Actividad 16. Cifrado con Openssl

OpenSSL es un proyecto de software libre basado en SSLeay, desarrollado por Eric Young y Tim Hudson.

Consiste en un robusto **paquete de herramientas de administración y bibliotecas relacionadas con la criptografía**, que suministran funciones criptográficas a otros paquetes como OpenSSH y navegadores web (para acceso seguro a sitios HTTPS).

Estas herramientas ayudan al sistema a implementar el Secure Sockets Layer (SSL), así como otros protocolos relacionados con la seguridad, como el Transport Layer Security (TLS). OpenSSL también permite crear certificados digitales que pueden aplicarse a un servidor, por ejemplo, Apache.

Es útil para operaciones criptográficas como:

- Creación de parámetros clave RSA, DH y DSA;
- Creación de certificados X.509, CSR y CRL;
- calcular resúmenes de mensajes;
- Cifrado y descifrado con cifras;
- Probar clientes y servidores SSL/TLS;
- Gestión de correo firmado o cifrado S/MIME.

Los siguientes algoritmos de criptografía son soportados por la herramienta:

Para cifrado

AES, Blowfish, Camellia, SEED, CAST-128, DES, IDEA, RC2, RC4, RC5, Triple DES, GOST 28147-893

Funciones HASH

MD5, MD4, MD2, SHA-1, SHA-2, RIPEMD-160, MDC-2, GOST R 34.11-944

Clave pública

RSA, DSA, Intercambio de claves Diffie–Hellman, curvas elípticas

INSTALACIÓN:

En Linux viene incorporado.

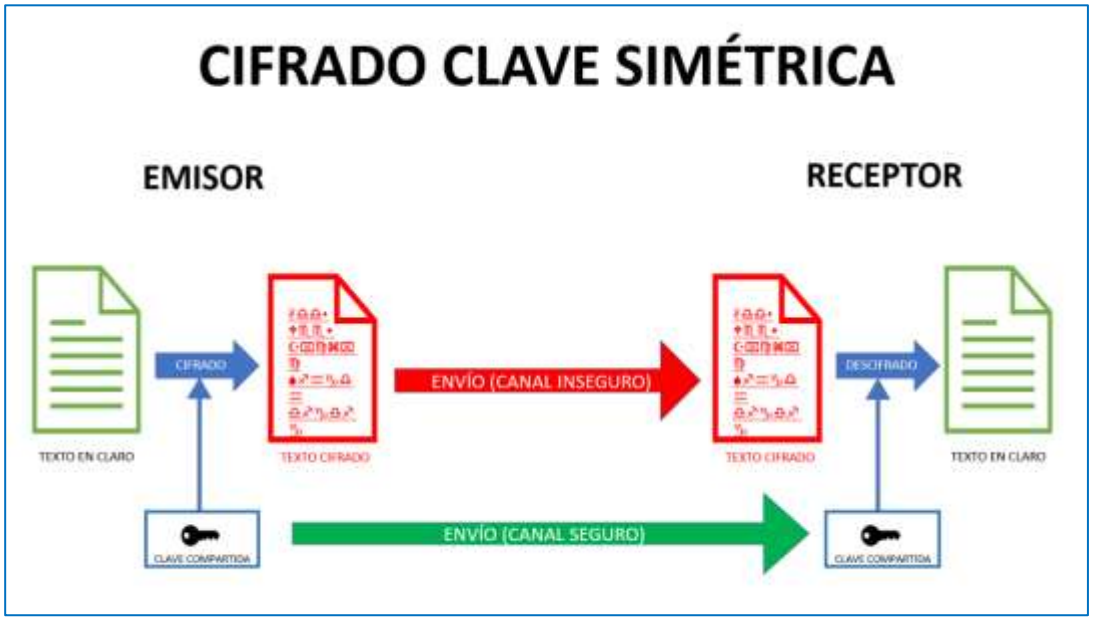
Cómo instalar y configurar Openssl Suite en Windows

OpenSSL es una biblioteca y herramienta de seguridad muy popular. OpenSSL proporciona una gran cantidad de algoritmos de seguridad, estándares, protocolos como biblioteca y herramienta. OpenSSL se desarrolló principalmente en la comunidad de software libre y Linux, pero esto no significa que Windows no utilice la biblioteca y las herramientas de OpenSSL. En este tutorial aprenderemos a instalar y configurar OpenSSL en los sistemas operativos de Windows.

