

# Actividad 1 – Criptografía

**Diego Mucci**

05/06/2024

Certificado de Profesionalidad: Seguridad Informática

Módulo formativo: Sistemas seguros de acceso y transmisión de datos

## ACTIVIDAD 1 - CRIPTOGRAFÍA

La universidad politécnica de Madrid ofrece recursos muy interesantes para aprender criptografía. En concreto dos colecciones:

- [Intypedia, disponible en Youtube](#)
- [Píldoras formativas Thoth](#)

Os aviso que la calidad, que sí el contenido, de los vídeos no es de lo mejor que se ha hecho.

Ambos son muy interesantes y recomendables para que los vayas viendo a tu elección. El segundo profundiza mucho en el tratamiento matemático de la criptografía y el criptoanálisis.

- Intypedia:
  - o Lección 2, sistemas de clave simétrica. [https://www.youtube.com/watch?v=46Pwz2V-t8Q&list=PL8bSwVy8\\_IcMOdOouph8-mFagDEcrXe1w&index=6&t=2s](https://www.youtube.com/watch?v=46Pwz2V-t8Q&list=PL8bSwVy8_IcMOdOouph8-mFagDEcrXe1w&index=6&t=2s)
  - o Lección 3, sistemas de clave pública, [https://www.youtube.com/watch?v=46Pwz2V-t8Q&list=PL8bSwVy8\\_IcMOdOouph8-mFagDEcrXe1w&index=6&t=2s](https://www.youtube.com/watch?v=46Pwz2V-t8Q&list=PL8bSwVy8_IcMOdOouph8-mFagDEcrXe1w&index=6&t=2s)
  - o Lección 9, introducción al protocolo SSL (hoy TLS), [https://www.youtube.com/watch?v=pOeWmStBOYY&list=PL8bSwVy8\\_IcMOdOouph8-mFagcrXe1w&index=9&t=2s](https://www.youtube.com/watch?v=pOeWmStBOYY&list=PL8bSwVy8_IcMOdOouph8-mFagcrXe1w&index=9&t=2s)
- Proyecto Thoth
  - o Píldora 27, Simétrica VS Asimétrica, [https://www.youtube.com/watch?v=0qfOVm-dtcQ&list=PL8bSwVy8\\_IcNNS5QDLjV7gUg8dleMFSEr&index=27](https://www.youtube.com/watch?v=0qfOVm-dtcQ&list=PL8bSwVy8_IcNNS5QDLjV7gUg8dleMFSEr&index=27)
  - o Píldora 36, Codificación BASE 64, [https://www.youtube.com/watch?v=TvlHDA0J7QM&list=PL8bSwVy8\\_IcNNS5QDLjV7gUg8dleMFSEr&index=36&t=307s](https://www.youtube.com/watch?v=TvlHDA0J7QM&list=PL8bSwVy8_IcNNS5QDLjV7gUg8dleMFSEr&index=36&t=307s)
- Píldora 43, funciones hash, [https://www.youtube.com/watch?v=FRBlcOudwv0&list=PL8bSwVy8\\_IcNNS5QDLjV7gUg8dleMFSEr&index=43](https://www.youtube.com/watch?v=FRBlcOudwv0&list=PL8bSwVy8_IcNNS5QDLjV7gUg8dleMFSEr&index=43)

Contesta a las siguientes preguntas:

### 1. ¿Qué tipo de criptografía usarías para cifrar un pendrive?

Usaría el tipo de Criptografía Simétrica. Preferiblemente utilizaría AES (Advanced Encryption Standard) debido a su:

- Eficiencia: La criptografía simétrica es rápida y adecuada para cifrar grandes volúmenes de datos como los que se encuentran en un pendrive.

- Seguridad: AES es un estándar ampliamente aceptado y robusto para el cifrado de datos, con opciones de claves de 128, 192 o 256 bits que proporcionan diferentes niveles de seguridad.
- Compatibilidad: AES es compatible con la mayoría de los sistemas operativos y herramientas de cifrado de disco disponibles.
- Herramientas Comunes:
  - VeraCrypt: Un software de cifrado de disco libre y de código abierto que soporta AES.
  - BitLocker: Una herramienta de cifrado de disco integrada en algunas versiones de Windows.
  - FileVault: Integrado en macOS para cifrar unidades de almacenamiento.

