Criptografía

Módulo de criptografía

Diego Martín Olea

Índice

1. 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?

Usando el programa Mascrypt realizamos un XOR con las claves.
Usando la clave fija en código B1EF2ACFE2BAEEFF y teniendo la clave final en memoria que es 91BA13BA21AABB12 el valor que nos da es:

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 B98A15BA31AEBB3F. ¿Qué clave será con la que se trabaje en memoria?

Con la clave fija en código B1EF2ACFE2BAEEFF y la parte dinámica que se modifica en los ficheros de propiedades es B98A15BA31AEBB3F la clave con la que se trabaja en memoria es:

8653f75d31455c0

2. 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/o0QKIWXg3nu1RRz4QWElezdrLAD5LO4USt3aB/i50nvvJbBiG+le1ZhpR84oI=

Para este caso, se ha usado un AES/CBC/PKCS7. Si lo desciframos, ¿qué obtenemos? ¿Qué ocurre si decidimos cambiar el padding a x923 en el descifrado? ¿Cuánto padding se ha añadido en el cifrado?

Se valorará positivamente, obtener el dato de la clave desde el keystore mediante codificación en Python (u otro lenguaje).

Usando la clave del KeyStore y el IV correspondiente al 00 en hexadecimal transformamos con la ayuda de Cyberchef el texto dado que es

TQ9SOMKc6aFS9SlxhfK9wT18UXpPCd505Xf5J/5nLI7Of/o0QKIWXg3nu1RRz4QWElezdrLAD5LO4US
t3aB/i50nvvJbBiG+le1ZhpR84oI=

Este texto está en base64 por lo que primero lo pasamos a hexadecimal y despues lo desciframos y nos sale este texto: Esto es un cifrado en bloque típico. Recuerda, vas por el buen camino. Ánimo.

4573746f20657320756e206369667261646f20656e20626c6f7175652074c3ad-
7069636f2e205265637565726412c2076617320706f7220656c206275656e2063616d696e6f2e20c3816e-
696d6f2e01 tenemos padding01

3. 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-chacha20-256". El nonce "9Yccn/f5nJJhAt2S". El algoritmo que se debe usar es un Chacha20.

¿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.

Ciframos el texto usando la contraseña pero añadiendo el nonce y nos da el mensaje cifrado en base64 junto al mensaje que queremos enviar:

aaxO58TFUlN6AKGbyvfwqu18nI92mVagm85vre9sNTXyIRyUZwZ89cSoQqs=

Mensaje en claro = KeepCoding te enseña a codificar y a cifrar

Para mejorar y garantizar la confidencialidad usaríamos en vez de ChaCha20 el ChaCha20Poly1305 que permite añadir datos asociados y un tag para garantizar la seguridad mediante una autenticación. Adjunto Código en el que cambio el nonce para generar uno aleatorio ya que no debe fijarse nunca el nonce.

4. Tenemos el siguiente jwt, cuya clave es "Con KeepCoding aprendemos".

eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c3VhcmlvIjoiRG9uIFBlcGl0byBkZSBsb3MgcGFsb3RlcyI-sInJvbCI6ImlzTm9ybWFsIiwiaWF0IjoxNjY3OTMzNTMzfQ.gfhw0dDxp6oixMLXXRP97W4TDTrv0y7B5Y-jD0U8ixrE

¿Qué algoritmo de firma hemos realizado?

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

¿Cuál es el body del jwt?

Un hacker está enviando a nuestro sistema el siguiente jwt:
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c3VhcmlvIjoiRG9uIFBlcGl0byBkZSBsb3MgcGFsb3RlcyI-sInJvbCI6ImlzQWRtaW4iLCJpYXQiOjE2Njc5MzM1MzN9.krgBkzCBQ5WZ8JnZHuRvmnAZdg4ZMeRNv-2CIAODlHRI

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

¿Qué está intentando realizar?

Cambiar el rol del usuario y modificarlo para su conveniencia y así conseguir realizar una ataque.

¿Qué ocurre si intentamos validarlo con pyjwt?

Al validarlo nos sale en Cyberchef:

JsonWebTokenError: invalid token

En python:

Password correcta

Token inválido

5. 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

32

bced1be95fbd85d2ffcce9c85434d79aa26f24ce82fbd4439517ea3f072d56fe

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

4cec5a9f85dcc5c4c6ccb603d124cf1cdc6dfe836459551a1044f4f2908aa5d63739506f6468833d77c07cfd-69c488823b8d858283f1d05877120e8c5351c833

¿Qué hash hemos realizado?