Descargar Binarios OpenSSL

Por defecto, los binarios de OpenSSL para Windows no proporcionan a los desarrolladores de OpenSSL. Pero los binarios son proporcionados por la comunidad compilados en versiones de 32 o 64 bits. Uno de los binarios proporcionados por la comunidad puede ser descargado desde el siguiente enlace.

CIFRADO Y DESCIFRADO SIMÉTRICO:



La forma general del comando será:

```
openssl -openssl command [options…] [parameters…]

Openssl list-standard-commands | digest-commands | cipher-commands | cipher-
algorithms | digest-algorithms | mac-algorithms | public-key-algorithms
```

Vamos a crear un documento de texto plano:

```
┌──(kali㉿kali)-[~]
└─$ cat texto.txt
En un lugar de la Mancha, de cuyo nombre no quiero acordarme, no ha mucho tiempo
que vivia un hidalgo de los de lanza en astillero, adarga antigua, rocin flaco y
galgo corredor. Una olla de algo mas vaca que carnero, salpicon las mas noches, d
uelos y quebrantos los sabados, lentejas los viernes, algun palomino de anadidura
 los domingos, consumian las tres partes de su hacienda. El resto della concluian
 sayo de velarte, calzas de velludo para las fiestas con sus pantuflos de lo mism
o, los dias de entre semana se honraba con su vellori de lo mas fino. Tenia en su
 casa una ama que pasaba de los cuarenta, y una sobrina que no llegaba a los vein
te, y un mozo de campo y plaza, que asi ensillaba el rocin como tomaba la podader
a. Frisaba la edad de nuestro hidalgo con los cincuenta anos, era de complexion r
ecia, seco de carnes, enjuto de rostro; gran madrugador y amigo de la caza. Quier
en decir que tenia el sobrenombre de Quijada o Quesada (que en esto hay alguna di
ferencia en los autores que deste caso escriben), aunque por conjeturas verosimil
es se deja entender que se llama Quijana; pero esto importa poco a nuestro cuento
; basta que en la narracion del no se salga un punto de la verdad.
```

Vamos a cifrar el documento prueba.txt utilizando el cifrado aes-256-cbc con el siguiente comando:

```
openssl enc -aes-256-cbc -in texto.txt -out texto.aes
```

Va a solicitar una contraseña, que hay que verificar.

```
┌──(kali㉿kali)-[~]
└─$ openssl enc -aes-256-cbc -in texto.txt -out texto.aes
enter AES-256-CBC encryption password:
Verifying - enter AES-256-CBC encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
```

A continuación, visualizamos el fichero encriptado (texto.aes):

