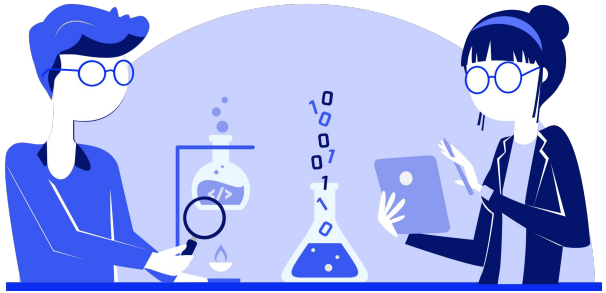
educación IT
CODE YOUR FUTURE

Criptografía y Blockchain

Módulo 2 - Laboratorio adicional

Para poder realizar este laboratorio, se recomienda:

- Revisar contenidos previos.



Ejercicio 1: OpenSSL

OpenSSL es una biblioteca de *software* para aplicaciones que proporcionan comunicaciones seguras a través de redes informáticas. Es ampliamente utilizado por los servidores de Internet, incluyendo la mayoría de los sitios web HTTPS.

OpenSSL contiene una implementación de código abierto de los protocolos SSL y TLS. La biblioteca central, escrita en el lenguaje de programación C, implementa funciones criptográficas básicas y proporciona varias funciones de utilidad.

Algunas variantes derivadas son Agglomerated SSL, LibreSSL y BoringSSL.

Para nuestras prácticas, usaremos la **máquina virtual Kali Linux** que encontrarán en la sección de Bienvenida del módulo 1, llamada ***Instructivo de Instalación***.

1. Abrir una terminal y escribir:

```
┌──(kali㉿kali)-[~]
└─$ man openssl
```

2. Se mostrará la página del manual de OpenSSL. Para salir, presionar **q**.

```
OPENSSL(1SSL)                                   OpenSSL

NAME
       openssl - OpenSSL command line program

SYNOPSIS
       openssl command [ options ... ] [ parameters ... ]

       openssl no-XXX [ options ]

DESCRIPTION
       OpenSSL is a cryptography toolkit implementing the Secure Sock
       Transport Layer Security (TLS v1) network protocols and relate
       required by them.

       The openssl program is a command line program for using the va
       OpenSSL's crypto library from the shell.  It can be used for
```

3. Para visualizar el menú principal de ayuda:

```
┌──(kali㉿kali)-[~]
└─$ openssl help
help:

Standard commands
asn1parse         ca                ciphers
cms               crl               crl2pkcs7
dhparam           dsa               dsaparam
ecparam           enc               engine
fipsinstall       gendsa            genpkey
help              info              kdf
```

Por ejemplo, buscar la ayuda de **enc**.

```
└─$ openssl help enc
Usage: enc [options]

General options:
 -help              Display this summary
 -list              List ciphers
 -ciphers           Alias for -list
 -e                 Encrypt
 -d                 Decrypt
 -p                 Print the iv/key
 -P                 Print the iv/key and exit
```

4. Para ver la versión:

```
└─$ openssl version -a
OpenSSL 3.0.10 1 Aug 2023 (Library: OpenSSL 3.0.10 1 Aug 2023)
built on: Tue Aug  1 20:00:05 2023 UTC
platform: debian-amd64
options:  bn(64,64)
compiler: gcc -fPIC -pthread -m64 -Wa,--noexecstack -Wall -fzero
ECURITY_LEVEL=2 -Wa,--noexecstack -g -O2 -ffile-prefix-map=/buil
tack-protector-strong -Wformat -Werror=format-security -DOPENSSL
DOPENSSL_BUILDING_OPENSSL -DNDEBUG -Wdate-time -D_FORTIFY_SOURCE
```

5. Para ver los cifrados disponibles:

```
└─$ openssl enc -list
Supported ciphers:
-aes-128-cbc               -aes-128-cfb               -aes-128-cfb1
-aes-128-cfb8              -aes-128-ctr               -aes-128-ecb
-aes-128-ofb               -aes-192-cbc               -aes-192-cfb
-aes-192-cfb1              -aes-192-cfb8              -aes-192-ctr
-aes-192-ecb               -aes-192-ofb               -aes-256-cbc
-aes-256-cfb               -aes-256-cfb1              -aes-256-cfb8
-aes-256-ctr               -aes-256-ecb               -aes-256-ofb
-aes128                    -aes128-wrap               -aes192
-aes192-wrap               -aes256                    -aes256-wrap
-aria-128-cbc              -aria-128-cfb              -aria-128-cfb1
```