SHA512:

4cec5a9f85dcc5c4c6ccb603d124cf1cdc6dfe836459551a1044f4f2908aa5d63739506f6468833d77c07cfd-69c488823b8d858283f1d05877120e8c5351c833

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?

sha3_256

32

302be507113222694d8c63f9813727a85fef61a152176ca90edf1cfb952b19bf

Podemos observar como con un solo punto cambia completamente el hash.Cualquier cambio en el mensaje provoca que no se parezca en nada un hash de otro.

6. 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

result: OK

OK

7. 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?

No se considera seguro ya que se han encontrado colisiones por lo que no se recomienda su uso.

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?

Usando el MAC como cifrado,en concreto el Encrypt-then-MAC(EtM) Para así evitar los ataques por longitud.

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

Agregaría en el algoritmo un valor aleatorio(salt) ya que en los hash se almacenan contraseñas en bases de datos. Tambien es recomendable el uso de algoritmos de criptografía asimétrica, como RSA o ECDSA, junto con SHA-256. Estos algoritmos pueden proporcionar una mayor seguridad.. Puedes usar funciones como PBKDF2 (Password-Based Key Derivation Function 2) o bcrypt para realizar el estiramiento de clave. Y tambien revisar y actualizar la seguridad los algoritmos y comprobar si existen vulnerabilidades.

8. 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?

Usaríamos algoritmos basados en cifrado + mac como por ejemplo un AES +CBC-MAC o un AES+HMAC para así tener un autenticador y garantizar la confidencialidad de los datos y una mejor protección de la API

9. Se requiere calcular el KCV de las 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.

El KCV(SHA-256) se corresponde KCV SHA256: db7df2

Mientras que el KCV(AES) se corresponde KCV: 5244db

10. 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.

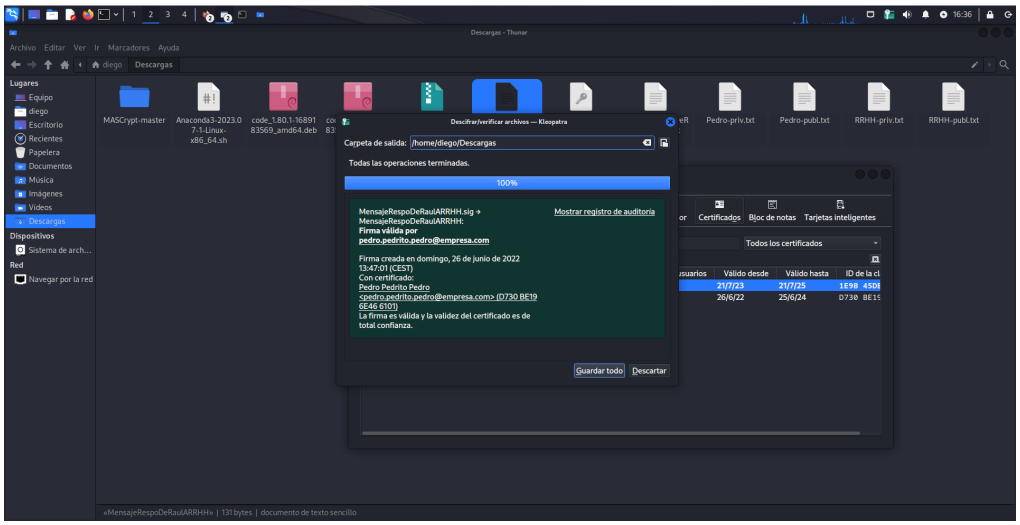
Así mismo, se requiere firmar el siguiente mensaje con la clave correspondiente de las anteriores, simulando que eres personal de RRHH.

