933533 }`. The Signature section shows the algorithm `HMACSHA256(base64UrlEncode(header) + ".", base64UrlEncode(payload), your-256-bit-secret)` with a checkbox for `secret base64 encoded`.

¿Qué ocurre si intentamos validarlo con pyjwt?

Obtenemos un error de tipo `InvalidTokenError`

Case 5

SHA3-KECCAK

El siguiente hash se corresponde con un SHA3 Keccak del texto “En KeepCoding aprendemos cómo protegernos con criptografía”.

bced1be95fbd85d2ffcce9c85434d79aa26f24ce82fbd4439517ea3f072d56fe

¿Qué tipo de SHA3 hemos generado?

SHA3-256

Y si hacemos un SHA2, y obtenemos el siguiente resultado:

**4cec5a9f85dcc5c4c6ccb603d124cf1cdc6dfe836459551a1044f4f2908aa5d6
3739506f6468833d77c07cfd69c488823b8d858283f1d05877120e8c5351c8
33**

¿Qué tipo de SHA2 hemos generado?

SHA2-512

Genera ahora un SHA3 Keccak de 256 bits con el siguiente texto: “En KeepCoding aprendemos cómo protegernos con criptografía.” ¿Qué propiedad destacarías del hash, atendiendo a los resultados anteriores?

302be507113222694d8c63f9813727a85fef61a152176ca90edf1cfb952b19bf
Propiedad de Difusión.

Case 6

HMAC-256

Calcula el hmac-256 (usando la clave contenida en el Keystore) del siguiente texto:

Siempre existe más de una forma de hacerlo, y más de una solución válida.

Se debe evidenciar la respuesta. Cuidado si se usan herramientas fuera de los lenguajes de programación, por las codificaciones es mejor trabajar en hexadecimal.

857d5ab916789620f35bcfe6a1a5f4ce98200180cc8549e6ec83f408e8ca0550

Case 7

SHA

Trabajamos en una empresa de desarrollo que tiene una aplicación web, la cual requiere un login y trabajar con passwords. Nos preguntan qué mecanismo de almacenamiento de las mismas proponemos.

Tras realizar un análisis, el analista de seguridad propone un hash SHA-1. Su responsable, le indica que es una mala opción. ¿Por qué crees que es una mala opción?

1. **Es vulnerable a colisiones**
2. **SHA-1 no usa la función "salt", lo que puede hacer vulnerables a tablas precalculadas**

Después de meditarlo, propone almacenarlo con un SHA-256, y su responsable le pregunta si no lo va a fortalecer de alguna forma. ¿Qué se te ocurre?

1. **Usar salt para agregar aleatoriedad**
2. **Aplicar "stretching", iterar multiples veces el resultado con SHA-256 para agregar complejidad de cálculo**
3. **Usar funciones de derivación de claves (PBKD).**
4. **Usar función de hash HMAC.**

Parece que el responsable se ha quedado conforme, tras mejorar la propuesta del SHA-256, no obstante, hay margen de mejora. ¿Qué propondrías?

1. **Usar SHA3-Keccak, que tiene mayor resistencia criptográfica y de seguridad que SHA-256.**
2. **Usar Argon2 que es uno de los algoritmos más seguros y eficientes para el almacenamiento seguro de contraseñas.**

Case 8

JWT - AES

Tenemos la siguiente API REST, muy simple.

Post /movimientos

Campo	Tipo	Requiere Confidencialidad	Observaciones
idUsuario	Number	N	Identificador
Usuario	String	S	Nombre y Apellidos
Tarjeta	Number	S	

Response

Campo	Tipo	Requiere Confidencialidad	Observaciones
idUsuario	Number	N	Identificador
movTarjeta	Array	S	Formato del ejemplo
Saldo	Number	S	Tendra formato 12300 para indicar 123.00
Moneda	String	N	EUR, DOLLAR

```
{
  "idUsuario": 1,
  "movTarjeta": [{
    "id": 1,
    "comercio": "Comercio Juan",
    "importe": 5000
  }, {
    "id": 2,
    "comercio": "Rest Paquito",
    "importe": 6000
  }],
  "Moneda": "EUR",
  "Saldo": 23400
}
```

Como se puede ver en el API, tenemos ciertos parámetros que deben mantenerse confidenciales. Así mismo, nos gustaría que nadie nos modificase el mensaje sin que nos enterásemos. Se requiere una redefinición de dicha API para garantizar la integridad y la confidencialidad de los mensajes. Se debe asumir que el sistema end to end no usa TLS entre todos los puntos.

¿Qué algoritmos usarías?

Se usará un JWT para enviar la información con cifrado simétrico AES, el cual está diseñado para ser rápido y eficiente, lo que significa que puede cifrar y descifrar datos de manera rápida, lo cual es ideal en respuestas de API como es el caso.

