

Criptografía-Asimetrica.pdf



jmp_0807



Seguridad de la Información



3º Grado en Ingeniería Informática



Escuela Técnica Superior de Ingeniería Informática
Universidad de Málaga



Que no te escriban poemas de amor
cuando terminen la carrera ►►►►►►►

☺
(a nosotros por
suerte nos pasa)

WUOLAH

**Que no te escriban poemas de amor
cuando terminen la carrera** ►►►►►



WUOLAH

(a nosotros por suerte nos pasa)

No si antes decirte
Lo mucho que te voy a recordar

Pero me voy a graduar.
Mañana mi diploma y título he de
pagar

Llegó mi momento de despedirte
Tras años en los que has estado mi
lado.

Siempre me has ayudado
Cuando por exámenes me he
agobiado

Oh Wuolah wuolah
Tu que eres tan bonita

File - /home/javimp2003/IdeaProjects/UMAPracs/Criptografia Asimetrica/ca.py

```
1 import funciones_rsa as f
2
3 keyAuxAlice = f.crear_RSAKey()
4 keyAuxBob = f.crear_RSAKey()
5
6 f.guardar_RSAKey_Publica("publicaAlice.txt",keyAuxAlice)
7 f.guardar_RSAKey_Publica("publicaBob.txt",keyAuxBob)
8 f.guardar_RSAKey_Privada("parClavesAlice.txt",keyAuxAlice,"claveAlice")
9 f.guardar_RSAKey_Privada("parClavesBob.txt",keyAuxBob,"claveBob")
```

WUOLAH

```

1 import funciones_rsa as r
2 import funciones_aes as a
3 import socket_class as socket
4
5 privadaBob = r.cargar_RSAKey_Privada("parClavesBob.txt","claveBob")
6 publicaAlice = r.cargar_RSAKey_Publica("publicaAlice.txt")
7
8 socketServer = socket.SOCKET_SIMPLE_TCP('127.0.0.1',5551)
9 socketServer.escuchar()
10 array_bytes = socketServer.recibir()
11 array_bytes_firma = socketServer.recibir()
12 mensajeDescifradoK1 = r.descifrarRSA_OAEP(array_bytes,privadaBob)
13 print(mensajeDescifradoK1)
14 validezBoolean = r.comprobarRSA_PSS(mensajeDescifradoK1,array_bytes_firma,
    publicaAlice)
15 print(validezBoolean)
16
17 cadena = "Hola Alice"
18 (aes_cif, nonce) = a.iniciarAES_CTR_cifrado(mensajeDescifradoK1)
19 cadenaCifrada = a.cifrarAES_CTR(aes_cif, cadena.encode("utf-8"))
20 signatureCadena = r.firmarRSA_PSS(cadena.encode("utf-8"),privadaBob)
21 # Una vez aqui ya tenemos Bob cifrará la cadena "Hola Alice" utilizando AESCTR-128
# con la clave K1, y firmará esa cadena con su clave privada
22
23 socketServer.enviar(nonce)
24 socketServer.enviar(cadenaCifrada)
25 socketServer.enviar(signatureCadena)
26
27 nonceAlice = socketServer.recibir()
28 cadenaCifAlice = socketServer.recibir()
29 hashFirmaAlice = socketServer.recibir()
30 aes_desc_A = a.iniciarAES_CTR_descifrado(mensajeDescifradoK1,nonceAlice)
31 datosClaroAlice = a.descifrarAES_CTR(aes_desc_A,cadenaCifAlice)
32 booleanFirmaAlice = r.comprobarRSA_PSS(datosClaroAlice,hashFirmaAlice,publicaAlice)
33 print(f"Cadena de caracteres recibida: {datosClaroAlice} Validez: {booleanFirmaAlice}")
34
35 socketServer.cerrar()

```

Que no te escriban poemas de amor cuando terminen la carrera

(a nosotros por suerte nos pasa)



Ayer a las 20:20

Oh Wuolah wuolitah
Tu que eres tan bonita

Siempre me has ayudado
Cuando por exámenes me he
agobiado

Llegó mi momento de despedirte
Tras años en los que has estado mi
lado.

