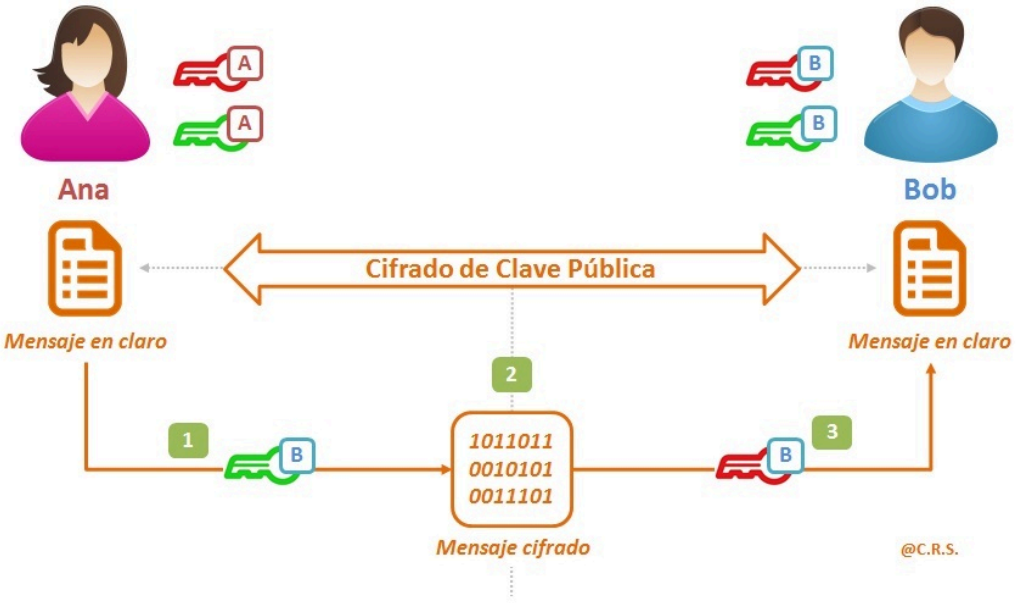
SAD - U2.4. Criptografía Asimétrica

Descargar estos apuntes

Índice

1. Criptografía Asimétrica



Los algoritmos de **cifrado asimétricos, o clave pública** utilizan dos claves para proveer seguridad. Una de las claves es **privada** (que solo conoce su propietario) y la otra es **pública**, que es de conocimiento general y, de hecho, se distribuye a todo el mundo.

Dichas claves deben respetar algunas propiedades:

- Las claves son unívocas, es decir, para una clave privada existe solamente una clave pública, y para cada clave pública existe solamente una clave privada.
- No es posible obtener una clave a partir de la otra. Así, teniendo la clave pública no es posible calcular la clave privada, y viceversa.
- Todo lo que se encripte con una clave del par podrá ser desencriptado con la otra. Así, si un mensaje se encripta usando la clave privada, podrá desencriptarse usando el par público de dicha clave, y viceversa.

Las claves son juegos de números primos grandes, del orden de los miles de bits… por ejemplo, RSA, uno de los algoritmos asimétricos más representativos, trabaja con claves de 2048 o 4096 bits.

Supongamos que Ana necesita enviar un mensaje cifrado a Bob:

1. Bob, en primera instancia, enviará a Alice su `clave pública`.
2. Alice utilizará un `algoritmo asimétrico` para encriptar el mensaje plano usando la clave pública de Bob.
3. Alice enviará a Bob el mensaje cifrado.
4. Bob utilizará su `clave privada`, par de la clave pública que envió a Alice, para descifrar el mensaje.

En este caso, un atacante podrá tener el mensaje cifrado y la clave pública de Bob pero, por las propiedades que mencionamos arriba, no podrá descifrar el mensaje.

Alice entonces podrá encriptar el mensaje usando la clave pública de Bob, y enviárselo. Bob podrá descifrar dicho mensaje usando su clave privada.

> ⚠ **Computación cuántica y criptografía asimétrica**
>
> Los algoritmos asimétricos, por sus características intrínsecas y a diferencia de los algoritmos simétricos, son vulnerables al criptoanálisis cuántico, es decir, al criptoanálisis realizado desde computadoras cuánticas.
>
> Por ello, se están investigando algoritmos cuánticos resistentes a este tipo de ataques, como el algoritmo de Shor, que puede factorizar números enteros en tiempo polinómico, lo que haría inseguros algoritmos como RSA.