## 2. ¿Y para enviar un correo?

La Infraestructura de Clave Pública (PKI, por sus siglas en inglés) es una excelente opción para garantizar la seguridad de las comunicaciones por correo electrónico. La PKI es un sistema de gestión de claves y certificados que permite la creación, distribución, almacenamiento y revocación segura de claves públicas y certificados digitales. **Con un cifrado asimétrico sería suficiente, PKI es un plus**

Al utilizar la PKI para cifrar correos electrónicos, los remitentes y destinatarios pueden obtener certificados digitales de una Autoridad Certificadora confiable. Estos certificados digitales contienen la clave pública asociada al propietario del certificado. El remitente puede entonces cifrar el mensaje utilizando la clave pública del destinatario, garantizando que solo el destinatario, que posee la clave privada correspondiente, pueda descifrarlo.

## 3. ¿Qué utilidad real ves a los hashes?

Los hashes son herramientas fundamentales en muchos aspectos de la informática y la seguridad digital. Una función hash es un algoritmo matemático que toma una entrada de datos y la proyecta en un conjunto de datos de tamaño fijo. Proporcionan métodos eficaces para:

- **Verificar la Integridad de Datos**

Utilidad: Asegurar que los datos no han sido alterados durante la transmisión o el almacenamiento.

Ejemplo: Cuando descargas un archivo de Internet, a menudo se proporciona un hash del archivo. Puedes calcular el hash del archivo descargado y compararlo con el hash proporcionado para asegurarte de que el archivo no ha sido corrompido o manipulado.

- **Almacenamiento Seguro de Contraseñas**

Utilidad: Almacenar contraseñas de manera segura en bases de datos.

Ejemplo: En lugar de almacenar contraseñas en texto plano, se almacena un hash de la contraseña. Cuando un usuario intenta iniciar sesión, el sistema *hashea* la contraseña proporcionada y la compara con el hash almacenado.

- **Firmas Digitales y Certificados**

Utilidad: Verificar la autenticidad y la integridad de un documento o mensaje.

Ejemplo: Las firmas digitales utilizan hashes para crear un resumen del documento. El hash se cifra con la clave privada del remitente, permitiendo que el receptor verifique tanto la autenticidad del remitente como la integridad del documento mediante la clave pública correspondiente.

- **Sistemas de duplicación de Datos**

Utilidad: Identificar y eliminar datos duplicados en sistemas de almacenamiento.

Ejemplo: Los sistemas de duplicación de datos utilizan hashes para identificar bloques de datos idénticos y almacenarlos solo una vez, reduciendo el uso de espacio en disco.

- **Índices y Tablas Hash**

Utilidad: Mejorar la eficiencia de las búsquedas y la recuperación de datos.

Ejemplo: Las tablas hash utilizan funciones hash para mapear claves a posiciones en una tabla, permitiendo búsquedas rápidas y eficientes en grandes conjuntos de datos.

- **Generación de Firmas de Mensajes (HMAC)**

Utilidad: Asegurar la integridad y autenticidad de un mensaje utilizando una clave secreta.

Ejemplo: HMAC (Hash-based Message Authentication Code) utiliza un hash y una clave secreta para crear una firma única del mensaje que puede ser verificada

por el receptor para asegurar que el mensaje no ha sido alterado y que proviene del remitente legítimo.

- **Verificación de Software**

Utilidad: Asegurar que el software no ha sido modificado o infectado por malware.

Ejemplo: Los desarrolladores de software a menudo proporcionan hashes de los archivos ejecutables. Los usuarios pueden verificar el hash del archivo descargado contra el hash proporcionado para asegurarse de que el software no ha sido manipulado.

- **Blockchain y Criptomonedas**

Utilidad: Garantizar la integridad y la inmutabilidad de los bloques en una cadena de bloques.

Ejemplo: En las criptomonedas como Bitcoin, los hashes se utilizan para vincular bloques de transacciones, asegurando que una vez que un bloque se agrega a la cadena, no se puede alterar sin cambiar todos los bloques posteriores, lo que es prácticamente imposible.

- **Funciones de Derivación de Claves (KDF)**

Utilidad: Generar claves criptográficas a partir de contraseñas.

Ejemplo: Las KDFs utilizan hashes para convertir una contraseña en una clave criptográfica fuerte que puede ser utilizada para cifrado o autenticación.

#### **4. ¿Es lo mismo codificación que encriptación? Averigua y cita algún sistema de codificación.**

No, la codificación y la encriptación no son lo mismo, aunque ambos procesos implican la transformación de información.

La codificación es el proceso de transformar datos de un formato a otro mediante un esquema conocido, de manera que los datos puedan ser fácilmente recuperados y utilizados por diferentes sistemas. El propósito principal de la codificación es la eficiencia y la facilidad de uso, no la seguridad.

- **Ejemplo de Sistema de Codificación:**

- Base64: Es un sistema de codificación que convierte datos binarios en un formato de texto que utiliza solo caracteres imprimibles. Esto es útil para transmitir datos a través de medios que solo soportan texto (por ejemplo, cuerpos de correos electrónicos, URL, etc.