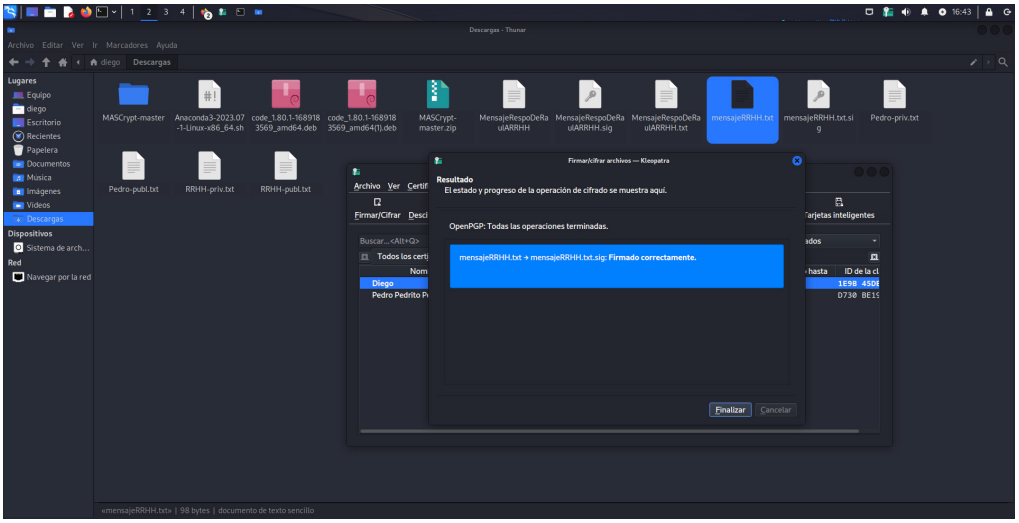
Viendo su perfil en el mercado, hemos decidido ascenderle y mejorarle un 25% su salario. Saludos.

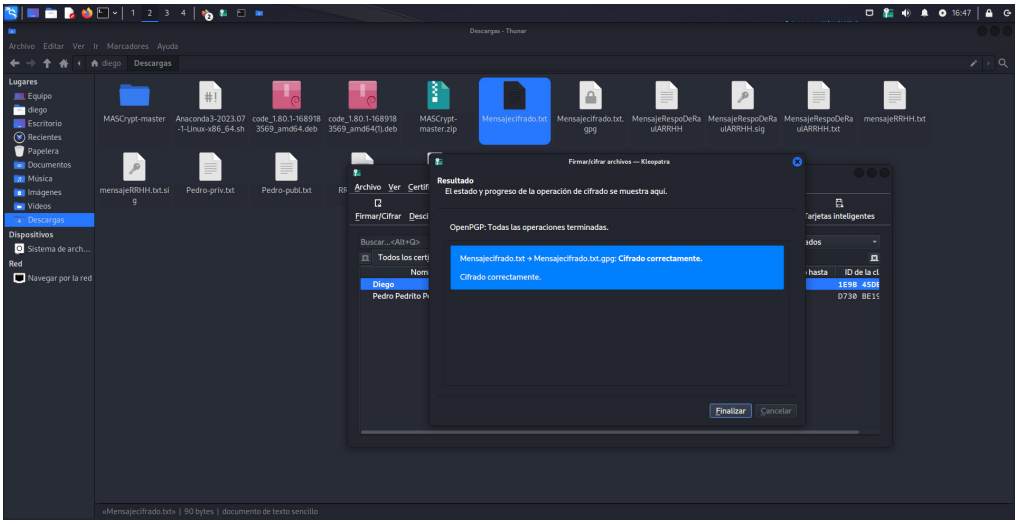
Por último, cifra el siguiente mensaje tanto con la clave pública de RRHH como la de Pedro y adjunta el fichero con la práctica.

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

Usando el programa Kleopatra realizamos todos los ejercicios:







11. 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:

b72e6fd48155f565dd2684df3ffa8746d649b11f0ed4637fc4c99d18283b32e1709b30c96b4a8a20d5db-c639e9d83a53681e6d96f76a0e4c279f0dffa76a329d04e3d3d4ad629793eb00cc76d10fc00475eb76b-fbc12733038826099 57c4c0ae2c4f5ba670a4126f2f14a9f4b6f41aa2edba01b4bd586624659fca82f5b-4970186502de8624071be78ccef573d896b8eac86f5d43ca7b10b59be4acf8f8e0498a455da04f67d3f98b4cd-907f27639f4b1df3c50e05d5bf63768088226e2a9177485c54f72407fdf358fe64479677d8296ad38c6f177ea-7cb74927651cf24b01dee27895d4f05fb5c161957845cd1b5848ed64ed3b03722b21a526a6e447cb8ee

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

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

El descifrado es : e2cff885901a5449e9c448ba5b948a8c4ee377152b3f1acfa0148fb3a426db72