A continuación podemos ver algunas ventajas e inconvenientes de la criptografía asimétrica

Ventajas	Inconvenientes
La clave pública se distribuye libremente, por lo que ya no existe el problema del intercambio de la clave que había en los métodos simétricos.	Requieren mayor tiempo de proceso que el cifrado simétrico, por lo que no sirven para grandes volúmenes de datos.
Solo es necesario un par de claves por interlocutor, con independencia del número de estos, por lo que el espacio de claves es más manejable cuando los interlocutores son muchos.	Dan lugar a mensajes cifrados de mayor tamaño que los originales.
	Para garantizar la seguridad, requieren claves de mayor tamaño que en el caso de los métodos simétricos.
	Puesto que las claves públicas se distribuyen libremente, hace falta un esquema de confianza que garantice la autenticidad de las claves públicas (que la clave pública sea de quien dice que es, que no ha sido comprometida, etc.).

1.1. Seguridad con criptografía asimétrica

Ya sabemos a grandes rasgos cómo funciona un algoritmo asimétrico: clave pública, clave privada, cifrado con una, descifrado con otra. Veamos ahora cómo se pueden lograr los pilares de la seguridad utilizando estos mecanismos.

> **Seguridad criptográfica**
>
> Recordemos Los pilares que logramos mediante el uso de la criptografía son la **confidencialidad**, la **integridad**, la **autenticidad** y el **no repudio**.

Una forma de lograr dicha seguridad es utilizando `cifrado simétrico` y `funciones Hash` de forma conjunta.

Sin embargo, la criptografía asimétrica nos permite lograr estos pilares de forma más sencilla y eficiente.

- **Confidencialidad**: Se logra `cifrando el mensaje con la clave pública del destinatario`. De esta forma, solo el destinatario podrá descifrar el mensaje con su clave privada. O bien realizando el proceso inverso, `cifrando el mensaje con la clave privada del emisor` y descifrando con la clave pública del emisor.
- **Integridad**: Se logra `calculando un hash del mensaje y cifrando dicho hash con la clave privada del emisor`. El destinatario, al recibir el mensaje, calculará el hash del mensaje y lo comparará con el hash descifrado. Si ambos coinciden, el mensaje no ha sido alterado.
- **Autenticidad**: Se logra `cifrando el mensaje con la clave privada del emisor`. El destinatario, al recibir el mensaje, podrá descifrarlo con la clave pública del emisor. Si el mensaje se descifra correctamente, el destinatario sabrá que el mensaje proviene del emisor. Esto es lo que conocemos como `firma digital`.
- **No repudio**: Se obtiene implícitamente con la `autenticidad`. Si el mensaje se descifra correctamente con la clave pública del emisor, este no podrá negar que ha enviado el mensaje.

1.2. Algoritmos de cifrado asimétrico

Existen varios algoritmos de cifrado asimétrico, aunque los más conocidos son **RSA, DSA y ECC**.

- **RSA** (Rivest, Shamir y Adleman) es uno de los algoritmos de cifrado asimétrico más utilizados. Fue creado en 1977. Se basa en la `factorización de números enteros grandes`. La seguridad de RSA se basa en la dificultad de factorizar números enteros grandes en sus factores primos. RSA es un algoritmo lento, por lo que se utiliza generalmente para cifrar claves simétricas.
- **DSA** (Digital Signature Algorithm) es un algoritmo de firma digital basado en el `problema del logaritmo discreto`. DSA se utiliza para firmar mensajes y garantizar la autenticidad e integridad de los mismos. DSA es más rápido que RSA, pero menos versátil. DSA se utiliza en aplicaciones donde la autenticidad y la integridad son críticas, como en la autenticación de usuarios y la firma de documentos. **DSA no se utiliza para cifrar mensajes, ya que no proporciona confidencialidad**.
- **ECC** (Elliptic Curve Cryptography) es un algoritmo de cifrado asimétrico basado en la `teoría de curvas elípticas`. ECC es más rápido y eficiente que RSA y DSA, y proporciona el mismo nivel de seguridad con claves más cortas. ECC se utiliza en aplicaciones donde el rendimiento y la eficiencia son críticos, como en dispositivos móviles y sistemas embebidos.
- **Diffie-Hellman** es un algoritmo de intercambio de claves que permite a dos partes acordar una clave de forma segura sobre un canal inseguro. Diffie-Hellman se basa en el `problema del logaritmo discreto` y se utiliza en aplicaciones donde se requiere un intercambio seguro de claves, como en la negociación de claves SSL/TLS.