La encriptación, en cambio, es el proceso de transformar datos en un formato cifrado utilizando una clave, de manera que los datos sean inaccesibles a personas no autorizadas. El propósito principal de la encriptación es proteger la confidencialidad y la integridad de la información.

- Ejemplo de Sistema de Encriptación:



- AES (Advanced Encryption Standard): Es un sistema de encriptación que utiliza claves simétricas para cifrar y descifrar datos. AES es ampliamente utilizado para proteger datos en reposo y en tránsito.

**5. Un texto codificado en BASE 64, ¿puede tener espacios? Quiere decirse, un texto claro codificado en Base64, cuando se obtiene la codificación y se manipula esa codificación, añadiendo espacios intermedios, ¿afecta esta operación al mensaje en claro inicial?**

En la codificación Base64, los espacios no afectan el resultado de la decodificación y no tienen ningún impacto en el mensaje en claro original. Base64 es un sistema de codificación que utiliza un conjunto específico de caracteres ASCII para representar datos binarios de una manera que es segura para su transporte en protocolos de texto como HTTP o correo electrónico. Los caracteres utilizados en Base64 incluyen letras mayúsculas y minúsculas, dígitos y algunos caracteres especiales como +, /, y =.

Cuando se decodifica un texto codificado en Base64, cualquier carácter que no forme parte del conjunto de caracteres de Base64 se ignora. Esto significa que los espacios agregados durante la manipulación de la codificación Base64 no tendrán ningún efecto en el mensaje en claro original, ya que serán ignorados durante la decodificación.


Podemos comprobar esto, yendo al sitio web [base64encode.org](https://base64encode.org) y escribiendo un texto para codificar, en este caso "Actividad práctica 1", esto nos dará un código (QWN0aXZpZGFkIHByw6FjdGljYSAx). Para comprobar lo que hemos explicado, si decodificamos este código tanto sin espacios como con espacios entre los caracteres, nos dará el mismo resultado:

 base64encode.org

## Encode to Base64 format

Simply enter your data then push the encode button.

Actividad práctica 1

 To encode binaries (like images, documents, etc.) use the file upload

UTF-8

Destination character set.


LF (Unix)

Destination newline separator.

☐ Encode each line separately (useful for when you have multiple entries)

☐ Split lines into 76 character wide chunks (useful for MIME).

☐ Perform URL-safe encoding (uses Base64URL format).



 Live mode OFF

Encodes in real-time as you type or paste (supported)


> ENCODE <

Encodes your data into the area below.

QWN0aXZpZGFkIHByw6FjdGJjYSAx

base64decode.org

# Decode and Encode


Encode

Do you have to deal with **Base64** format? Then this site is perfect for you! Use

## Decode from Base64 format

Simply enter your data then push the decode button.


QWN0aXZpZGFkIHByw6FjdGljYSAx

 For encoded binaries (like images, documents, etc.) use the file upload form a little further down.

UTF-8

Source character set.

☐ Decode each line separately (useful for when you have multiple entries).

 Live mode OFF

Decodes in real-time as you type or paste (supports only the UTF-8 format)

< **DECODE** >

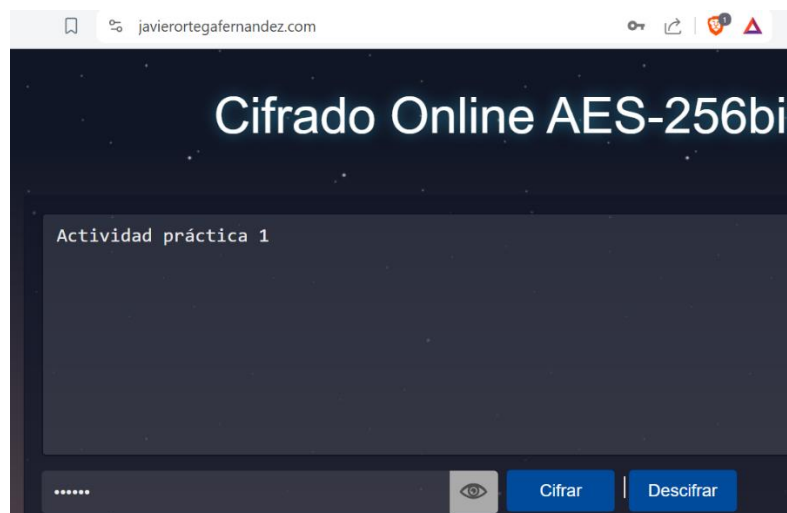
Decodes your data into the area below.