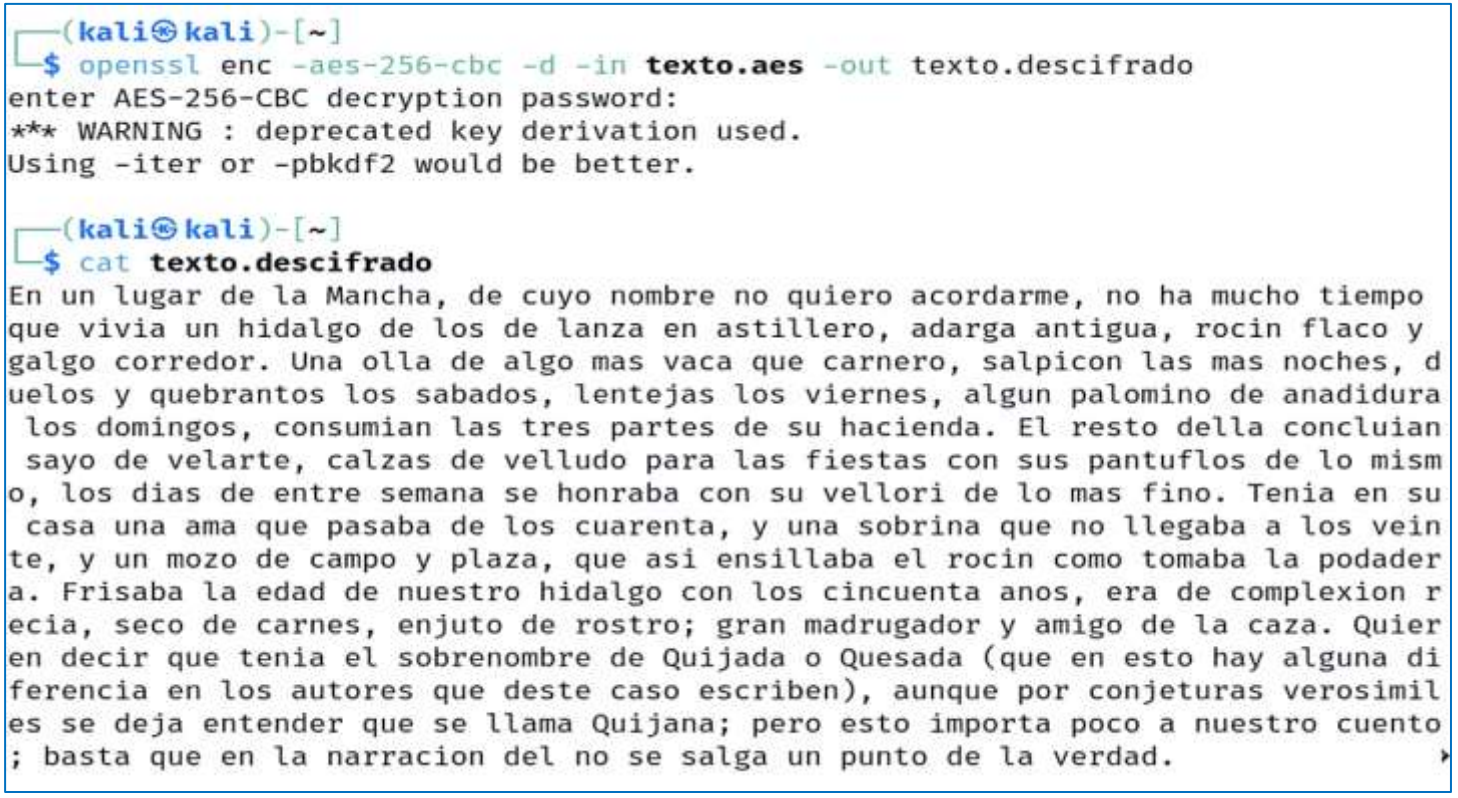
```
cat texto.aes
```

Vamos a descifrar el documento texto.aes con el siguiente comando:

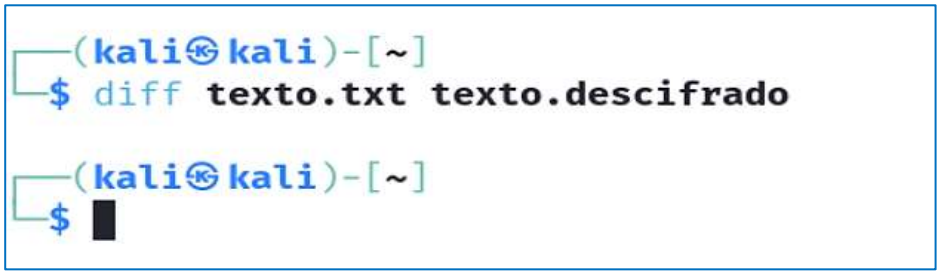
```
openssl enc -aes-256-cbc -d -in texto.aes -out texto.descifrado
```

A continuación, visualizamos el fichero encriptado:

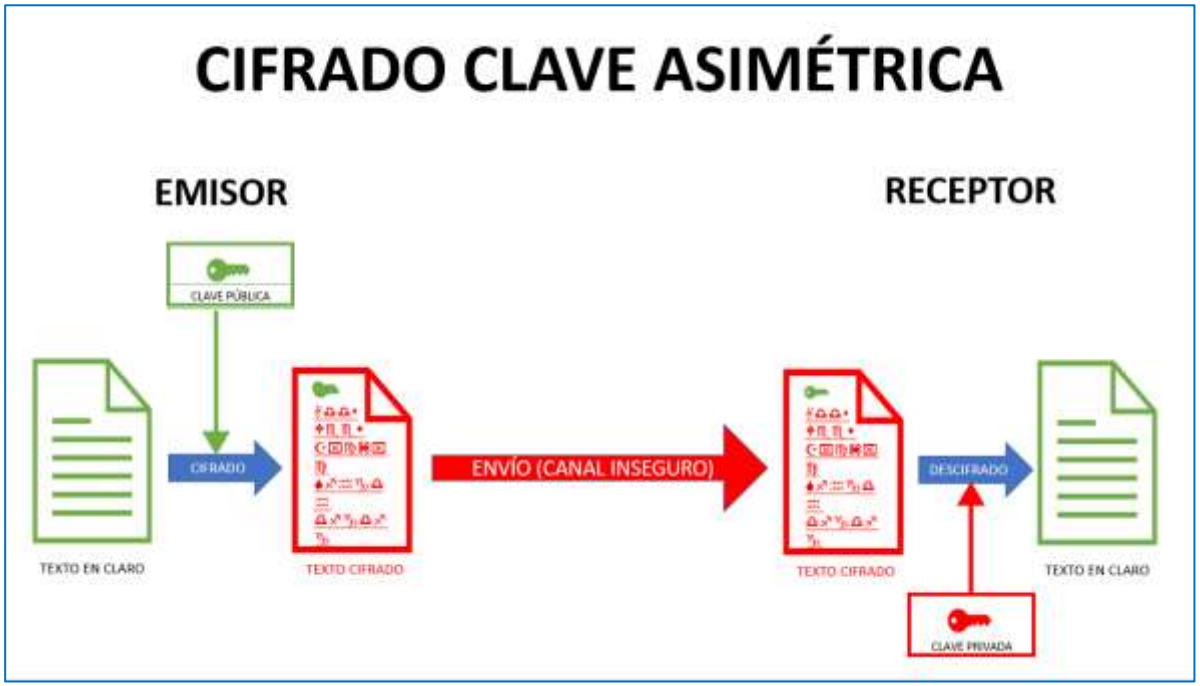
```
cat texto.descifrado
```



Podemos comprobar que el fichero texto.descifrado es igual a texto.txt con el comando diff:



CIFRADO Y DESCIFRADO ASIMÉTRICO:



Vamos a crear un documento de texto plano:

```
┌──(kali㉿kali)-[~]
└─$ cat texto.txt
En un lugar de la Mancha, de cuyo nombre no quiero acordarme, no ha mucho tiempo
que vivia un hidalgo de los de lanza en astillero, adarga antigua, rocin flaco y
galgo corredor. Una olla de algo mas vaca que carnero, salpicon las mas noches, d
uelos y quebrantos los sabados, lentejas los viernes, algun palomino de anadidura
 los domingos, consumian las tres partes de su hacienda. El resto della concluian
 sayo de velarte, calzas de velludo para las fiestas con sus pantuflos de lo mism
o, los dias de entre semana se honraba con su vellori de lo mas fino. Tenia en su
 casa una ama que pasaba de los cuarenta, y una sobrina que no llegaba a los vein
te, y un mozo de campo y plaza, que asi ensillaba el rocin como tomaba la podader
a. Frisaba la edad de nuestro hidalgo con los cincuenta anos, era de complexion r
ecia, seco de carnes, enjuto de rostro; gran madrugador y amigo de la caza. Quier
en decir que tenia el sobrenombre de Quijada o Quesada (que en esto hay alguna di
ferencia en los autores que deste caso escriben), aunque por conjeturas verosimil
es se deja entender que se llama Quijana; pero esto importa poco a nuestro cuento
; basta que en la narracion del no se salga un punto de la verdad.
```

Generación de claves privada y pública

Vamos a generar la clave privada utilizando el algoritmo aes128. De esta forma, solicitará una contraseña para generar la clave, cada vez que se utilice. Para ello utilizamos el siguiente comando:

```
openssl genrsa -aes128 -out claveprivada.pem
```

Va a solicitar una clave:

```
┌──(kali㉿kali)-[~]
└─$ openssl genrsa -aes128 -out claveprivada.pem
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
```