Pero me voy a graduar.
Mañana mi diploma y título he de
pagar

No si antes decirte
Lo mucho que te voy a recordar



Envía un mensaje...



WUOLAH

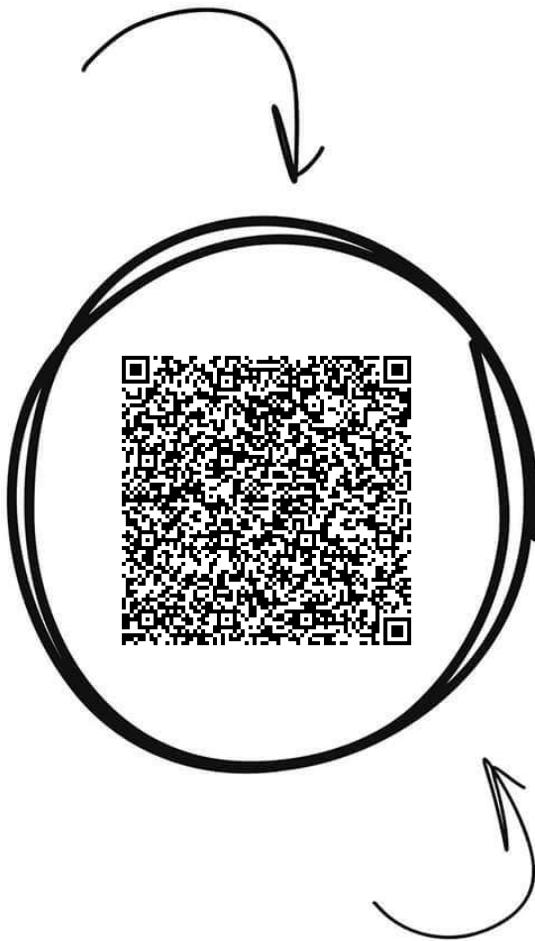


Seguridad de la Información



Comparte estos flyers en tu clase y consigue más dinero y recompensas

- 1 Imprime esta hoja
- 2 Recorta por la mitad
- 3 Coloca en un lugar visible para que tus compis puedan escanear y acceder a apuntes
- 4 Llévate dinero por cada descarga de los documentos descargados a través de tu QR



Banco de apuntes de la

WUOLAH



```
1 import funciones_rsa as r
2 import funciones_aes as a
3 import socket_class as socket
4
5 privadaAlice = r.cargar_RSAKey_Privada("parClavesAlice.txt", "claveAlice")
6 publicaBob = r.cargar_RSAKey_Publica("publicaBob.txt")
7
8 k1 = a.crear_AESKey()
9 arrayCifrado = r.cifrarRSA_OAEP(k1, publicaBob)
10 signatureFirma = r.firmarRSA_PSS(k1, privadaAlice)
11
12 socketClient = socket.SOCKET_SIMPLE_TCP('127.0.0.1', 5551)
13 socketClient.conectar()
14 socketClient.enviar(arrayCifrado)
15 socketClient.enviar(signatureFirma)
16
17 nonceBob = socketClient.recibir()
18 cadenaCif = socketClient.recibir()
19 hashFirmaBob = socketClient.recibir()
20 aes_descif = a.iniciarAES_CTR_descifrado(k1, nonceBob)
21 datosClaroBob = a.descifrarAES_CTR(aes_descif, cadenaCif)
22 booleanFirmaBob = r.comprobarRSA_PSS(datosClaroBob, hashFirmaBob, publicaBob)
23 print(f"Cadena de caracteres recibida: {datosClaroBob}. Validez: {booleanFirmaBob}")
24
25 cadenaA = "Hola Bob"
26 (aes_cif_a, nonceA) = a.iniciarAES_CTR_cifrado(k1)
27 cadenaCifA = a.cifrarAES_CTR(aes_cif_a, cadenaA.encode("utf-8"))
28 signA = r.firmarRSA_PSS(cadenaA.encode("utf-8"), privadaAlice)
29
30 socketClient.enviar(nonceA)
31 socketClient.enviar(cadenaCifA)
32 socketClient.enviar(signA)
33
34 socketClient.cerrar()
```

```
1 -----BEGIN PUBLIC KEY-----
2 MIIBIjANBgkqhkiG9w0BAQEAAQ8AMIIIBcGKCAQEaTYWfYOSx2eanpqYrmz
3 MlxeiGh5lje4pRpL+ykSVw3X1pDcl1VY0ie0tGpMI2qfc3rP5DdsoSk/07T19nDD
4 RUmKMhRf/EfJdd8GkcciZCEXfrbubVNYKc/xg7ADu3QVbRXuW4AJUKAeWuiPkfV
5 wvuab9X+HdHQMD2yYhxJPL48uJtr0F7VrkRLT3lj0jbyugpKG0vB/mdi7reoUkNl
6 eK+jkZBRzE8gFANjhTFFiJRrHp11n+7h7n30ucPSuGBmA9apkFLjoCEtWAAIKvRh
7 S9/ztt6HfVExx3s/3WvxqeZI/SeEZp+s8clZgiW4tCSeIOgpDzRlo7Xa4Xjos/7M
8 owIDAQAB
9 -----END PUBLIC KEY-----
```