Actividad práctica 1

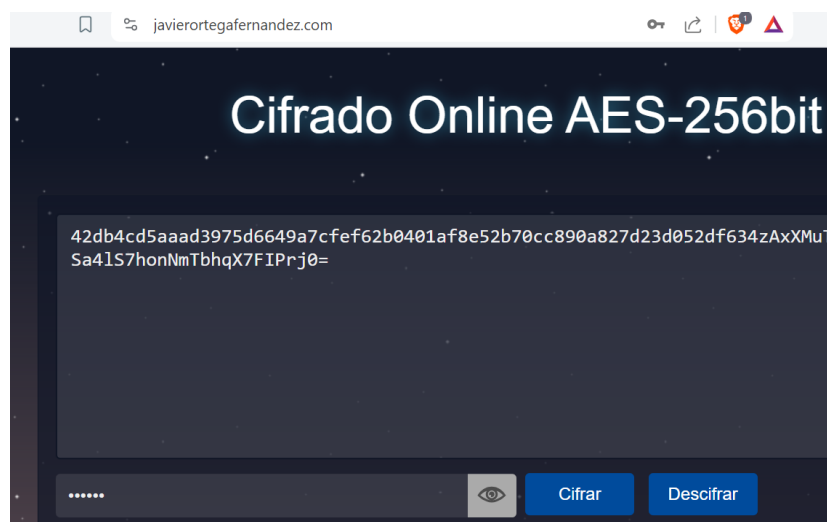
**6. Si fuera un mensaje encriptado, ocurriría lo mismo. Comprueba y documenta ambas cosas.**

En este caso, la inserción de espacios en un mensaje encriptado, sí que puede tener un impacto significativo en la decodificación y el contenido del mensaje en claro original, dependiendo del algoritmo de encriptación utilizado y del modo en que se manejen los espacios durante la encriptación y la decodificación.

Esto lo podemos comprobar yendo al sitio web <https://www.javierortegafernandez.com/> e introducimos un texto y una contraseña cualquiera para poder cifrar:

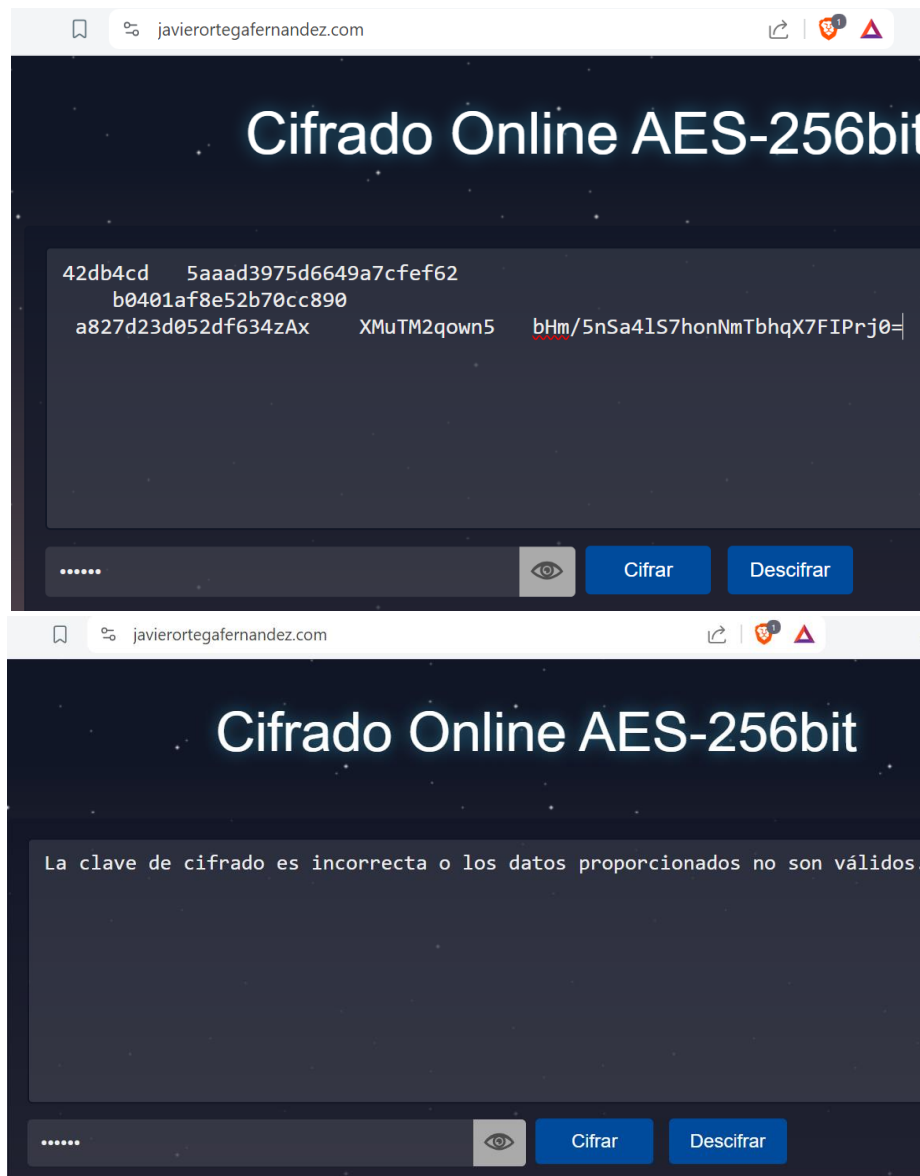


Esto nos dará un mensaje encriptado:



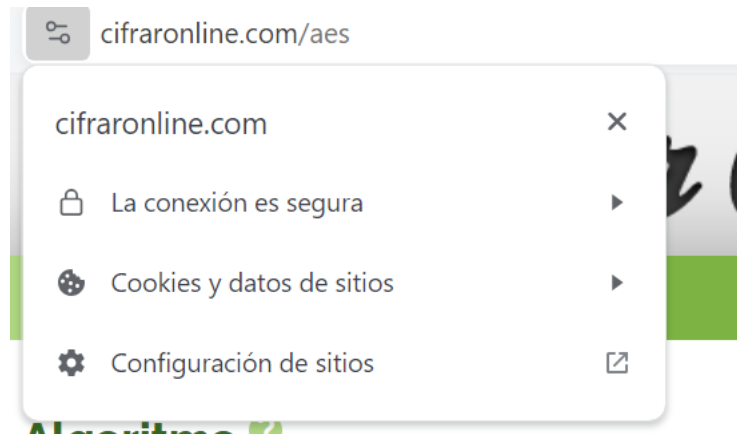


Si ahora le damos al botón de descifrar nos sale el mismo mensaje, pero si introducimos espacios en este mensaje encriptado y le damos a descifrar nos saldrá un error:

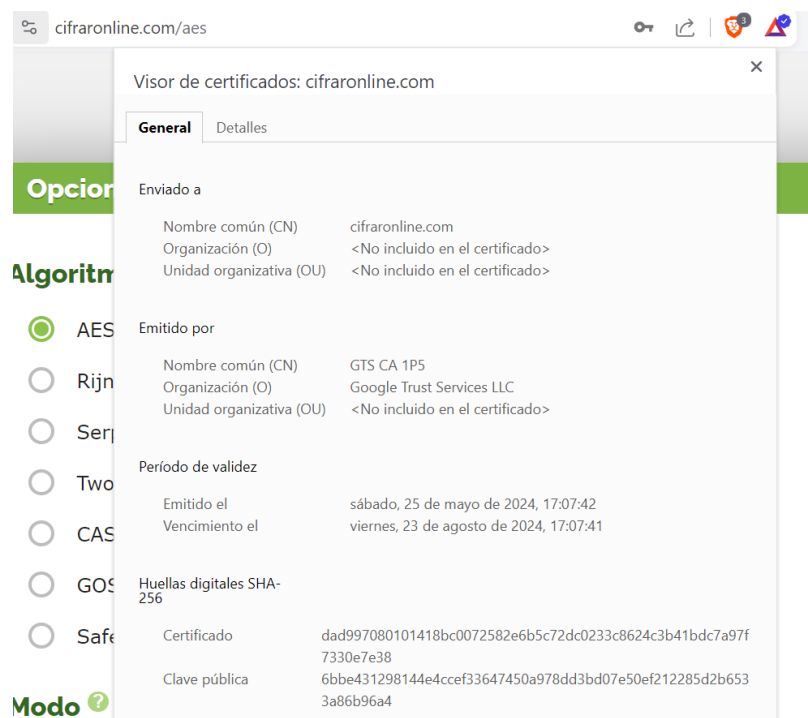
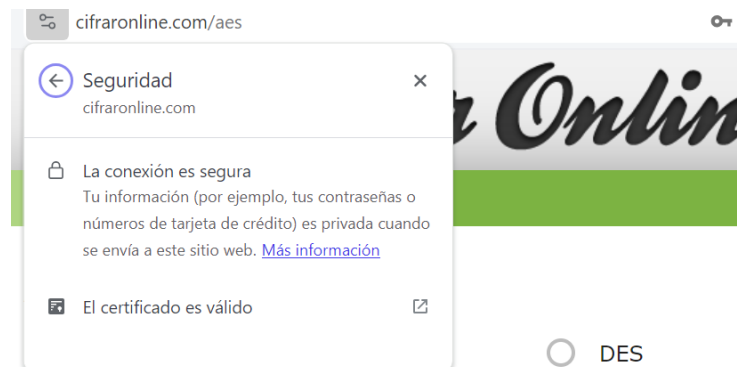