Y ya tenemos la clave privada:

```
┌──(kali㉿kali)-[~]
└─$ cat claveprivada.pem
-----BEGIN ENCRYPTED PRIVATE KEY-----
MIIFLTBXBgkqhkiG9w0BBQ0wSjApBgkqhkiG9w0BBQwwHAQIcw50dcvUsDICAggA
MAwGCCqGSIb3DQIJBQAwHQYJYIZIAWUDBAECBBDhmF+jSdHcTnXYXAsvNDo3BIIE
0IGteUDtsYMXvjVDpQmpoMNfM+YAikHvNuAuyGjRTo5l+xsTO7DwfvozYaiRsoYr
DlMqBdUWMdpy25l4VmGCpamudrp0o/X1o2HZgMDqo6msEGTyh5sCSDBOeHILXUNf
JpkMH4RM4pmwzP/bviyk0T+vN9dftx4PEdD0rPxCBsoVJZeIvNpJW1IZOCp4dov6
2mzzxdmBmAPUmEENgccUWI6ZA4vRQxMGRJNuYL0UM1WkjrocWXEP0uPykc3Ba8Vk
YuxT2zyyVoUBHjGBqbi5xkyizBV9xkMQI/ubxMrf2kPQhkWxZIMLtdOxbCT3oXm9
KdiZb3Ygnjp88Rk8/Nm6O3tkzBypDxAHv3AXWQcQzHD44uX6WaydLgHhFZbaliRj
C+eUFumuFoSLGUonA+Mh37/vA95DOXr+QIKLPaEpO+4B6XlhffiXDwXSSRX3JQFy
MU3ajnG5sPq6kZo43JTtIgNRmRo4vQf4ev86H7kekinlCuYTNAte0qxXwBmVxPTc
0qKoT8evgwck7y+sN/aO8GjckW4AcwBsI1mLLb8tRLpf+bwd4EKSIolf3u0+OKYP
GwvWI62fyUQgX9uiyFONNEzdw68ZHVuRJ0/B2x/sPjXtCtrDdU+53smesQl+/+gK
xP4bq8cGJ92c8Lru1ds/1U2ucwE1JsuvymaqSDU/olliMEAVi2E1Dei5uRbmfvof
83czLm7XknUvAlT0+ezc4w6guGQDAiaotV3l5WgWbPWjfW10nWMaKKk4xRokheZ4
EUHo+RJHC5uSkqDxY22baQ1kfU2LB4XmOR1L9hJOUYqsOIozNjsJzP8tG6diqDnV
siNvThFQKPKx9TVf/shkovrm5rLlRfKCKIZIS5rd1×14UqFCJcU8M/G66GpIvBYm
/zDRb/fdzsoL8Yfz8xEj1iEBmHfd5tJxhWprY6rKKb7W6ntOUV26iU29Rq2TBKOO
X6GF55kUt4hZiABvzZ23vhzAbEaQNaE+DJ7Y5pU3xSkcRvSQqpY6YPaGxUTNYhwx
uKZSMgWIY9Oq5A/OKDa2+5QTwi2a1eX4J08atq8c0pd3ADGyLsgrWKwij4CbP6ip
hZ6u0XbK7s38afRV7EqKZbKt9v2q283hesMU7nna32+wLpQdvOElgniJhUYuDoXu
OaSWkPJtsr+5nf02v09jx/mAJbc4dROubvv1K0cDIovg4fbpRaRlE+h7g4xnTvhs
kkjb54SFBpcZaKbiSGdrkHv+2bIS0PrvL1LbtlAqpj9IFnBesUw5VUor0BCvDWeS
iuLhsj5poi2dm8zLO+sY8ROFgUTkfUwCUZf3m1SCwP7m5dI8liZmi+US1CcWKsY5
OXnNtUBIaphArWgBcBmE8RCRzWuVxkobuEVA6HM84KewYGARK8cJ+wPLuLCpJoL9
cqrLfU4YN34Kdt4F/wO9YhHgMwaAm11ZTfSdKFbFwRI89nBw8avQQfuZtW0fyTul
M3iPFqM/S93gOCUki36Msre5eWxvn7uUGEtADTUtnFTX6mDXTcfqZT9uY5Bur7Tj
tdM+uy4GyVfeDjsnZLhr8mBFc35VXyOlluV4nDtyvGeCLXgoxyg5Xx+8gHs97JRe
XgDGUzm2jLjPCZ4plo9xd+A91N22ONeYkUnPbfqahUdf
-----END ENCRYPTED PRIVATE KEY-----
```

A continuación, generamos la clave pública. A la clave pública, no le vamos a solicitar contraseña. Para ello, utilizamos el comando:

```
openssl rsa -in claveprivada.pem -pubout -out clavepublica.pem
```

```
┌──(kali㉿kali)-[~]
└─$ openssl rsa -in claveprivada.pem -pubout -out clavepublica.pem
Enter pass phrase for claveprivada.pem:
writing RSA key
```