Que no te escriban poemas de amor cuando terminen la carrera ➤➤➤➤➤



WUOLAH

(a nosotros por suerte nos pasa)

No si antes decíte
Lo mucho que te voy a recordar

Pero me voy a graduar.
Mañana mi diploma y título he de
pagar

Llegó mi momento de despedirme
Tras años en los que has estado mi
lado.

Siempre me has ayudado
Cuando por exámenes me he
agobiado

Oh Wuolah wuolah
Tu que eres tan bonita

File - /home/javimp2003/IdeaProjects/UMAPracs/Criptografia Asimetrica/socket_class.py

```
1 """
2 Clase de ejemplo para el envio y la recepcion de mensajes en un canal TCP
3 No utilizar en un entorno de produccion
4 """
5
6 import socket
7 import struct
8 import sys
9
10 class SOCKET_SIMPLE_TCP:
11
12     def __init__(self, host, puerto):
13         """Inicializa un objeto socket TCP, proporcionando un host y a un puerto"""
14         self.host = host
15         self.puerto = puerto
16         self.server = None
17
18     def conectar(self):
19         """Convierte el objeto socket en un cliente, y se conecta a un servidor"""
20         self.socket = socket.create_connection((self.host, self.puerto))
21
22     def escuchar(self):
23         """Convierte el objeto socket en un servidor, y recibe la peticion de un
24         cliente"""
25         self.server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
26         server_address = (self.host, self.puerto)
27         self.server.bind(server_address)
28         self.server.listen(1)
29         self.socket, dir_cliente = self.server.accept()
30         return dir_cliente
31
32     def __recvall(self, count):
33         """PRIVADO: Recibe "count" bytes del buffer de entrada"""
34         buffer = b''
35         while count:
36             # Puede que no reciba "count", asi que tengo que leer hasta recibirlo
37             todo
38             newbuf = self.socket.recv(count)
39             if not newbuf: return None
40             buffer += newbuf
41             count -= len(newbuf)
42         return buffer
43
44     def enviar(self, datos):
45         """Envia un array de bytes "datos" del origen al destino."""
46         longitud = len(datos)
47         self.socket.sendall(struct.pack('!I', longitud)) # unsigned int en formato de
48         red(!)
49         self.socket.sendall(datos)
50
51     def recibir(self):
52         """Recibe un array de bytes "datos" del destino al destino."""
53         lenbuf = self.__recvall(4)
54         longitud, = struct.unpack('!I', lenbuf)
55         return self.__recvall(longitud)
56
57     def cerrar(self):
58         """Cierra la conexion"""
59         if self.socket != None:
60             self.socket.close()
61         if self.server != None:
62             self.server.close()
```