AES también es resistente a ataques criptográficos, y tiene alta compatibilidad al poder ser implementado en diferentes lenguajes de programación.

eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZnVzdWYiOiJESm1vdIRhcmpldGEiOlt7ImkljoxLCJjb2lcmNpbyI6IklNvbWVYI2IvIExp1YW4iLCJpbXBvcnRlIjo1MDAwfSx7fV0sImkljoyLCJjb2lcmNpbyI6IUIlc3QgUGFxdWl0byIsImltcG9ydGUiOiJYwMDAsIk1vbmVkYSI6IklVVUilsIlNhbGRvIjoyMzQwMH0.QOHjuhiKKTGrySmOMiaf9rI9CHquvWY3XgdRLoiwZ6pwJo2xQouFYrn4mU7JEt5BnVNkuH_gRP6zFF8Qton_aqJhIb3y5VeXWHZjCvR1HJgA4aO6hy20R8YGlbcZWq6pvMaj36df69YKLifU2K95EVtEYrD-SzjUqe1C-CeG_G6rL8vbHbBWZu6Ble8neFcfWyx35KJcMbTv_Ww0GBi_Uv-TQmc6dwp6i6Xc-mkW5lLehqB8PG6baOW8CKe2UsJV3vpNwU7GgBwZox5yDwpwHWoOlwF_b8ax0RMJqR4SpszvQonZ6XCrG0kBaxtQV30YQ0PpVnD_SlF26ipc8xQ

[Debugger](#)[Libraries](#)[Introduction](#)[Ask](#)Crafted by auth0
by OktaAlgorithm RS256

Encoded

PASTE A TOKEN HERE

```
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZmVzdWFyaW8iOiJEsIm1vd1RhcmpldGEiOi01t7ImlkIjoxLCJjb21lcmNpbyI6IkNvbWVyY21vIEp1YW4iLCJpbXBvcnRlIjo1MDAwfSx7fV0sImlkIjo1LCJjb21lcmNpbyI6I1Jlc3QgUGFxdWl0byIsIm1tcG9ydGUiOiJYwMDAsIk1vbWVkYSI6IkVVUiIsI1NhbnGRvIjoyMzQwMH0.Q0HjuhiKKTGrySmOMiaf9rI9CHquvWY3XgdRLoiwZ6pwJo2xQouFYrn4mU7JEt5BnVNkuH_gRP6zFF8Qton_aqJhIB3y5VeXWHzjCvR1HJgA4a06hy20R8YG1bCZWq6pvMaj36df69YKLiflJ2K95EVtEYrD-SzjUqe1C-CeG_G6rL8vbHbBWZu6B1e8neFcfWywx35KJcMbtv_Ww0GBi_Uv-TQmc6dwp6i6Xc-mkW5ILehqB8PG6ba0W8CKe21JsJV3vpNwU7GgBwZox5yDwpwHw0IwF_b8ax0RMJqR4SpszvQonZ6XCrG0kBaxtQV30YQ0PpVnD_S1F26ipc8xQ
```

Decoded

EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "RS256",
  "typ": "JWT"
}
```

PAYLOAD: DATA

```
{
  "idUsuario": 1,
  "movTarjeta": [
    {
      "id": 1,
      "comercio": "Comercio Juan",
      "importe": 5000
    },
    {}
  ],
  "id": 2,
  "comercio": "Rest Paquito",
  "importe": 6000,
  "Moneda": "EUR",
  "Saldo": 23400
}
```

VERIFY SIGNATURE