2. Criptografía asimétrica con OpenSSL

A diferencia de la criptografía simétrica, para la criptografía asimétrica OpenSSL nos proporciona una serie de comandos para generar claves, cifrar y descifrar mensajes, firmar y verificar mensajes, etc.

Los comandos más utilizados son:

- `genpkey` : Genera un par de claves pública y privada.
- `pkeyutl` : Utiliza una clave pública o privada para cifrar o descifrar un mensaje.
- `pkey` : Se utiliza para la manipulación tanto de claves públicas como privadas.

Hay otros comandos más específicos que realizan las mismas funciones, pero **están obsoletos y se recomienda no utilizarlos**:

- `genrsa` : Genera un par de claves RSA.
- `rsa` : Se utiliza para la manipulación tanto de claves públicas como privadas.
- `rsautl` : Utiliza una clave RSA para cifrar o descifrar un mensaje.
- `gendsa` : Genera un par de claves DSA.
- `dsa` : Utiliza una clave DSA para firmar o verificar un mensaje.

2.1. Uso de OpenSSL para crear pares de claves

Para generar pares de claves se hará uso del comando `genpkey` . Se utiliza tanto para generar una clave privada, a partir de la cual se obtiene la clave pública.

2.1.1. RSA

Para generar un par de claves RSA, se utiliza alguno de los siguientes comandos:

```
$ openssl genpkey -algorithm RSA -out privateRSA.pem
$ openssl genpkey -algorithm RSA -outform DER -out privateRSA.pem
$ openssl genpkey -algorithm RSA -aes256 -out privateRSA.pem -pkeyopt rsa_keygen_bits:2048
```

En este caso, el comando `genpkey` genera un par de claves RSA y lo guarda en el archivo `private.pem` . Por defecto, se generan claves de 2048 bits.

> ⚠ **Claves protegidas con contraseña**
>
> En el caso de que se desee proteger la clave privada con una contraseña, se puede añadir la opción `-aes256` o cualquier otro algoritmo de cifrado simétrico al comando. De esta forma, se cifrará la clave privada con AES-256 y se pedirá una contraseña cada vez que se quiera acceder o utilizar a la clave privada.
>
> Por ejemplo, para proteger la clave privada con AES-256, se añade la opción `-aes256` al comando:

```
$ openssl genpkey -aes256 -algorithm RSA -out privateRSA.pem
.+............+.....+.......+...+...+...+...........+....+.........+......+..+..................+...+...+.....
+......+..+.+..+.......+...+........+....+...........+....+..+.+.....++++++++++++++++++++++++++++++++++++++++*..
++++++++++++++++++++++++++++++++++++++*..+...+...+.....+..........+..+.++++++
.........+...+.......+...+...+..+.......+..+..........+...+.....+......+.+...+.....+...
++++++++++++++++++++++++++++++++++++++*.....+......+..+...+....+++++++++++++++++++++++++++++++++++++++*.......
+...........+.........+......+.+.....+....+........+..........+...........+......+....+.........+..+......+....
+...+........+.+..............+.....++++++
Enter PEM pass phrase:

Verifying - Enter PEM pass phrase:
```