WUOLAH

```

1 from Crypto.Random import get_random_bytes
2 from Crypto.Cipher import AES
3
4 def crear_AESKey():
5     """Devuelve un número aleatorio de 16 bytes - 128 bits"""
6     return get_random_bytes(16)
7
8 def iniciarAES_GCM_cifrado(key_16):
9     """Inicia el engine de cifrado AES CTR"""
10    nonce_16_ini = get_random_bytes(16)
11    aes_cifrado = AES.new(key_16, AES.MODE_GCM, nonce = nonce_16_ini, mac_len = 16)
12    return aes_cifrado, nonce_16_ini
13
14 def iniciarAES_GCM_descifrado(key_16, nonce_16_ini):
15     """Inicia el engine de descifrado AES CTR"""
16     aes_descifrado = AES.new(key_16, AES.MODE_GCM, nonce = nonce_16_ini, mac_len = 16)
17
18     return aes_descifrado
19
20 def cifrarAES_GCM(aes_cifrado, datos):
21     """Cifra el parámetro datos (de tipo array de bytes), y devuelve el texto cifrado
22     binario Y el mac Y el nonce """
23     # En escenarios reales, el objeto AES se inicializa una vez, y luego vamos
24     # llamando a aes_cifrado
25     datos_cifrado, mac_cifrado = aes_cifrado.encrypt_and_digest(datos)
26     return datos_cifrado, mac_cifrado
27
28 def descifrarAES_GCM(aes_descifrado, datos, mac):
29     """Descifra el parámetro datos (de tipo binario), y devuelve los datos
30     descifrados de tipo array de bytes.
31     También comprueba si el mac es correcto"""
32
33     try:
34         datos_claro = aes_descifrado.decrypt_and_verify(datos, mac)
35         return datos_claro
36     except (ValueError, KeyError) as e:
37         return False
38
39 def iniciarAES_CTR_cifrado(key_16):
40     """Inicia el engine de cifrado AES CTR"""
41     nonce_16_ini = get_random_bytes(8) # nonce aleatorio de 64 bits
42     # --64 nonce--/-64 contador--
43     ctr_16 = 0 # contador, empezando desde 0
44     aes_cifrado = AES.new(key_16, AES.MODE_CTR, nonce = nonce_16_ini, initial_value
45     = ctr_16)
46     return aes_cifrado, nonce_16_ini
47
48 def iniciarAES_CTR_descifrado(key_16, nonce_16_ini):
49     """Inicia el engine de cifrado AES CTR"""
50     ctr_16 = 0 # contador, empezando desde 0. Origen y destino
51     # DEBEN tener este mismo valor
52     # Si lees esto, piensa: ¿Que problema puede haber
53     # en que este valor sea siempre 0 en esta librería?
54     aes_descifrado = AES.new(key_16, AES.MODE_CTR, nonce = nonce_16_ini,
55     initial_value = ctr_16)
56     return aes_descifrado
57
58 def cifrarAES_CTR(aes_cifrado, datos):
59     """Cifra el parámetro datos (de tipo array de bytes), y devuelve el texto cifrado
60     binario Y el mac"""
61     # En escenarios reales, el objeto AES se inicializa una vez, y luego vamos
62     # llamando a aes_cifrado
63     datos_cifrado = aes_cifrado.encrypt(datos)
64     return datos_cifrado
65
66 def descifrarAES_CTR(aes_descifrado, datos):
67     """Descifra el parámetro datos (de tipo binario), y devuelve los datos
68     descifrados de tipo array de bytes"""
69     datos_claro = aes_descifrado.decrypt(datos)
70     return datos_claro