```
RSASHA256(
  base64UrlEncode(header) + "." +
```

Case 9

KCV

Se requiere calcular el KCV de la siguiente clave AES:

A2CFF885901A5449E9C448BA5B948A8C4EE377152B3F1ACFA0148FB3A426DB72

Para lo cual, vamos a requerir el KCV(SHA-256) así como el KCV(AES). El KCV(SHA-256) se corresponderá con los 3 primeros bytes del SHA-256. Mientras que el KCV(AES) se corresponderá con cifrar un texto del tamaño del bloque AES (16 bytes) compuesto con ceros binarios (00), así como un iv igualmente compuesto de ceros binarios. Obviamente, la clave usada será la que queremos obtener su valor de control.

KCV: 5244db

KCV SHA256: db7

Case 10

PGP

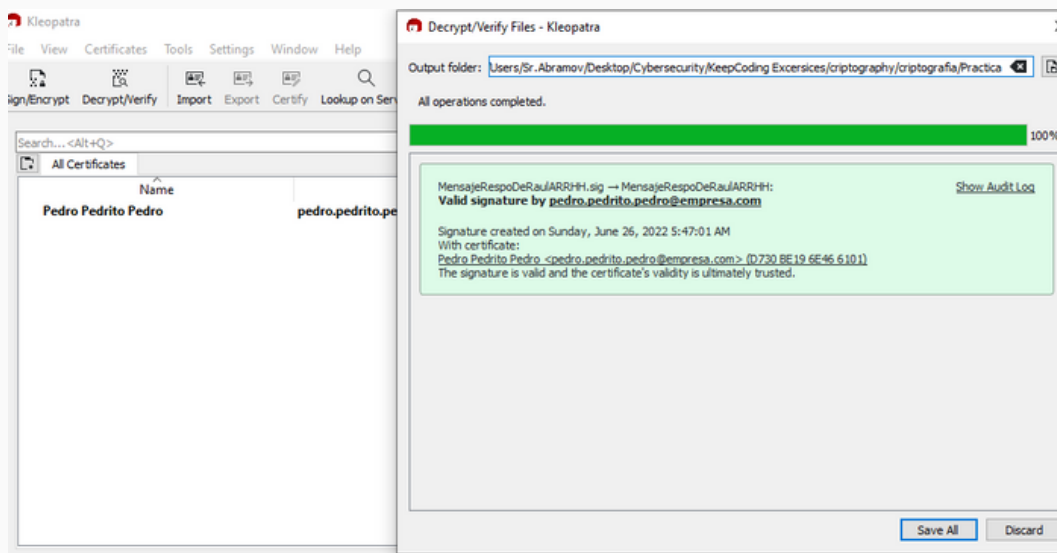
El responsable de Raúl, Pedro, ha enviado este mensaje a RRHH:

Se debe ascender inmediatamente a Raúl. Es necesario mejorarle sus condiciones económicas un 20% para que se quede con nosotros.

Lo acompaña del siguiente fichero de firma PGP (MensajeRespoDeRaulARRHH.txt.sig). Nosotros, que pertenecemos a RRHH vamos al directorio a recuperar la clave para verificarlo. Tendremos los ficheros Pedro-priv.txt y Pedro-publ.txt, con las claves privada y pública.

Las claves de los ficheros de RRHH son RRHH-priv.txt y RRHH-publ.txt que también se tendrán disponibles.

Se requiere verificar la misma, y evidenciar dicha prueba.



Así mismo, se requiere firmar el siguiente mensaje con la clave correspondiente de las anteriores, simulando que eres personal de RRHH.

Viendo su perfil en el mercado, hemos decidido ascenderle y mejorarle un 25% su salario. Saludos.

Por último, cifra el siguiente mensaje tanto con la clave pública de RRHH como la de Pedro y adjunta el fichero con la práctica.

Estamos todos de acuerdo, el ascenso será el mes que viene, agosto, si no hay sorpresas.

Case 11

RSA-OEP

Nuestra compañía tiene un contrato con una empresa que nos da un servicio de almacenamiento de información de videollamadas. Para lo cual, la misma nos envía la clave simétrica de cada videollamada cifrada usando un RSA-OAEP. El hash que usa el algoritmo interno es un SHA-256.

El texto cifrado es el siguiente:

**b72e6fd48155f565dd2684df3ffa8746d649b11f0ed4637fc4c99d18283b32e1709b30c
96b4a8a20d5dbc639e9d83a53681e6d96f76a0e4c279f0dffa76a329d04e3d3d4ad629
793eb00cc76d10fc00475eb76bfbcb1273303882609957c4c0ae2c4f5ba670a4126f2f14
a9f4b6f41aa2edba01b4bd586624659fca82f5b4970186502de8624071be78ccef573d
896b8eac86f5d43ca7b10b59be4acf8f8e0498a455da04f67d3f98b4cd907f27639f4b1
df3c50e05d5bf63768088226e2a9177485c54f72407fdf358fe64479677d8296ad38c6f
177ea7cb74927651cf24b01dee27895d4f05fb5c161957845cd1b5848ed64ed3b0372
2b21a526a6e447cb8ee**

Las claves pública y privada las tenemos en los ficheros clave-rsa-oaep-publ.pem y clave-rsa-oaep-priv.pem.

e2cff885901a5449e9c448ba5b948a8c4ee377152b3f1acfa0148fb3a426db72

Si has recuperado la clave, vuelve a cifrarla con el mismo algoritmo. ¿Por qué son diferentes los textos cifrados?

Porque al realizar la encriptación se hace una serie de calculos que agregan generación de aleatoriedad como capa adicional de seguridad, esto se hace previo a la encriptación de la clave (en este caso) lo que vuelve al cifrado no determinista.

Case 12

AES-GCM

Nos debemos comunicar con una empresa, para lo cual, hemos decidido usar un algoritmo como el AES/GCM en la comunicación. Nuestro sistema, usa los siguientes datos en cada comunicación con el tercero:

Key: E2CFF885901A5449E9C448BA5B948A8C4EE377152B3F1ACFA0148FB3A426DB74

Nonce: 9Yccn/f5nJJhAt2S

1. ¿Qué estamos haciendo mal?