6. Generar un simple archivo de texto para las pruebas de cifrado y descifrado. Para ello, escribir el siguiente comando:

```
└─$ seq 100 > test.txt
```

7. Esto genera una lista de números del 1 al 100 y la almacena en el archivo ***test.txt***. Se puede comprobar imprimiendo su contenido:

```
└─$ cat test.txt
1
2
3
4
```

8. Cifrar su contenido con AES de 256 bits y modo CBC con *padding* estándar. Ejecutar el siguiente comando:

El primer comando genera la clave (**K**), el segundo el vector de inicialización (**iv**) de 128 bits (del tamaño del bloque). El comando de cifrado utiliza las opciones:

enc	Modo cifrado.
-p	Impresión de resultados.
-aes-256-cbc	Algoritmo de cifrado.
-K	Clave de cifrado.
-iv	Vector de inicialización.
-e	Encriptar.
-in	Archivo a encriptar.
-out	Archivo encriptado.

Este será el contenido del archivo cifrado:

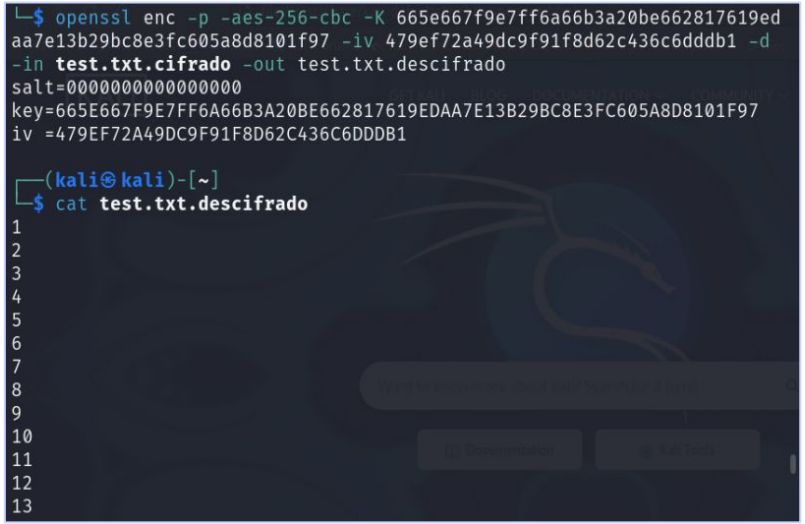
```
└─$ cat test.txt.cifrado

◆v◆◆
xqvv◆◆◆◆◆◆◆o◆\◆◆◆◆K◆◆◆◆◆◆O`j◆3┐E27◆T◆e◆◆◆2◆◆◆◆3◆rT{T0◆◆◆
2L◆/:|aj◆bV◆◆y◆◆◆T◆Y◆◆◆Y◆◆◆?◆{◆ ◆◆_◆◆x◆*◆)◆e[{◆{◆/◆h2JW◆N
◆l◆z◆u◆a◆◆AcL◆>?◆Rh◆◆n 7◆!◆◆◆◆◆
8◆(◆!◆◆.◆◆◆◆wk.v◆ ◆◆>K/Zd9[Q◆◆1 ◆◆:◆I
                                    ◆}ɨ?◆◆=◆0D◆◆9×5◆Q
◆n◆q◆◆◆aF◆◆◆g◆K◆◆m◆◆◆u◆◆m&◆◆ʃu/wsHA*S◆f◆◆◆◆Wk$s
```

9. Para descifrar, la entrada será el archivo cifrado, y la salida, un nuevo archivo ***test.txt.descifrado***.

   Colocar **-d** (desencriptar) en lugar de **-e**.

10. Comprobar que el archivo ha sido correctamente descifrado.



Encriptar con claves asimétricas

1. Generar una clave RSA privada de 4096 bits y almacenarla en un archivo en formato PEM.

```
└─$ openssl genpkey -algorithm RSA -pkeyopt rsa_keygen_bits:4096
rsa_privada.pem
```

Explicación del comando:

`genpkey`	Genera una clave privada.
`-algorithm RSA`	El algoritmo.
`-pkeyopt`	Opciones de clave privada.
`rsa_keygen_bits:4096`	Tamaño de la clave.
`rsa_privada.pem`	El archivo de la clave privada.