```

```

1  from Crypto.PublicKey import RSA
2  from Crypto.Cipher import PKCS1_OAEP
3  from Crypto.Signature import pss
4  from Crypto.Hash import SHA256
5
6  def crear_RSAKey():
7      key = RSA.generate(2048)
8
9      return key
10
11 def guardar_RSAKey_Privada(fichero, key, password):
12     key_cifrada = key.export_key(passphrase=password, pkcs=8, protection="scryptAndAES128-CBC")
13     file_out = open(fichero, "wb")
14     file_out.write(key_cifrada)
15     file_out.close()
16
17 def cargar_RSAKey_Privada(fichero, password):
18     key_cifrada = open(fichero, "rb").read()
19     key = RSA.import_key(key_cifrada, passphrase=password)
20
21     return key
22
23 def guardar_RSAKey_Publica(fichero, key):
24     key_pub = key.publickey().export_key()
25     file_out = open(fichero, "wb")
26     file_out.write(key_pub)
27     file_out.close()
28
29 def cargar_RSAKey_Publica(fichero):
30     keyFile = open(fichero, "rb").read()
31     key_pub = RSA.import_key(keyFile)
32
33     return key_pub
34
35 def cifrarRSA_OAEP(datos, key):
36     engineRSACifrado = PKCS1_OAEP.new(key)
37     cifrado = engineRSACifrado.encrypt(datos)
38
39     return cifrado
40
41 def descifrarRSA_OAEP(cifrado, key):
42     engineRSADescifrado = PKCS1_OAEP.new(key)
43     datos = engineRSADescifrado.decrypt(cifrado)
44
45     return datos
46
47 def firmarRSA_PSS(datos, key_private):
48     h = SHA256.new(datos)
49     # print(h.hexdigest())
50     signature = pss.new(key_private).sign(h)
51
52     return signature
53
54 def comprobarRSA_PSS(datos, firma, key_public):
55     h = SHA256.new(datos)
56     # print(h.hexdigest())
57     verifier = pss.new(key_public)
58     try:
59         verifier.verify(h, firma)
60         return True
61     except (ValueError, TypeError):
62         return False
63

```

```
1 -----BEGIN ENCRYPTED PRIVATE KEY-----
2 MIIFJTBPBgkqhkiG9w0BBQ0wQjAhBgkrBgEEAdpHBAswFAQIXvQFu1oDcIMCAKAA
3 AgEIAgEBMB0GCWCGSASF1AwQBAgQQDWxw/JTz/+ULCHact/VjAgSCBNAKGJJb99L
4 592HQRh3sd8VsEVAIluwb7iJUpeRqbt+v8c3swqqGUL/mJ0tbQXMKSlapG+gLT
5 36+lXZRacst2ws/E1/8uiRAFxHCcWI3AtuQ0HZ9ywt8+JH/ELQs3MPoqKbW4IFV0
6 aHwHLnzS6b/tFNzQmYbCKJvULEk2VuF+It72US0ks2HkGP43SFHH5cZvcRZZ8NY
7 Gig3K4f4EsThHW3C6JvEtIKrjXIfUE5AmnsvhprD4sQ9N6AZwtMll+H0DZxm8hd7
8 i14kaCoC5pE0tBhrW2F8vhWBEBJ8i/wPVxY9bAtgvdDrhb5UC4/C5jsCfMlBQjGpD
9 I/fzwrttTn6ojEfR9JTLoiFzTLFH850iC1wSWmMI0rbCA4IQB1LTehpoEJE7i7DQ
10 ox3dTIZlpdmoyGwAQuw1/RHTjJQxzvVh12lPz+nJd1bspEuAS0zisQTo1oM/TR8
11 uHIrmHR9PhdCJ7jcoPSwfyaa2EJcyh+ZACUIFF91u4yCjXE8grcc1dummen58jaLP
12 Llstk1ykKipoNtWzqeFdCru2je43fn1xSvRKljlv9yilerT+bYMoNhGz/eAzit1
13 9dlwRpcJqCU1nPLNciwLHWrbm9YoNDtFPwLg0oY+K4SN5K+iSiBlnRXVp9h0vnS5
14 IjQYmDoYlEPm70aL40DjNhG9+m9u58zgYA50v40Umib/W29o2p/AN8tXJF8nMmC
15 mp0QzJKhvCGNwUc6flLg9aiH/bk4lBFevUT8pKOMlNuGGlPJL1dPTtPExC7s/T
16 zvDqALrkAgRu1WprEH/T2KIagKUBBm/garcVBkyH3C3cpRmCmf2QOK55ZVbAeAt
17 AhspCrIgfZopZtVY1MryNTrKKdQP8Le7XGpCkT3t1Y8KlrJyVkuPzg9v72zTmo7F
18 okVXWza8i6vWAHirXe3brhVCepetnU5Fg+NT6fpuQ7cPYc2WduUFIDsD3XNAcx09A
19 +ewkJNdcqQSSu3QifJGleMI1U8Cy4oB412X0VKjXhcObjNoEKc21EACZibFA8NTK
20 MV90l8ukOS7arkAu1+eSJbvLfccVUeTjTxIZZ0ENVSTQ0QpYhpfi0J359JwINX9P
21 cK5mHB5wpCCA0EqiJbstRLVn7uvK/sSuyBJvEozdwwG3fnAdJgJlqYBlc5x1bCV8
22 YALhRgJOM89PguPjfA9czALKz1mYCepuoJDAjXH9uZ0nDz13eTClqoJGgTCH4k
23 BBolUBPIlmOQgZc/+HwDi6mIL5JklFhbBadDGsMjkRb31V3fScs6pYgneRy3vd2K
24 KYOUf13tpAwWdhflzF6ZRSauDMx5zdaBq9R8RMSUHTf0CHjAAbzG3MZ9jhzqDy
25 /WRYugEup+ok34YPXEV3LOcpCbgxB3mQqIjzaZdStWPdYDohMONEwuJ/UHe7HEv1
26 Gpr3Jnlmtab0Rihl2EVbhc8H5/7IIzhF53F/WDNzhENKduv5CNqpCWw4JvNMwCvk
27 ZYq4RWJB/VxfZ3rkHyIIKmA Af2c1cc8Ro4vvp+bsRd2lLM2IZqyq7c+0GMRJyh91
28 hcAHicc0dXTsZt90WIYElyjkWk115I/rX9aPn45L8HYqpCjJ6A8PAhwxzSrIRYmB
29 78yy4QSExxQfpqa4RhRrhLE4XCujPOE55Q==
30 -----END ENCRYPTED PRIVATE KEY-----
```

