assignment1_rsa_signatures

March 3, 2023

1 Assignment 1

1.1 RSA signatures task (2)

1.1.1 Malthe Faurschou Tøttrup au644177

1.1.2 Problem context:

In this task we want to show the grade 12 to a server. The professor has made a server that can give grades automatically. Once a grade is given it can be sendt back to the server in form of a cookie to be verified. The issue is that the server will never give a grade that is higher than 10. Therefore, we need to construct this grade message ourselves.

Since the source code for the server is given, we can see what the message the server expects for the grade 12 looks like. However, the server uses RSA to verify messages, and therefore we cannot simply send this message to the server without a valid signature to go with it.

The server provides a service for signing messages. However, this service is smart enough to detect messages containing grade information such as '12', and will reject the request. Therefore it is neccessary to bypass this message filter.

1.1.3 The attack:

We want to send message m with valid signature s to the server in form of a cookie. However, s cannot be obtained directly and will need to be forged from two different signatures s_1 and s_2. These will need to be obtained through the signing of two randomly looking messsages m_1 and m_2. However, the following relationship must be true:

$m = m_1 m_2 \texttt{mod} N$

We can obtain signatures s_1 and s_2 by sending m_1 and m_2 to the signature generating service. Once we have s_1 and s_2 we can calculate s as follows:

$s = s_1 s_2 \texttt{mod} N$

Once we have s we can convince the server that our message m is valid and we did indeed get a 12.

[17]:
```
# import libraries
```

```
import requests
import json
import random
import base64
```

[18]:
```
srv_url = 'http://localhost:5000' # server url

# get public key and N from server

res = requests.get(srv_url + '/pk')
j = json.loads(res.text)
N = j['N']
e = j['e']

# compute private key
# d = pow(e, -1, phi(N))
```

[19]:
```
# figureing out string to hex to int to hex to string gymnstics

m = 'You got a 12 because you are an excellent student! :)'

m_hex = m.encode('utf-8').hex() # convert to hex

# print(m_hex)

m_string = bytes.fromhex(m_hex) # convert back to string

# print(m_string)

m_int = int.from_bytes(m_string, 'big') # convert to int

# print(m_int)

m_int_hex = m_int.to_bytes((m_int.bit_length() + 7) // 8, 'big').hex() #␣
 ↪convert back to hex

# print(m_int_hex)

m_int_string = bytes.fromhex(m_int_hex) # convert back to string

# print(m_int_string)
```

```
596f7520676f74206120313220626563617573652079 6f752061726520616e20657863656c6c656e
742073747564656e7421203a29
b'You got a 12 because you are an excellent student! :)'
151351788555452735223904455039834250872377730862888030415187484362466989940706556
152140647234255473905853415615608708528141911 45
```

```
596f7520676f74206120313220626563617573652079 6f752061726520616e20657863656c6c656e
742073747564656e7421203a29
b'You got a 12 because you are an excellent student! :)'
```

Now we need to make two seemingly random messages m_1 and m_2. We start by picking a random integer m_1 that divides m. Then we can calculate m_2 as follows:

$m_2 = m/m_1 \mathrm{mod} N$

Afterward, we verify that the messages have the following relationship:

$m_1 m_2 = m \mathrm{mod} N$

[20]:
```
# generate random looking messages m1 and m2

m1 = random.randint(1, 100) # pick random starting number between 1 and 100

# loop until m1 divides m_int
while(m_int % m1 != 0):
    m1 += 1

# calculate m2
m2 = m_int // m1 % N

# check if m1 * m2 is equal to m_int % N
if (m1 * m2) == m_int % N:
    print('works')
else:
    print('doesnt work')
```

```
works
```

For the next step, we need to get signatures s_1 and s_2 for m_1 and m_2 respectively. To this end, we can send the messages to the server signing service. If the messages are not caught by the message filter the server will return the signatures.

[21]:
```
# sign m1 and m2

m1_hex = m1.to_bytes((m1.bit_length() + 7) // 8, 'big').hex() # convert to hex

m1_res = requests.get(f'{srv_url}/sign_random_document_for_students/{m1_hex}')

m1_j = json.loads(m1_res.text)
s1 = m1_j['signature']


m2_hex = m2.to_bytes((m2.bit_length() + 7) // 8, 'big').hex() # convert to hex

m2_res = requests.get(f'{srv_url}/sign_random_document_for_students/{m2_hex}')
```

```
m2_j = json.loads(m2_res.text)
s2 = m2_j['signature']

# check signatures
# print(str(m1_j['signature']))
# print(str(m2_j['signature']))
```

Now we can forge the signature s from s_1 and s_2 by using the formula:

$s = s_1 s_2 \mathrm{mod} N$

[22]:
```
# compute signature for m
s1_int = int.from_bytes(bytes.fromhex(s1), 'big')
s2_int = int.from_bytes(bytes.fromhex(s2), 'big')

s = (s1_int * s2_int) % N

s_hex = s.to_bytes((s.bit_length() + 7) // 8, 'big').hex() # convert to hex

# print(s_hex)
```

Now with a valid signature s for the message m, we should be able to convince the server that we got the grade 12. The server expects the signature and message encoded using base64 and packaged in a cookie.

[23]:
```
# send the grade message and signature

grade = base64.b64encode(json.dumps({'msg': m_hex, 'signature': s_hex}).
↪encode()).decode() # thanks CoPilot

res = requests.get(f'{srv_url}/grade', cookies={'grade': grade})

print(res.text) # celebration!
```

```
<p>You got a 12 because you are an excellent student! :)</p>
```

Finally we can get a quote!

[24]:
```
# recieve a quote

res = requests.get(f'{srv_url}/quote', cookies={'grade': grade})

res.text # hmm interesting quote
```

[24]:
```
'<quote>\n<redacted> for testing purposes\n</quote>'
```