El cifrado es : 3800483afb62973f3ddbdefaaa8958ca4107ffbe11258eaadcd2c8cf2424cb-d6e26d8cfd39a13e01bcf7e0cc4106f21b40b537ef47dca3c0e1a2b063463bfc3cb1b9eca8ad7f76ebe0f75b30143a-f7f17f15e2058501c02b21d47c111539ed7bbce59c6e952c3db51664209a32f35539b8bca674366e6d75b1df7bb-9431004cb4eaa61b55ccccd8f9d844470d679b4d347c8d4588600103c1b058a3d7a4cc39b189204879ab1c76ca-f36a389431ed058c69150c14229f534770a861f8c4b7d7be2233e654f0a778a0277222840b3a386dd53bf4abed-c1ecec576dc56f148bdbf55bb410a1f8f8e53f0f14c46fab4e74a49e8ffe589bbd20cb24b193c63821946

RSA-OAEP utiliza dos máscaras aleatorias, una para enmascarar el mensaje y otra para enmascarar la semilla de un generador de números aleatorios utilizado en el proceso. Esto provoca que con el uso del hash incluso si ciframos lo mismo el resultado será distinto.

12. 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:E2CFF885901B3449E9C448BA5B948A8C4EE322152B3F1ACFA0148FB3A426DB74

Nonce:9Yccn/f5nJJhAt2S

¿Qué estamos haciendo mal?

Que estamos usando el mismo nonce en cada comunicación.

Cifra el siguiente texto:

He descubierto el error y no volveré a hacerlo mal

Usando para ello, la clave, y el nonce indicados. El texto cifrado presentalo en hexadecimal y en base64.

He pasado primero el nonce a hexadecimal y despues ejecuté el codigo.
El texto en base64 es:

Xcu2Jh0PuinOOUMemgE7NMvKKk4Euy2QFJ1h9K/QTWXiq92dhLum64MHCV9QePv8FiVt

Y con un conversor a hexadecimal es:

5dcbb6261d0fba29ce39431e9a013b34cbca2a4e04bb2d90149d61f4afd04d65e2abdd9d84bba6eb-8307095f5078fbfc16256d

13. 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.

¿Cuál es el valor de la firma en hexadecimal?

La firma en hexadecimal:

a4606c518e0e2b443255e3626f3f23b77b9d5e1e4d6b3dcf90f7e118d6063950a23885c6dece92aa3d6eff2a-72886b2552be969e11a4b7441bdeadc596c1b94e67a8f941ea998ef08b2cb3a925c959bcaae2ca9e6e60f95b-989c709b9a0b90a0c69d9eaccd863bc924e70450ebbbb87369d721a9ec798fe66308e045417d0a56b86d84b-305c555a0e766190d1ad0934a1befbbe031853277569f8383846d971d0daf05d023545d274f1bdd4b00e-8954ba39dacc4a0875208f36d3c9207af096ea0f0d3baa752b48545a5d79cce0c2ebb6ff601d92978a33c1a8a-707c1ae1470a09663acb6b9519391b61891bf5e06699aa0a0dbae21f0aaaa6f9b9d59f41928d

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

La firma en hexadecimal calculandola con la curva eliptica ed25519 es:

bf32592dc235a26e31e231063a1984bb75ffd9dc5550cf30105911ca4560dab52abb40e4f7e2d3af828abac1467d-95d668a80395e0a71c51798bd54469b7360d

14. 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?

La clave que se obtiene es

Clave Cifrado: e716754c67614c53bd9bab176022c952a08e56f07744d6c9edb8c934f52e448a

15. Nos envían un bloque TR31:

D0144D0AB00S000042766B9265B2DF93AE6E29B58135B77A2F616C8D515ACDBE6A5626F79FA-7B4071E9EE1423C6D7970FA2B965D18B23922B5B2E5657495E03CD857FD37018E111B

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

A1A10101010101010101010101010102

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

Está protegido con un AES.

¿Para qué algoritmo se ha definido la clave?

Para un AES.

¿Para qué modo de uso se ha generado?

Para cifrar y descifrar y para envolver y desenvolver(wrap,unwrap)

¿Es exportable?

Si, aunque es sensible y se exporta bajo una clave no confiable.

¿Para qué se puede usar la clave?

Para cifrar de forma simetrica.

¿Qué valor tiene la clave?

c1c1c1c1c1c1c1c1c1c1c1c1c1c1c1