Para ver la clave privada generada, se puede utilizar el comando `cat` (Linux) o `type` (Windows):

```
$ cat private.pem
-----BEGIN ENCRYPTED PRIVATE KEY-----
MIIFLTBXBgkqhkiG9w0BBQ0wSjApBgkqhkiG9w0BBQwwHAQIzgqJ6uIcHCUCAggA
MAwGCCqGSIb3DQIJBQAwHQYJYIZIAWUDBAEqBBAqkESJ+5bfHBdYvwTZFGW3BIIE
0LskJiiJAihpTJCsEOPKMqTh6r+XyNQTAoVtkKQ31d63dbe2vpIP7RfCNYNCcDRQ
Vyqpt/YN83ltkQ46tuh68NzyT2oUa/LS5I34xj9ABI7CLD2XqQWZ6kTPakgRYaKb
WwEV0WhXCFLkiX+kAHWX+LcfR9s/DyCRpNzxwBjJAvysolhWx0gBjc0R5aEOtQqt
E5ErEcsYX+iPJUMIEVtrKy3PIB9doqGlSG45ZFC8NSu8tDizrBvVicbnT/2jsJb5
IThcfIEQZp9X31tRN/OmnxARhNtRn2YW+9/dTBOlLXPJ/Ikdm0kx6JHMbfw/hH8S
+3HP7p5W5wGe6xwhAb1soD3ZPXDpu76zTLmv4IoXu2kuamNNNGUqpDDG3OT54QsR
dtBn8jUhR+niWB2uBGSZ902+Ru+VY0M42SjVYAb83EoAB86HyG6XAKmJRBmzXZfh
Qmk8svfC1n6+7vv5MUaqTTRsqeTwGu4jenOTEVjHUyKQamMak4F83oV/4wFsuK5V
y7VfVv3DkCy9/CJVdmS3ZHc4V5D741VLc8keNKhH17rShbNHE+lFlrLzA7YwRdw4
lpEbaoLxzjz/yB2LKjoWRbngpxcB7/qz1NkObm3S2ROWcRDi5hNHEc0ctlT0kuXi
KSG6Sqlald9bast1zmMpIpvvfKVG3moq2umVR1B2JhwYRmdKEsJ7NjioI7pmuZc4
fr485I1SDBGjhSmsQAdUq5b7aljCbgrSLjHYFVxWLOutfhE1P3E7sgiUWlQmvZWP
qLygQUFBKE9Q/WjRR3cN4WOFDIE3wVpE8uejkDy++pbGV2xb7mQG+77S3lWvtdDp
5Zu4qDl0daZjAP6W/636aBqiQbGvFM1Ch0+UNyYie9vh7zjX5K3viGiUHW1yb9+o
QQfpgIIODz7X3bk6kQBS5j+xDAx+oKAw0sxlTGqp5hX6nPP9/LI7XeCeWQdAV/ls
G5q8UjVDKyKjBQUWftnRDpoUnCOSTschFifktDInHrSaVetaWn5yFs/cx+DAE2xa
wRJn6k8gXhk72ytTqd64BLw3evZMPnaldEhMGle44ZYf7cebL8RhKDEgoT99DLKU
1TIiOzsMz+nzkI0t8U6nsF+pEpWuOjPP4lPSOqhAbwGMIdTgvtWSmLBJzdQ0uP85
Vo6Gez8mgjcFh39VGuq2WmkY6hkmiodpv2IjFBUAkC4vrGh1E/NQGIBWj/5HwDiy
iA+gw/mQts2eDQw00UbpCAXDuxIsmLwc5HI/gmiXxx82H1ZtGxljrut3vMotukKf
4gr+T4umvvEl/3/uGGEMT/WQySoso/47JS9tPiogA6D+GFJ2AZA1XsPuvuZ/WPWI
RbUlgOrVbq4a+7LBC/tzC28hSDF+VRdAT3FoiS2HXXFJYrCMCjmTFz3YTnKc6U/k
k1rfuoMLnFMimy+tR14zMuH6Dxu+SDFj862cZeMrQOAaplnGpxDiO37u8CKXaQJ7
JpiK5n2mrkSZ6RHM4m5VZeSJNSTtxJbs/3kaJZQLJ1e2wufD3Rzy05VKdQSyFjU9
J+D3spLNvz72gAvZJGLdP5BT/uIfAI40vvNx5csei9kqTQ2QLoL5bpvmV+d4uz6D
1RNVIYJ4cAkg5AjRpP6kGWHGCx1GTc0MD34mCwNNW/ll
-----END ENCRYPTED PRIVATE KEY-----
```