**7. ¿Podemos saber la clave pública de un servidor SSL (TLS)? Gráficamente y por comando, si es posible, en sistemas operativos.**

Podemos saberlo yendo a la barra de búsqueda del navegador (Brave en nuestro caso), clicamos en el símbolo de la izquierda de donde se introduce el nombre del dominio y le damos a "La conexión es segura":



Luego clicamos en “El certificado es válido” y podremos ver el certificado y la clave pública:



## Windows

En Windows, para ver la clave pública podemos usar OpenSSL. Después de instalarlo, abrimos el símbolo del sistema (cmd) o PowerShell, nos dirigimos al directorio donde se encuentra OpenSSL (C:\Program Files\OpenSSL-Win64\bin), ejecutamos el siguiente comando:

`.\openssl.exe s_client -connect www.cifraronline.com:443 | openssl x509 -pubkey -noout`

```
PS C:\Program Files\OpenSSL-Win64\bin> .\openssl.exe s_client -connect www.cifraronline.com:443 | openssl x509 -pubkey -noout
Connecting to 104.21.95.183
depth=2 C=US, O=Google Trust Services LLC, CN=GTS Root R1
verify error:num=20:unable to get local issuer certificate
verify return:1
depth=1 C=US, O=Google Trust Services LLC, CN=GTS CA 1P5
verify return:1
depth=0 CN=cifraronline.com
verify return:1
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEa2aum0katps4VL7qEb4C2
ULW8kXXdoCB9+XcTwce+WBYSd268J4TW6hCEEoLgZkra1+Px9Qyanx3SRZGNH2mJ
q/vY6SHzMSx3vU+/bxBUl2RSrIMQzbNiJeTGUSZa1Ks4WeZfwv1q/FOS0zw1NQ9
wIomk48T4CjTmvOLiLPze4d/j9FIQterRZCc5gPN2U40iQbL11Lax7AvuBmNOa0b
ifigCwZOM91wVxmITFg0i2rcUhY3ocuMz7L8Gn0iIVAUTffx+NLUlNassQX+2oog
okhxpgh54c/ihWtH0iJ6nSKnNzwooB9UusucHeMYz2ZSEduJpFt+7a6rQvQJ3EkdH
dwIDAQAB
```

También podemos ver el certificado de este sitio web ejecutando el siguiente comando:

`.\openssl.exe s_client -connect www.cifraronline.com:443 > cert.txt`

```
PS C:\Program Files\OpenSSL-Win64\bin> .\openssl.exe s_client -connect www.cifraronline.com:443 > cert.txt
Connecting to 104.21.95.183
depth=2 C=US, O=Google Trust Services LLC, CN=GTS Root R1
verify error:num=20:unable to get local issuer certificate
verify return:1
depth=1 C=US, O=Google Trust Services LLC, CN=GTS CA 1P5
verify return:1
depth=0 CN=cifraronline.com
verify return:1
PS C:\Program Files\OpenSSL-Win64\bin>
```

Y en el directorio especificado arriba, podremos ver como se ha generado un archivo de texto con el certificado de este sitio web:

Este equipo > Windows (C:) > Archivos de programa > OpenSSL-Win64 > bin			
Nombre	Fecha de modificación	Tipo	
PEM	03/06/2024 17:54	Carpeta de archivo	
CA.pl	11/04/2024 9:30	Archivo de origen	
capi.dll	11/04/2024 9:30	Extensión de la apl	
cert.txt	03/06/2024 18:16	Documento de tex	
certificate.crt	29/05/2024 17:01	Certificado de seg	
dasync.dll	11/04/2024 9:30	Extensión de la apl	
legacy.dll	11/04/2024 9:30	Extensión de la apl	
libcrypto-3-x64.dll	11/04/2024 9:30	Extensión de la apl	
libssl-3-x64.dll	11/04/2024 9:30	Extensión de la apl	
loader_attic.dll	11/04/2024 9:30	Extensión de la apl	
openssl.exe	11/04/2024 9:30	Aplicación	
ossltest.dll	11/04/2024 9:30	Extensión de la apl	
p_minimal.dll	11/04/2024 9:30	Extensión de la apl	
p_test.dll	11/04/2024 9:30	Extensión de la apl	

cert.txt: Bloc de notas

Archivo Edición Formato Ver Ayuda

CONNECTED(000001C8)