**Que no te escriban poemas de amor
cuando terminen la carrera** ➤➤➤➤➤



WUOLAH

(a nosotros por suerte nos pasa)

No si antes decirte
Lo mucho que te voy a recordar

Pero me voy a graduar.
Mañana mi diploma y título he de
pagar

Llegó mi momento de despedirte
Tras años en los que has estado mi
lado.

Siempre me has ayudado
Cuando por exámenes me he
agobiado

Oh Wuolah wuolah
Tu que eres tan bonita

File - /home/javimp2003/IdeaProjects/UMAPracs/Criptografia Asimetrica/publicaAlice.txt

```
1 -----BEGIN PUBLIC KEY-----  
2 MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAgIQTcE7pl3w2eiVdPQE  
3 VNrhYjQb5KQ0c0C7w/fkMLjs911wWyS/oFzvGIUP2kV8l7zk13ufxeFUV0va5H84  
4 Hg7mqn0sLJXOVjIOKYz7qmFBSPmhFWYBOYiSiemcq+m7fuI2Bw7KmyU7ZS4/w2AB  
5 eAg+qRRLjynvFh2Umf0OhLN72FFvWZLNqYh927KnayI2XPlc4K3SjRS57njYvoHD  
6 NknbOHALn6M1E+cma14gF3wCkynPrpLLV+dKon9CmA2n8oYDJE8WUHs fURB3FYv  
7 i+n18/GiYCOVCZ20+ZzBdgsiVb588KTB8Yo+40SAu2wnwyN5QG5wZmaVs luokxZA  
8 ZwIDAQAB  
9 -----END PUBLIC KEY-----
```