Para obtener la clave pública a partir de la clave privada, se utiliza el siguiente comando:

```
$ openssl pkey -in privateRSA.pem -out publicRSA.pem -pubout
$ cat publicRSA.pem
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAvXck+2msWq8LEhX/AwJB
7hIZvW+QwpODVU5RkfFEk+8L2IPK+BWM6gMZO6HfQqn8yjkjAHTEY/RlBSGaCX1V
AA5V3RQs4SBbcQAFo0tBpqhTUw36Se9sozaxnN4kObTZxvFmzX3BqtKzbZP33/og
RgKELBpe43GTDkhQtfWpCzGQRiGodFBLBPQ7HibdKB+RK/3wTftUOgriabb5HSdI
9ySyrkvykEheRHgFe6XUo6gU3BWA1h276wXqdw2TiJ+VR/GX8tzXGV1RSthxqNpW
2/EpGOV12vRz9MnTEdt2Rfmx9x3VzMrxjiXSK97xpWXlqmoRCj+cdb1xvD05vVcG
MwIDAQAB
-----END PUBLIC KEY-----
```

En este caso, el comando `pkey` extrae la clave pública del archivo `private.pem` y la guarda en el archivo `public.pem`.

2.1.2. DSA

Para generar un par de claves DSA, hay que realizar algunos pasos adicionales

Se genera un fichero que contendrá los parámetros DSA en dsa.params.pem usando `-genparam`. Además se añade `-pkeyopt` para especificar el tamaño en bits (2048) de la clave.

```
openssl genpkey -genparam -algorithm DSA -out dsa.params.pem -pkeyopt dsa_paramgen_bits:2048
```

Y a continuación, se genera la clave DSA

```
$ openssl genpkey -paramfile dsa.params.pem -out privateDSA.pem
$ openssl genpkey -paramfile dsa.params.pem -outform DER -out privateDSA.pem
$ openssl genpkey -aes256 -paramfile dsa.params.pem -out privateDSA.pem
```

En este caso, el comando `genpkey` genera un par de claves DSA y lo guarda en el archivo `privateDSA.pem`. Por defecto, se generan claves de 2048 bits, aunque aquí se hará lo que se haya configurado en el archivo `dsa.params.pem`.

Para ver la clave privada generada, se puede utilizar el comando `cat` (Linux) o `type` (Windows):

```
$ cat privateDSA.pem
-----BEGIN ENCRYPTED PRIVATE KEY-----
MIIFLTBXBgkqhkiG9w0BBQ0wSjApBgkqhkiG9w0BBQwwHAQIzgqJ6uIcHCUCAggA
MAwGCCqGSIb3DQIJBQAwHQYJYIZIAWUDBAEqBBAqkESJ+5bfHBdYvwTZFGW3BIIE
0LskJiiJAihpTJCsEOPKMqTh6r+XyNQTAoVtkKQ31d63dbe2vpIP7RfCNYNCcDRQ
Vyqpt/YN83ltkQ46tuh68NzyT2oUa/LS5I34xj9ABI7CLD2XqQWZ6kTPakgRYaKb
WwEV0WhXCFLkiX+kAHWX+LcfR9s/DyCRpNzxwBjJAvysolhWx0gBjc0R5aEOtQqt
E5ErEcsYX+iPJUMIEVtrKy3PIB9doqGlSG45ZFC8NSu8tDizrBvVicbnT/2jsJb5
IThcfIEQZp9X31tRN/OmnxARhNtRn2YW+9/dTBOlLXPJ/Ikdm0kx6JHMbfw/hH8S
+3HP7p5W5wGe6xwhAb1soD3ZPXDpu76zTLmv4IoXu2kuamNNNGUqpDDG3OT54QsR
dtBn8jUhR+niWB2uBGSZ902+Ru+VY0M42SjVYAb83EoAB86HyG6XAKmJRBmzXZfh
Qmk8svfC1n6+7vv5MUaqTTRsqeTwGu4jenOTEVjHUyKQamMak4F83oV/4wFsuK5V
y7VfVv3DkCy9/CJVdmS3ZHc4V5D741VLc8keNKhH17rShbNHE+lFlrLzA7YwRdw4
lpEbaoLxzjz/yB2LKjoWRbngpxcB7/qz1NkObm3S2ROWcRDi5hNHEc0ctlT0kuXi
KSG6Sqlald9bast1zmMpIpvvfKVG3moq2umVR1B2JhwYRmdKEsJ7NjioI7pmuZc4
fr485I1SDBGjhSmsQAdUq5b7aljCbgrSLjHYFVxWLOutfhE1P3E7sgiUWlQmvZWP
qLygQUFBKE9Q/WjRR3cN4WOFDIE3wVpE8uejkDy++pbGV2xb7mQG+77S3lWvtdDp
5Zu4qDl0daZjAP6W/636aBqiQbGvFM1Ch0+UNyYie9vh7zjX5K3viGiUHW1yb9+o
QQfpgIIODz7X3bk6kQBS5j+xDAx+oKAw0sxlTGqp5hX6nPP9/LI7XeCeWQdAV/ls
G5q8UjVDKyKjBQUWftnRDpoUnCOSTschFifktDInHrSaVetaWn5yFs/cx+DAE2xa
wRJn6k8gXhk72ytTqd64BLw3evZMPnaldEhMGle44ZYf7cebL8RhKDEgoT99DLKU
1TIiOzsMz+nzkI0t8U6nsF+pEpWuOjPP4lPSOqhAbwGMIdTgvtWSmLBJzdQ0uP85
Vo6Gez8mgjcFh39VGuq2WmkY6hkmiodpv2IjFBUAkC4vrGh1E/NQGIBWj/5HwDiy
iA+gw/mQts2eDQw00UbpCAXDuxIsmLwc5HI/gmiXxx82H1ZtGxljrut3vMotukKf
4gr+T4umvvEl/3/uGGEMT/WQySoso/47JS9tPiogA6D+GFJ2AZA1XsPuvuZ/WPWI
RbUlgOrVbq4a+7LBC/tzC28hSDF+VRdAT3FoiS2HXXFJYrCMCjmTFz3YTnKc6U/k
k1rfuoMLnFMimy+tR14zMuH6Dxu+SDFj862cZeMrQOAaplnGpxDiO37u8CKXaQJ7
JpiK5n2mrkSZ6RHM4m5VZeSJNSTtxJbs/3kaJZQLJ1e2wufD3Rzy05VKdQSyFjU9
J+D3spLNvz72gAvZJGLdP5BT/uIfAI40vvNx5csei9kqTQ2QLoL5bpvmV+d4uz6D
1RNVIYJ4cAkg5AjRpP6kGWHGCx1GTc0MD34mCwNNW/ll
-----END ENCRIPTED PRIVATE KEY-----
```