---

#### Certificate chain

```
0 s:CN=www.google.com
  i:C=US, O=Google Trust Services LLC, CN=GTS CA 1C3
  a:PKKEY: id-ecPublicKey, 256 (bit); sigalg: RSA-SHA256
  v:NotBefore: May 13 07:36:00 2024 GMT; NotAfter: Aug 5 07:35:59 2024 GMT
1 s:C=US, O=Google Trust Services LLC, CN=GTS CA 1C3
  i:C=US, O=Google Trust Services LLC, CN=GTS Root R1
  a:PKKEY: rsaEncryption, 2048 (bit); sigalg: RSA-SHA256
  v:NotBefore: Aug 13 00:00:42 2020 GMT; NotAfter: Sep 30 00:00:42 2027 GMT
2 s:C=US, O=Google Trust Services LLC, CN=GTS Root R1
  i:C=BE, O=GlobalSign nv-sa, OU=Root CA, CN=GlobalSign Root CA
  a:PKKEY: rsaEncryption, 4096 (bit); sigalg: RSA-SHA256
  v:NotBefore: Jun 19 00:00:42 2020 GMT; NotAfter: Jan 28 00:00:42 2028 GMT
```

---

#### Server certificate

-----BEGIN CERTIFICATE-----

```
MIIEHTCCA22gAwIBAgIQPFY9uCwqWUKWRc5m1CURDANBgkqhkiG9w0BAQsFADBG
MQswCQYDVQQGEwJVUzEiMCAGA1UEChMZR29vZ2x1IFRydXN0IFN1cnZpY2VzIEEx
QzETMBEGA1UEAxMKR1RTIENBIDFDMzAeFw0yNDA1MTMwNzY2MDBAFw0yNDA4MDUw
NzM1NTIlaMBkxZAVBgNVBAMTDnd3dy5nb29nbGUuY29tMFkwEwYHKoZIzj0CAQYI
KoZIzj0DAQcDQgAE8JrientTtvSmptXfDx/mOHURPnlt9P9zMv0XNyyv5GUYAotm3
eYzjBpFCEoxE+Xz8qd15+MKMhrU8tsdtS810Y6OCAMUwggJhMA4GA1UdDwEB/wQE
AwIHgDATBgNVHUEDDAKBggrBgEFBQcDATAMBgNVHRMBAf8EAjAAMB0GA1UdDgQW
BBTToJ4OgnKu3HZhPhxqN1xnAM6mpzAfBgNVHSMEGDAWgBSKdH+vhc3ulc09nNDi
RhTzcTudJzBqBggrBgEFBQcBAQReMFwwJwYIKwYBBQUHMAGGG2h0dHA6Ly9vY3Nw
LnBraS5nb29nL2d0czFjMzAxZBggrBgEFBQcwAoY1aHR0cDovL3BraS5nb29nL3Jl
cG8vY2VydHMvZ3RzMWMMZmRlcjZAZBgNVHREEEjAQQgg53d3cuZ29vZ2x1LmNvbTAh
BgNVHSAEGjAYMAgGBmeBDAECATAMBggrBgEEAdZ5AgUDMDwGA1UdHwQ1MDMwMaAv
oC2GK2h0dHA6Ly9jcmxzLnBraS5nb29nL2d0czFjMy96ZEFUdDBFeF9Gay5jcmww
ggECBggrBgEEAdZ5AgQCBiHwA04AdQDatr9rP7W2Ip+bwrtca+hwkXFsU1GE
hTS9pD0wSNf7qwAAAY9xGDCEAAAEAwBGMEQCICGG1J9E+nP3Rf2dQ1Od+Fqrw/Hx
gkq5PmZ+LW1a70srAiBjAvR4nVGL4TmB4Y6z2rHAeiSVcqlm0smYy1DhZDA08wB1
```

## Linux

En Linux, es muy similar el proceso de obtención de clave pública, pero aun incluso más fácil, ya que mediante el siguiente comando, obtendremos tanto la clave pública como el certificado:

*openssl s\_client -connect www.cifraronline.com:443 | openssl x509 -pubkey -noout*

```
(kali㉿kali)-[~]
└─$ openssl s_client -connect www.cifraronline.com:443 | openssl x509 -pubkey -noout~
depth=2 C = US, O = Google Trust Services LLC, CN = GTS Root R1
verify return:1
depth=1 C = US, O = Google Trust Services LLC, CN = GTS CA 1P5
verify return:1
depth=0 CN = cifraronline.com
verify return:1
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEa2aUm0katps4VL7qEb4C2
ULW8kXkdoCB9sXcTWce+WBYSd268J4TW6hCEEoLgZkra1+Px9Qyanx3SRZGNH2mJ
q/vY6SHhzMSx3vU+/bxBL2RSrLMQzbNiJeTGUSSza1Ks4WeZfwv1q/FOS0zW1NQ9
w1omk48T4CjTmv0LilPze4d/j9FIQterRZCc5gPN2U40iQbL11LAX7AvuBmNOaOb
ifigCwZOM91wVxmITFg0i2rcUhY3ocuMz7L8Gn0iIVAUtffx+NlULNassQX+2oOg
okhxpgh54c/ihWtH0iJ6nSKnNzwoob9UusucHeMYz2ZSEduJpFt+7a6rQvQJ3Ekdh
dwIDAQAB
-----END PUBLIC KEY-----
-----BEGIN CERTIFICATE-----
MIIEFjCCBGKgAwIBAgIQP7WKOH880NoRTgraX89i6DANBgkqhkiG9w0BAQsFADBG
MQswCQYDVQQGEwJVUzEiMCAGA1UEChMZMj9vZ2xlIFRydXN0IFNlcnZpY2VzIEExM
QzETMBEGA1UEAxMKR1RTIENBIDFQNTAeFw0yNDA1MjUxNTA3NDJhFw0yNDA4MjMx
NTA3NDFAbMBxGTAXBgNVBAMTEG9uZGluZS5jb20wggeiMA0GCSqGSIb3
DQEBAAQAA4IBDwAwggEKAoIBAQQDZpSbSRq2mzhUvuorVgZLZQtbyRcp2gIH35dxNZ
x75YFhJ3brwnhNbqEIQSguBmStrX4/H1D3qfHdJFkY0faYmr+9jpIeHmXlHe9T79
vEFSZXFKuXUDns2Il5MZRJlR24mkW37trqtC9AncSR0d3AgMBAAgJggKNMIIC
iTAOBgNVHQ8BAf8EBAMCBaAwEwYDVR0lBAwwCgYIKwYBBQUHAwEwDAYDVROTAQH/
BAIwADAdBgNVHQ4EFgQU+80XX3kYBStoXZ5htVhS1/f03nwwHwYDVR0jBBgwFoAU
1fyedD8eyt0IL5duK8VfxSv17LgweAYIKwYBBQUHAQEEdBQMDUGCCsGAQUFBzAB
hilodHRwOi8vb2NzcC5wa2kuZ29vZy9zL2d0czFwNS9XNjBfTD0tR3B2WTAxBggr
BgEFBQcwAoYlaHR0cDovL3BraS5nb29nL3JlcG8vY2VydHMvZ3RzMXA1LmRlcjAv
BgNVHREEDAmghBjAwZyYXJvbm9pbnUuY29tghIqLmNpZnJhcm9ubGluZS5jb20w
IQYDVROgBBowGDAIBGZngQwBAGAwDAYKKwYBBAHwEwQIFAzA8BgNVHR8ENTAzMDGg
L6AthitodHRwOi8vY3Jscy5wa2kuZ29vZy9ndHMxcDUvdXR0MmZiZWtKkUuY3J5
MIIBBgYKKwYBBAHwEwQIFAgSB9wSB9ADyAHcA2ra/az+1tiKfm8K7XGvocJFxbLtr
hiU0vaQ9MEjX+6sAAAGPsIIHpgAABAMASDBGAiEAs6C4BYMmlxdd7bhaRVp0n5m3
0NMrbhMo1Li/P1PYvxheXpo+MS0F/yGn1BYwDQYJKoZIhvcNAQELBQADggEBAH7J
gtaivZ9BC+xmwPkeZzeo6UepH5dXh3QpYGErg5qmKbKctbhqDnRnJz0S5NPSMsNJ
glxuNi1CtX0gmrVrWYviDTFEB0AzN6KdWsmTMA4uawC8Ledl6nHw67MwKwRIQpUL
CnadWQ4BtDrqgiypz5RBlqJCj2twVv7Ix91NAELpi9cNho/PyZMc4MDTemB0TjFY
KbmP9jSuCAYtKBFV33pBxWThyyEZ2oIRbhY3UT0SYLKSxVykml/LzhFh6X15XaEeG
Ety7ihQ6+p3fDgqfQTR8fUA1TEVmc5TVKqKWBt8dwtlgbVEBor8eq4o0teVgOvc
1LltMmhP4A3yjjwV98nk=
-----END CERTIFICATE-----
```

Buen trabajo, mira lo de certificado asimétrico y PKI

9.75/10