Y ya tenemos la clave pública:

```
┌──(kali㉿kali)-[~]
└─$ cat clavepublica.pem
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAwPbD/IlKIDQtGZiDUgb4
Q/s11WmbNE2Ub6xJRFPNj4BusXkWkNyLIcIsHfWTGL1ea5E0DIDc2SsuSK6Uce3b
DDCZz23PMNeO3hIl5rB/GnaGczwu7ySGJbD8y6PUM4RVGYrMmLggYNj4e8sdHS2J
ZILNtxfvYS2OgF4xXgGdA99bwQdUQA9EQl/J4JdfipheU5×11HFcEXVxQLCJpOYG
hNnERNNEVO2JwQNZBGNlGV/SfZuK9ync69mVqsm4fpw741/gECbzohkEONW3D2Sl
lp13Cj1vs4762mgORenxcL27j7EmouDYywVy0JWReGfeVp2cCARnL1AcEb9+Wi2z
2QIDAQAB
-----END PUBLIC KEY-----
```

Cifrado con clave pública y descifrado con clave privada

El texto debe ser menor que en el cifrado simétrico, ya que el cifrado con clave pública y privada no permite ficheros muy grandes:

```
┌──(kali㉿kali)-[~]
└─$ cat texto.txt
En un lugar de la Mancha, de cuyo nombre no quiero acordarme, no ha mucho tiempo que vivia un hid
algo de los de lanza en astillero, adarga antigua, rocin flaco y galgo corredor.
```

En primer lugar, vamos a cifrar el documento de texto plano con la clave pública:

```
openssl pkeyutl -encrypt -in texto.txt -inkey clavepublica.pem -pubin -out
texto.cifrado
```

```
┌──(kali㉿kali)-[~]
└─$ openssl pkeyutl -encrypt -in texto.txt -inkey clavepublica.pem -pubin -out texto.cifrado

┌──(kali㉿kali)-[~]
└─$ cat texto.cifrado
◆◆x◆◆m◆CR◆(◆w◆◆k◆◆g◆ ◆[◆◆H◆Ω-◆◆[]Pw◆
◆x◆Y◆7◆Wp
        ◆O!◆◆◆◆◆2rCЦ◆◆*◆◆◆◆vx◆F◆Vɔ_◆%7iA`◆bN~◆Vz◆'◆B蹭◆◆$~pd◆9◆◆◆*k◆EHA◆◆◆!◆ksx◆◆◆◆I◆[&9◆/◆R/◆'}
z\<◆6C◆M▒on◆q◆◆▒+◆◆~,◆◆◆)◆◆o◆iik◆◆-◆o◆&◆_v◆QW◆◆C=;◆y|Z
```

Ahora, vamos a descifrar el documento de texto plano con la clave privada:

```
openssl pkeyutl -decrypt -inkey claveprivada.pem -in texto.cifrado -out
texto.descifrado
```