Para obtener la clave pública a partir de la clave privada, se utiliza el siguiente comando:

```
$ openssl pkey -in privateDSA.pem -pubout -out publicDSA.pem
$ cat publicDSA.pem
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAvXck+2msWq8LEhX/AwJB
7hIZvW+QwpODVU5RkfFEk+8L2IPK+BWM6gMZO6HfQqn8yjkjAHTEY/RlBSGaCX1V
AA5V3RQs4SBbcQAFo0tBpqhTUw36Se9sozaxnN4kObTZxvFmzX3BqtKzbZP33/og
RgKELBpe43GTDkhQtfWpCzGQRiGodFBLBPQ7HibdKB+RK/3wTftUOgriabb5HSdI
9ySyrkvykEheRHgFe6XUo6gU3BWA1h276wXqdw2TiJ+VR/GX8tzXGV1RSthxqNpW
2/EpGOV12vRz9MnTEdt2Rfmx9x3VzMrxjiXSK97xpWXlqmoRCj+cdb1xvD05vVcG
MwIDAQAB
-----END PUBLIC KEY-----
```

En este caso, el comando `pkey` extrae la clave pública del archivo `privateDSA.pem` y la guarda en el archivo `publicDSA.pem`.

2.2. Uso de OpenSSL para cifrar y firmar mensajes

Una vez que disponemos de las claves, podemos cifrar y descifrar mensajes, firmar y verificar mensajes, etc.

2.2.1. RSA para cifrar y descifrar mensajes

Suponiendo que tenemos la clave pública de una persona, podemos cifrar un mensaje con su clave pública y enviarlo.Sólo esa persona podrá después descifrar el mensaje con su clave privada.

Cifrado con clave asimétrica

El cifrado asimétrico sólo se puede aplicar a "cosas pequeñas" como claves simétricas o resúmenes.
Cualquier intento de cifrar con RSA un fichero normal, de más de 200 caracteres, producirá un error de tipo "input too long".