Nunca se debe usar el mismo 'nonce' para encriptar diferentes mensajes ya que este nos garantiza que el resultado del cifrado sea diferente cada vez que ciframos un mensaje.

El uso de un mismo nonce podría:

1. **Confidencialidad Comprometida:** Si se comparan los textos cifrados podríamos deducir información sobre los mensajes originales, lo que podría revelar detalles o incluso permitir el descifrado completo de los mensajes.
2. **Integridad Comprometida:** En AES-GCM se utiliza un 'contador' para cifrar y autenticar bloques de datos individuales, cuando usamos el mismo nonce el contador SE REPITE lo que conduce a la repetición de la secuencia de cifrado y autenticación. Se podría manipular el mensaje o cifrado para obtener resultados indeseados.
3. **Autenticación Comprometida:** AES-GCM usa la función de autenticación "Cifrado en modo Galois", con un 'nonce' repetido se podrían generar textos cifrados y etiquetas de autenticación VALIDOS para mensajes diferentes, es decir, FALSIFICACIÓN o MANIPULACIÓN de mensajes.

Case 13

EC25519

Se desea calcular una firma con el algoritmo PKCS#1 v1.5 usando las claves contenidas en los ficheros clave-rsa-oaep-priv y clave-rsa-oaep-publ.pem del mensaje siguiente:

El equipo está preparado para seguir con el proceso, necesitaremos más recursos.

1. ¿Cuál es el valor de la firma en hexadecimal?

a4606c518e0e2b443255e3626f3f23b77b9d5e1e4d6b3dcf90f7e118d6063950a238
85c6dece92aa3d6eff2a72886b2552be969e11a4b7441bdeadc596c1b94e67a8f941e
a998ef08b2cb3a925c959bcaae2ca9e6e60f95b989c709b9a0b90a0c69d9eaccd86
3bc924e70450ebbbb87369d721a9ec798fe66308e045417d0a56b86d84b305c555
a0e766190d1ad0934a1befbbe031853277569f8383846d971d0daf05d023545d274f1
bdd4b00e8954ba39dacc4a0875208f36d3c9207af096ea0f0d3baa752b48545a5d7
9cce0c2ebb6ff601d92978a33c1a8a707c1ae1470a09663acb6b9519391b61891bf5e
06699aa0a0dbae21f0aaaa6f9b9d59f41928d

Calcula la firma (en hexadecimal) con la curva elíptica ed25519, usando las claves ed25519-priv y ed25519-publ.

bf32592dc235a26e31e231063a1984bb75ffd9dc5550cf30105911ca4560dab52abb4
0e4f7e2d3af828abac1467d95d668a80395e0a71c51798bd54469b7360d

Case 14

HKDF

Necesitamos generar una nueva clave AES, usando para ello una HKDF (HMAC-based Extract-and-Expand key derivation function) con un hash SHA-512. La clave maestra requerida se encuentra en el keystore con la etiqueta “cifrado-sim-aes-256”. La clave obtenida dependerá de un identificador de dispositivo, en este caso tendrá el valor en hexadecimal:

e43bb4067cbcfab3bec54437b84bef4623e345682d89de9948fbb0afedc461a3

¿Qué clave se ha obtenido?

Clave: e716754c67614c53bd9bab176022c952a08e56f07744d6c9edb8c934f52e448a

Case 15

TR31

Nos envían un bloque TR31:

Bloque:

**D0144D0AB00S000042766B9265B2DF93AE6E29B58135B77A2F616C8D515ACDBE6A562
6F79FA7B4071E9EE1423C6D7970FA2B965D18B23922B5B2E5657495E03CD857FD37018
E111B**

Donde la clave de transporte para desenvolver (unwrap) el bloque es:

Key: A1A10101010101010101010101010102

1. ¿Con qué algoritmo se ha protegido el bloque de clave?

Algoritmo: AES

2. ¿Para qué algoritmo se ha definido la clave?

AES, Key block protected using the AES Key Derivation Binding Method

3. ¿Para qué modo de uso se ha generado?

Modo de uso: Encrypt & Decrypt / Wrap & Unwrap

4. ¿Es exportable?

Exportabilidad: Sensitive, exportable under untrusted key

5. ¿Para qué se puede usar la clave?

Uso de la clave: Symmetric Key for Data Encryption

6. ¿Qué valor tiene la clave?

c1c1c1c1c1c1c1c1c1c1c1c1c1c1c1c1