Esta será la salida (recortada)

```
└─$ cat rsa_privada.pem
-----BEGIN PRIVATE KEY-----
MIIJQgIBADANBgkqhkiG9w0BAQEFAASCCSwwggkoAgEAAoICAQCd+4kWIYS+CneE
6DROIikXRWUcaztURgXD7FIQ6YdcKqBNxeSLdWkdbJWTEMyFAuNT5PAnKkpPk8IG
K7xWU8sZV0tFYcn+GhUDCT4jC9ThV1Bj3TXcl9Gw1fJSDz55cVOlcelzexgtCzuo
o500hcxKItcsQiqkYlU+Ew0g5su3Jst/1Wz6CobiJDCxPM9wfIaE+K+TGHyl7bNF
kmxYV8RxzUK9VrjXWsEj7OlDkXBaJ8VRqSTjvHTnZd+QKbtNsrxAbk7MUQHZPNPT
qkFnhUH3f4NSX7iuS2HvMPagLQ/0lScX32zyWIRMo8kZ9Hq0unIXMu4mfv8Y3zTH
JiTABVIDl/eOOYkDEb62R6oHvyyhiJMUAs6uE1iDeC64Yfp+0N5ccOylD2oqaWXU
Ws0PJBo12iaL8koZ38/AvUSaQrYKLMGx/hthSpm/UdViSxyHkNbJ8JtpAFKeAYNn
hE3fmnTkg0aRYTmtoDeMMQR5io5kuA==
-----END PRIVATE KEY-----
```

2. Se puede inspeccionar la estructura de la clave privada con el siguiente comando:

```
└─$ openssl pkey -in rsa_privada.pem -noout -text
```

3. Generar la clave pública a partir de la clave privada:

```
└─$ openssl pkey -in rsa_privada.pem -pubout -out rsa_publica.pem


┌──(kali㉿kali)-[~]
└─$ cat rsa_publica.pem
-----BEGIN PUBLIC KEY-----
MIICIjANBgkqhkiG9w0BAQEFAAOCAg8AMIICCgKCAgEAnfuJFiGEvgp3hOg0TiIp
F0VlHGs7VEYFw+xSEOmHXCqgTcXki3VpHWyVkxDMhQLjU+TwJypKT5PCBiu8VlPL
GVdLRWHJ/hoVAwk+IwvU4VdQY9013JfRsNXyUg8+eXFTpXHpc3sYLQs7qKOdNIXM
SiLXLEIqpGJVPhMNIObLtybLf9Vs+gqG4iQwsTzPcHyGhPivkxh8pe2zRZJsWFfE
cc1CvVa411rBI+zpQ5FwWifFUakk47×052XfkCm7TbK8QG5OzFEB2TzT06pBZ4VB
93+DUl+4rkth7zD2oC0P9JUnF99s8liETKPJGfR6tLpyFzLuJn7/GN80×5ruQBz3
/3H+eH8aw09ZMWVDWQL84U9ica/g6rq0cc4zVYMWhbMNOERvVibbfaXRp12rLLQq
07ZmOWqLj//+Wdvk6wc07v4tHvVDNPPZ/tyluJcPEbb3Y4fD5HXtwuSPhz8auJeK
PMRyPV9crgnxg5BykWleEa78aCAQQIx+1gVDgemIyrXuEsB/e1sDzG2EwWUgEFjN
8RqZD74HU6BWyKJtHBnT7p2HRsA0ojycPPcHjK4TlJvcebC3a8GoA3/93onJ5QNT
fQl0450LVo3fryfmY/Oj6lt61wbIiYEETNfX79kokJjciLsz3u1dwtsh7MZ/HM6Z
gNIZq32yKUwkOwGEIqhjbaECAwEAAQ==
-----END PUBLIC KEY-----
```

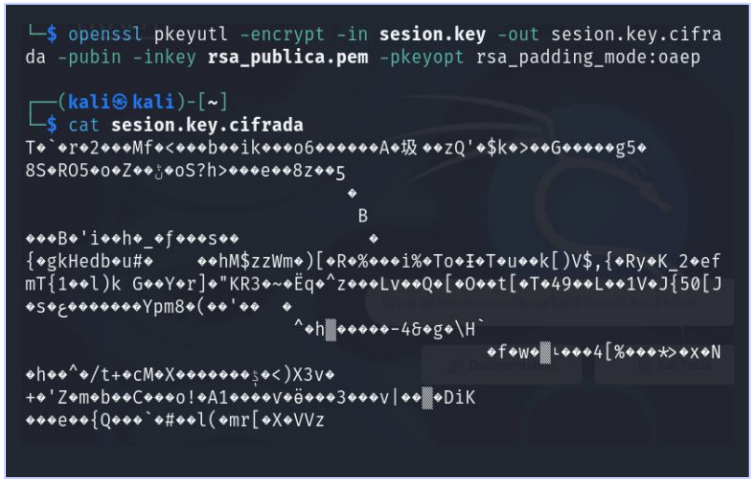
A primera vista se observa el tamaño mucho menor de la clave pública.