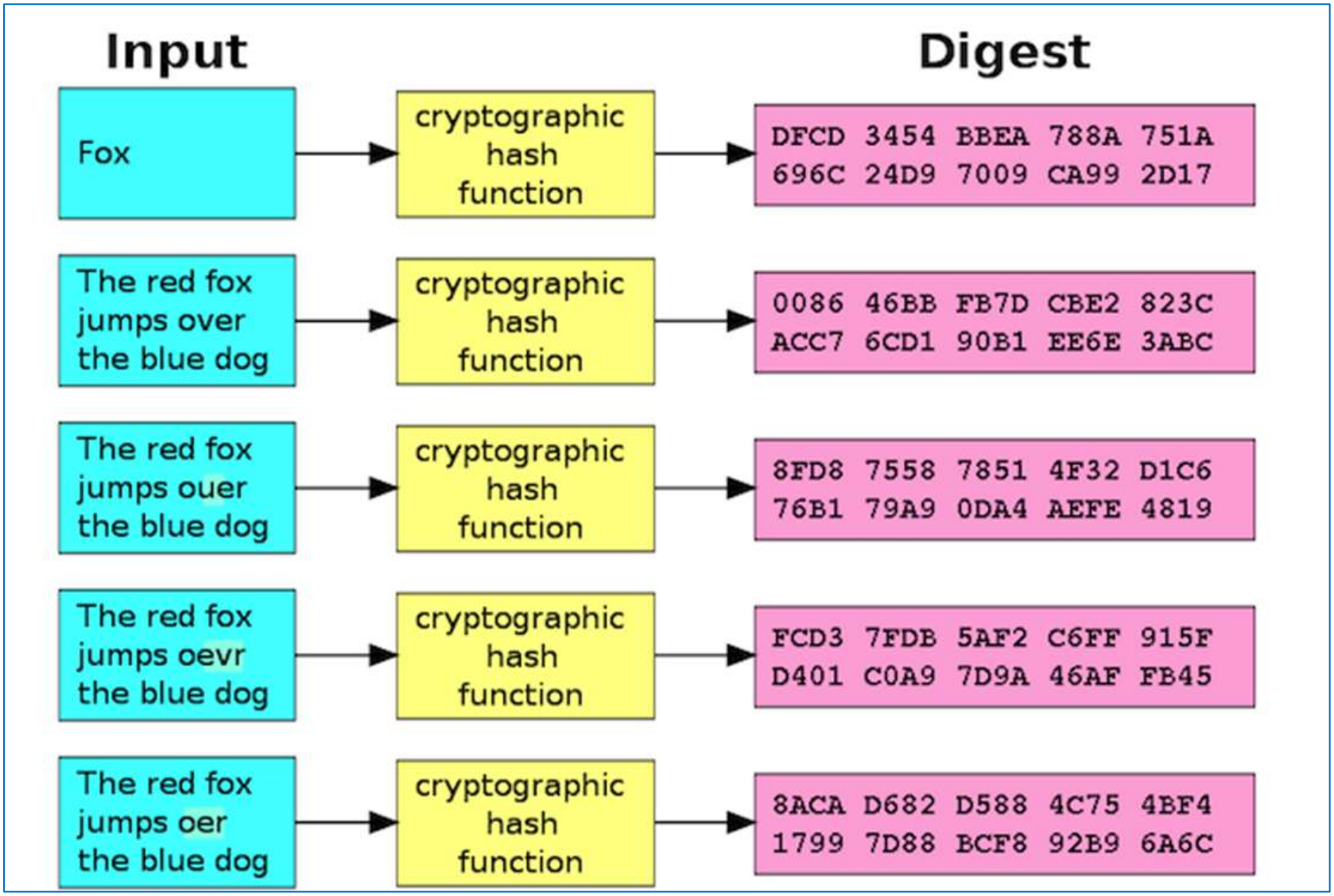
Nos va a solicitar la contraseña para utilizar la clave privada:

```
┌──(kali㉿kali)-[~]
└─$ openssl pkeyutl -decrypt -inkey claveprivada.pem -in texto.cifrado -out texto.descifrado
Enter pass phrase for claveprivada.pem:

┌──(kali㉿kali)-[~]
└─$ cat texto.descifrado
En un lugar de la Mancha, de cuyo nombre no quiero acordarme, no ha mucho tiempo que vivia un hid
algo de los de lanza en astillero, adarga antigua, rocin flaco y galgo corredor.

┌──(kali㉿kali)-[~]
└─$ diff texto.txt texto.descifrado
```

FUNCIÓN HASH:



Vamos a trabajar con los documentos de texto plano:

- texto.txt
- don_quijote_de_la-mancha.txt

El primero es un documento pequeño, de varias líneas, y el segundo es un fichero grande que contiene miles de líneas:

Vamos a generar un hash utilizando la función sha-256:

```
openssl dgst -sha256 texto.txt

openssl dgst -sha256 don_quijote_de_la_mancha.txt
```

```
┌──(kali㉿kali)-[~]
└─$ openssl dgst -sha256 texto.txt
SHA2-256(texto.txt)= 2e32c3247cc388545bbe8cbd822df0769797ad5f57d2d1149d2f451a0da9b071

┌──(kali㉿kali)-[~]
└─$ openssl dgst -sha256 don_quijote_de_la_mancha.txt
SHA2-256(don_quijote_de_la_mancha.txt)= a81ab846875269cf71f85e7cf03292e7a41b6ad9ebb2828890e9598263da9c37
```

Como se puede observar, el tamaño del hash es el mismo.

También, se puede generar una **función HMAC** (con contraseña):

```
openssl dgst -hmac “1E4Us-23TY56” texto.txt
```

```
┌──(kali㉿kali)-[~]
└─$ openssl dgst -hmac "1E4Us-23TY56" texto.txt
HMAC-SHA256(texto.txt)= 1e999fc9c0b0acdab9e95904195d5e14164c50966628882bb81e1f1dfd4ca374
```

El algoritmo de Hash que utiliza es sha-256. De esta manera, con una contraseña previamente convenida con el receptor se puede calcular el Hash. Esto añade la **autenticación** al algoritmo.