> Public Key operation error 346F0000:error:0200006E:rsa routines:ossl_rsa_padding_add_PKCS1_type_2_ex:**data too large for key size**:crypto\rsa\rsa_pk1.c:133:

Para cifrar un mensaje con la clave pública, se utiliza el comando `pkeyutl` con la opción `-encrypt`:

```
$ echo "Hola, mundo" > mensaje.txt
$ openssl pkeyutl -encrypt -inkey publicRSA.pem -pubin -in mensaje.txt -out mensaje.txt.rsa
```

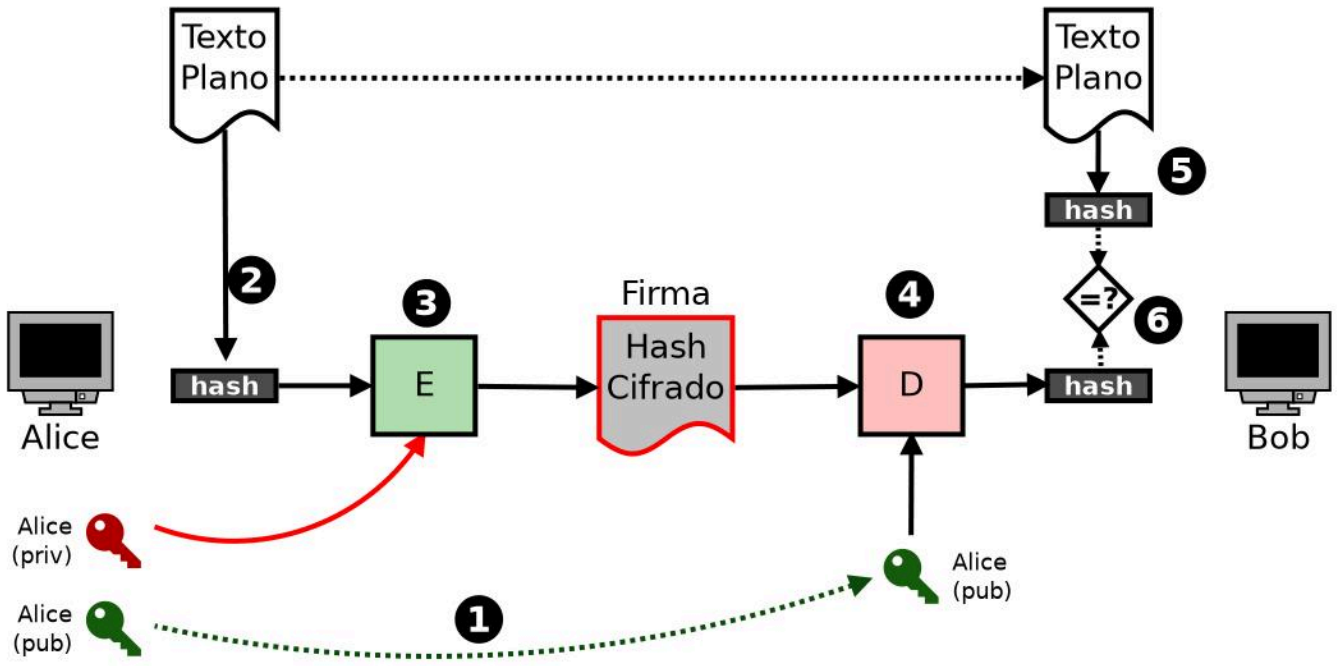
En este caso, el comando `pkeyutl` cifra `-encrypt` el archivo `mensaje.txt` utilizando la clave pública `-inkey -pubkey` y guarda el resultado en `mensaje.txt.rsa`.

Para descifrar el mensaje cifrado, se utiliza el comando `pkeyutl` con la opción `-decrypt`:

```
$ openssl pkeyutl -decrypt -inkey privateRSA.pem -in mensaje.txt.rsa
Hola, mundo
```

En este caso, el comando `pkeyutl` descifra `-decrypt` el archivo `mensaje.txt.rsa` utilizando la clave privada `-inkey` y muestra el resultado por pantalla.

2.2.2. RSA para firmar y verificar mensajes



El proceso inverso, cifrar con la clave privada y descifrar con la clave pública, se utiliza para **firmar mensajes**. Para firmar un mensaje con la clave privada, se utiliza el comando `pkeyutl` con la opción `-sign`:

```
$ openssl pkeyutl -sign -inkey privateRSA.pem -in mensaje.txt -out mensaje.txt.sig
```

En este caso, el comando `pkeyutl` firma `-sign` el archivo `mensaje.txt` utilizando la clave privada `-inkey` y guarda el resultado en `mensaje.txt.sig`.