Se puede usar el mismo comando usado con la clave privada para ver su estructura.

Según lo visto anteriormente, RSA se utiliza para cifrar una clave de sesión. Generar una de 256 bits que almacenaremos en el archivo ***sesion.key***.

```
-$ openssl rand -out sesion.key 32
```

4. Cifrar la clave de sesión con la clave pública RSA:



Detalle del comando:

pkeyutl	Utilidades de claves asimétricas.
-encrypt	Encriptar.
-in	Archivo a encriptar.
-out	Archivo encriptado.
-pubin	Usar la clave pública.
-inkey	Archivo de la clave.
-pkeyopt	Opciones de clave.
rsa_padding_mode:oaep	*Optimal Asymmetric Encryption Padding* (PKCS#1 v2.0)

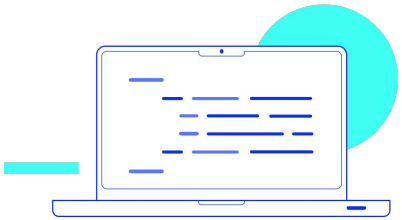
5. Descifrar la clave de sesión con la clave privada RSA:

```
└─$ openssl pkeyutl -decrypt -in sesion.key.cifrada -out sesion.key
.descifrada -inkey rsa_privada.pem -pkeyopt rsa_padding_mode:oaep
```

6. User el siguiente comando para comparar ambas claves.
   Se comprobará que se ha descifrado correctamente.

```
└─$ sha256sum sesion.key sesion.key.descifrada
874b6a69f8e80d109bba24d438749d5112a6d42ab5e63c0126bb38af696f530e  s
esion.key
874b6a69f8e80d109bba24d438749d5112a6d42ab5e63c0126bb38af696f530e  s
esion.key.descifrada
```

7. Por último, generar claves con criptografía de curva elíptica. Consultar las curvas disponibles (salida recortada).



```
$ openssl ecparam -list_curves
secp112r1 : SECG/WTLS curve over a 112 bit prime field
secp112r2 : SECG curve over a 112 bit prime field
secp128r1 : SECG curve over a 128 bit prime field
secp128r2 : SECG curve over a 128 bit prime field
secp160k1 : SECG curve over a 160 bit prime field
secp160r1 : SECG curve over a 160 bit prime field
secp160r2 : SECG/WTLS curve over a 160 bit prime field
secp192k1 : SECG curve over a 192 bit prime field
secp224k1 : SECG curve over a 224 bit prime field
secp224r1 : NIST/SECG curve over a 224 bit prime field
secp256k1 : SECG curve over a 256 bit prime field
secp384r1 : NIST/SECG curve over a 384 bit prime field
secp521r1 : NIST/SECG curve over a 521 bit prime field
prime192v1: NIST/X9.62/SECG curve over a 192 bit prime field
prime192v2: X9.62 curve over a 192 bit prime field
prime192v3: X9.62 curve over a 192 bit prime field
prime239v1: X9.62 curve over a 239 bit prime field
prime239v2: X9.62 curve over a 239 bit prime field
prime239v3: X9.62 curve over a 239 bit prime field
prime256v1: X9.62/SECG curve over a 256 bit prime field
sect113r1 : SECG curve over a 113 bit binary field
sect113r2 : SECG curve over a 113 bit binary field
sect131r1 : SECG/WTLS curve over a 131 bit binary field
```

8. Elegir la curva `secp512r1` y generar la clave privada en el archivo ***ec_privada.pem***. Podrá observarse el tamaño mucho menor que en el caso de RSA.

```
┌──(kali㉿kali)-[~]
└─$ openssl genpkey -algorithm EC -pkeyopt ec_paramgen_curve:secp521r1
 -out ec_privada.pem

┌──(kali㉿kali)-[~]
└─$ cat ec_privada.pem
-----BEGIN PRIVATE KEY-----
MIHuAgEAMBAGByqGSM49AgEGBSuBBAAjBIHWMIHTAgEBBEIAhS5xObztZ6Fyrf1g
sLZ14ltjh4wAwmyNPXt4uEJhPbEB3KPPVd1Akd0nPrnO7H3+GeuUxkRszLLEJqlr
JFlb1U2hgYkDgYYABADvomlfDjhyzZTkNfB2ujg0eH845OzuDRSOj3fKR/BdKJf5
z3dOZzTX6Ce7dEScyrRYfN0YPWYhkZGNxS2qc/yLeACD4MNe+sbklhH8ugcAgiR8
4QG2m0QOUaK/4eXfWhbcr4CSgJblXqF5B3DDHfC73XtlF32gGadK6qaDFwIDX4YV
zg==
-----END PRIVATE KEY-----
```