WUOLAH

```
1 -----BEGIN ENCRYPTED PRIVATE KEY-----
2 MIIFJTBPBgkqhkiG9w0BBQ0wQjAhBgkrBgEEAdpHBAswFAQId4s+cbBgBMCAKAA
3 AgEIAgEBMB0GCWCGSALAwQBAgQQxCW+ap+kJCA7c3brDcIuiQSCBNBmB3Y3jElZ
4 NBBL9c6JmyebaL4lIwHMqoTtf+1MgXDJP1KU/wLTC10l+Z4M7CUHqvSWgJDjV
5 t54z/W3+uYNNlmUFnlnHZ+Hm0nnKC2RFW37MAygsKW87KozIOzGwsYlyE5K7GmLB
6 NJdS/5Z/rZc7CSJDjiLDNKTMS6S+hwP84Uy/wbZcJHkp3pnCirNCLqjvifRwt7mJ
7 UAWGfGO6PFi6jjqdFRd0M6UED0mDE57lNMz1kCAwPVrIOCD7ZQHNvK/aw5fjG5/x
8 u0H3IDJrfE3IwQpNif4ONBG2ss//enBj45LT+bp4QaM9z+pNAIxNVYQQk2AGggdND
9 8u5Cd/L0wzF3+r7MHzPX7vV2uwVPw0d+E+KJShobDkRxpxJKwqPhF4OK5pegJf4
10 wp9qHc54H1tPTZ90uasRur0U5g+pEcJVA8MjUnWb11/1nc++1rs8ebXiKFXhGg0l
11 WCals39iW/z5VjbNrimj46kKAhgeotzUm5Bo2avMJhrQ2NMwa2MSJt18jN5pNz0
12 Bwdty5msWflgJ0q4e920PhrjlCSah1HN/zXgJG5XT3tvFLgPFLCaD4RrGicgSucJ
13 JYJJJoCVLOIm1fWpFB45/bEDdbhnuYQXN+siKXMtGdt0mpJwQwib/xM0+a3a5G73/
14 NC15kiQtynZprwgaoCoa3HnCgroCGn6x1HqQ47Rb34n/te90jcd75R2tzy2sZA4
15 Xr1JTBRitRO+SKdT/m/caZjoWo45nq3yukzxeoS+dQKMXarg/QKQZns0J14xxdz
16 phaN8c9KxWkb9G0z5nn/9KcNAbrtrrpGfYB/zUCzUnoIsiSCkDj6vssQctIqhyLko
17 Kw+RbJrxNsikh47cdzKJ8pQHCgnWGwdpciu93LP3pLC70mjIjmbQXV1EqANPbdaj
18 b9y9GPLZ4yziwSnSdZuc232CL6DFyUQtXYVbbZx2vt7Aim7y13SjwRnSF0/Zbrld
19 8APY5Q58hUpqpLuZ0ptDI984NE0yz0VL8CtoV/cbbvh9swi6V/uIm8e4Y2uoFW77
20 eOXjN17Y6Sl14dUZ6MglZRVi6rvbS6Japwpc7vvRNphbqMMUbZfIgFKAq8w0DWfd
21 yIzzJhuh9R6q20l2ya2KL2vb12S9tWV4TuLJCM6S4lKWHxhi5ickSh+SpDJSQeeD
22 Icq6tjkHc3skUVlFRm6vajmkmYrop9uw0WdTbgeeVC8H/t168ftGOOHAW723xTPE
23 DS8xzLoFbAQegc6elP0mgVPeK1jumRrl/89tLfPxTIBh00QsWT2rfJEa3haySbJY
24 T67UrOrH9IXeMlxPd599N6srE2nG4gIdIb6pq6GcGdjnFFu3kk14oqpNmYNPIJ+q
25 7tZsLcPxMvtPkb2/0kmDRzX7zN/C7nXXXvx5qANAzHQ46dBdMgfTpqXdicPEjET
26 sgpHFyzis6m6h5jKhuTnLdw6g5RymrWVs10pzWyq2ySkMBEAJJuRy5EP6a7Uzrd
27 Lspw4fMmHlZmUK+cumL7PWkz1na4HTRQyhPoP7S5Xxq7yAtZG2p77KyGjd1Ho1eQ
28 NmIJJ4LMEdr9ZBEVzwUSf4w7561w5FXejjTECqacB0lyq93zhhn16fyNAN+0VnSr
29 Ok3KJQRjuQdMcQ7V1+gdBGayJJ8Vni0eQ==
30 -----END ENCRYPTED PRIVATE KEY-----
```