Para verificar la firma del mensaje, el destinatario, usando la clave pública utiliza el comando `pkeyutl` con la opción `-verify`:

```
$ openssl pkeyutl -verify -inkey publicRSA.pem -pubin -sigfile mensaje.txt.sig -in mensaje.txt
Signature Verified Successfully
```

En este caso, el comando `pkeyutl` verifica `-verify` la firma del archivo `mensaje.txt` utilizando la clave pública `-inkey -pubin` y el archivo de firma `mensaje.txt.sig`.

> **Integridad del mensaje**
>
> ¿Qué sucede si modificamos el mensaje `mensaje.txt` y volvemos a verificar la firma?
>
> ¿Qué sucede si alguien intercepta el mensaje, lo modifica y lo firma con otra clave pública?

2.2.3. DSA para firmar y verificar mensajes

> **DSA no se utiliza para cifrar mensajes**

DSA no se utiliza para cifrar mensajes, ya que no proporciona confidencialidad. DSA se utiliza para firmar mensajes y garantizar la autenticidad e integridad de los mismos.

En primer lugar, ya hemos comentado que el cifrado asimétrico no se utiliza para cifrar mensajes, principalmente porque no permite cifrar mensajes de gran tamaño.

Además, si queremos enviar un mensaje firmado por nosotros para que se pueda comprobar la autenticidad del mensaje, lo lógico es que dicho mensaje se envíe en claro.

Una vez que disponemos de un par de claves DSA, podemos firmar y verificar mensajes de forma similar a como lo hemos hecho anteriormente, aunque en este caso, no vamos a cifrar el mensaje, sino su resumen.

Realizamos el resumen del fichero "texto.txt" y lo guardamos en formato binario en el fichero de salida "resumen.sha".

```
$ openssl dgst -sha512 -binary -out mensaje.sha mensaje.txt
```

A continuación, firmamos el resumen del mensaje con la clave privada DSA:

```
$ openssl pkeyutl -sign -inkey privateDSA.pem -in mensaje.sha -out mensaje.sha.sig
```

Al destinatario le llegan los ficheros **mensaje.txt y mensaje.sha.sig**. Para verificar la firma del mensaje, el destinatario, usando la clave pública, verifica la firma del resumen del mensaje:

```
$ openssl dgst -sha512 -binary -out mensaje.sha mensaje.txt
$ openssl pkeyutl -verify -inkey publicDSA.pem -pubin -sigfile mensaje.sha.sig -in mensaje.sha
Signature Verified Successfully
```

En este caso, el comando `pkeyutl` verifica `-verify` el resumen del archivo `mensaje.txt` `mensaje.sha` utilizando la clave pública `-inkey -pubin` y el archivo de firma `mensaje.sha.sig` recibido.

Actividades

1. Genera un par de claves DSA de 2048 bits y guárdalas en los archivos `privateDSA_tunombre.pem` y `publicDSA_tunombre.pem` y realiza la siguientes operaciones:
   i. Cifra un documento cualquiera (texto, imagen, etc.) con la clave pública DSA y guárdalo en el archivo `documento.xxx.dsa`.
      - **¿Qué ha pasado? ¿Por qué pasa esto? ¿Cómo podemos solucionar este problema?**
   ii. Firma el documento (sin cifrar) con la clave privada DSA y guárdalo en el archivo `documento.xxx.sig`.
   iii. Intercambia las claves públicas, el documento cifrado y el archivo de firma con alguno de tus compañeros por correo electrónico **con copia a tu profesor**.
2. Descarga los archivos recibidos y realiza las siguientes operaciones:
   i. Descifra/Verifica el archivo recibido con la clave pública del compañero.
   ii. Verifica que la firma del documento es correcta.
3. Modifica el documento recibido y vuelve a comprobar la firma
   i. ¿Qué sucede?
4. ¿Se puede hacer una interceptación y modificación del mensaje en algún punto de la cadena?
   i. Intenta modificar el documento y vuelve a firmarlo con tu clave privada.
   ii. Reenvíalo como si fuera el documento original (firmado por otra persona)

iii. ¿Qué sucede?

5. ¿Qué puedes asegurar, a nivel de seguridad, de la información recibida en el correo electrónico?