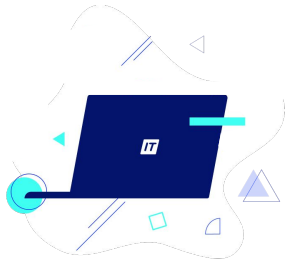
Para ver su estructura, ejecutar:

```
└─$ openssl pkey -in ec_privada.pem -noout -text
Private-Key: (521 bit)
priv:
    00:85:2e:71:39:bc:ed:67:a1:72:ad:fd:60:b0:b6:
    75:e2:5b:63:87:8c:00:c2:6c:8d:3d:7b:78:b8:42:
    61:3d:b1:01:dc:a3:cf:55:dd:40:91:dd:27:3e:b9:
    ce:ec:7d:fe:19:eb:94:c6:44:6c:cc:b2:c4:26:a9:
    6b:24:59:5b:d5:4d
pub:
    04:00:ef:a2:69:5f:0e:38:72:cd:94:e4:35:f0:76:
    ba:38:34:78:7f:38:e4:ec:ee:0d:14:8e:8f:77:ca:
    47:f0:5d:28:97:f9:cf:77:4e:67:34:d7:e8:27:bb:
    74:44:9c:ca:b4:58:7c:dd:18:3d:66:21:91:91:8d:
    c5:2d:aa:73:fc:8b:78:00:83:e0:c3:5e:fa:c6:e4:
    96:11:fc:ba:07:00:82:24:7c:e1:01:b6:9b:44:0e:
    51:a2:bf:e1:e5:df:5a:16:dc:af:80:92:80:96:e5:
    5e:a1:79:07:70:c3:1d:f0:bb:dd:7b:65:17:7d:a0:
    19:a7:4a:ea:a6:83:17:02:03:5f:86:15:ce
ASN1 OID: secp521r1
NIST CURVE: P-521
```

9. Generar la clave pública.

```
└─$ openssl pkey -in ec_privada.pem -pubout -out ec_publica.pem

┌──(kali㉿kali)-[~]
└─$ cat ec_publica.pem
-----BEGIN PUBLIC KEY-----
MIGbMBAGByqGSM49AgEGBSuBBAAjA4GGAAQA76JpXw44cs2U5DXwdro4NHh/OOTs
7g0Ujo93ykfwXSiX+c93Tmc01+gnu3REnMq0WHzdGD1mIZGRjcUtqnP8i3gAg+DD
XvrG5JYR/LoHAIIkfOEBtptEDlGiv+Hl31oW3K+AkoCW5V6heQdwwx3wu917ZRd9
oBmnSuqmgxcCA1+GFc4=
-----END PUBLIC KEY-----
```



Ver su estructura.

```
└─$ openssl pkey -in ec_publica.pem -pubin -noout -text
Public-Key: (521 bit)
pub:
    04:00:ef:a2:69:5f:0e:38:72:cd:94:e4:35:f0:76:
    ba:38:34:78:7f:38:e4:ec:ee:0d:14:8e:8f:77:ca:
    47:f0:5d:28:97:f9:cf:77:4e:67:34:d7:e8:27:bb:
    74:44:9c:ca:b4:58:7c:dd:18:3d:66:21:91:91:8d:
    c5:2d:aa:73:fc:8b:78:00:83:e0:c3:5e:fa:c6:e4:
    96:11:fc:ba:07:00:82:24:7c:e1:01:b6:9b:44:0e:
    51:a2:bf:e1:e5:df:5a:16:dc:af:80:92:80:96:e5:
    5e:a1:79:07:70:c3:1d:f0:bb:dd:7b:65:17:7d:a0:
    19:a7:4a:ea:a6:83:17:02:03:5f:86:15:ce
ASN1 OID: secp521r1
NIST CURVE: P-521
```

En el próximo módulo se verá cómo implementar la firma digital (con la clave privada) y la verificación de firma (con la clave pública).

¡Sigamos trabajando!