Group: Cicada 3301
Members: Eric, Sidney, Xuexuan, Zheyu

Category: Exploit

Challenge name: You are THE MAN!

Challenge:
Alice and Bob are cheating at Computer Vision, using DHE-RSA for communication! Details here: https://www.linkedin.com/pulse/ephemeral-diffie-hellman-rsa-dhe-rsa-william-buchanan/

You are THE MAN (in the middle) working for Man! To gain evidence of this nefarious activity, you want Bob to provide a reply for image 7258.

DHE-RSA is meant to be robust to man in the middle attacks, but crack it anyway. The CA's certificate is provided (CA.crt).

You will need Python 3.6, rsa and pycrypto libraries:

- rsa: https://pypi.python.org/pypi/rsa
- pycrypto: https://pypi.python.org/pypi/pycrypto/2.6.1 (py3.6 can import it, don't worry.)

You are highly encouraged to read the long challenge document.

Group: Cicada 3301
Members: Eric, Sidney, Xuexuan, Zheyu

Category: Exploit

Challenge name: You are THE MAN!

Long Challenge description:
You are THE MAN! (No, I don't care if you are female.)

Alice and Bob are cheating in CV. Alice's program doesn't do computer vision, it does human vision. Bob is part of Cognitive Accomplice, an underground group of students that secretly provides image recognition to people who have registered with their group. To hide their tracks, they don't communicate with each other elsewhere (thus no shared secret key). You are THE MAN working for Man and want to get to get proof of this.

In class $50\frac{1}{50}$, students have learnt about the weakness of Diffie Hellman Key Exchange (DHKE) to Man in the middle attacks. Thus, Cognitive Accomplice implements a better version of DHKE, called DHE-RSA.[1]

To send their public keys with integrity, the group, Cognitive Accomplice, acts as a certificate authority to sign Alice and Bob's public keys.

To cover their tracks too, Cognitive Accomplice also functions as a public certificate authority for poor starving students (like Alice and Bob).
Their certificate uses a simplified json format with only 3 fields:

- subject_principal
  - The ID or name of the certificate holder.
- rsa_pub
  - The RSA public key of the certificate holder in PEM format.
- Signature
  - base64 encoded string of the signature of the certificate.
  - i.e. this is RSA(H(certificate_content), CA_private_key)
  - Hash function H(x) = sha256(x) % RSA modulus (n).
  - certificate_content is "subject_principal"+"|"+"rsa_pub", all concatenated together

The CA's public key certificate is ca.crt.

They also kindly provided cert.py publicly, a library to interface with their certificate files and sign data (python dicts). As a new certificate authority, they only support python 3.6.
You will also need rsa and pycrypto libraries:

- rsa: https://pypi.python.org/pypi/rsa
- pycrypto: https://pypi.python.org/pypi/pycrypto/2.6.1 (py3.6 can import it, don't worry.)

Now, you, THE MAN, has access to the router Alice and Bob are communicating on. DHE-RSA is meant to be robust to man in the middle attacks, but crack it anyway.

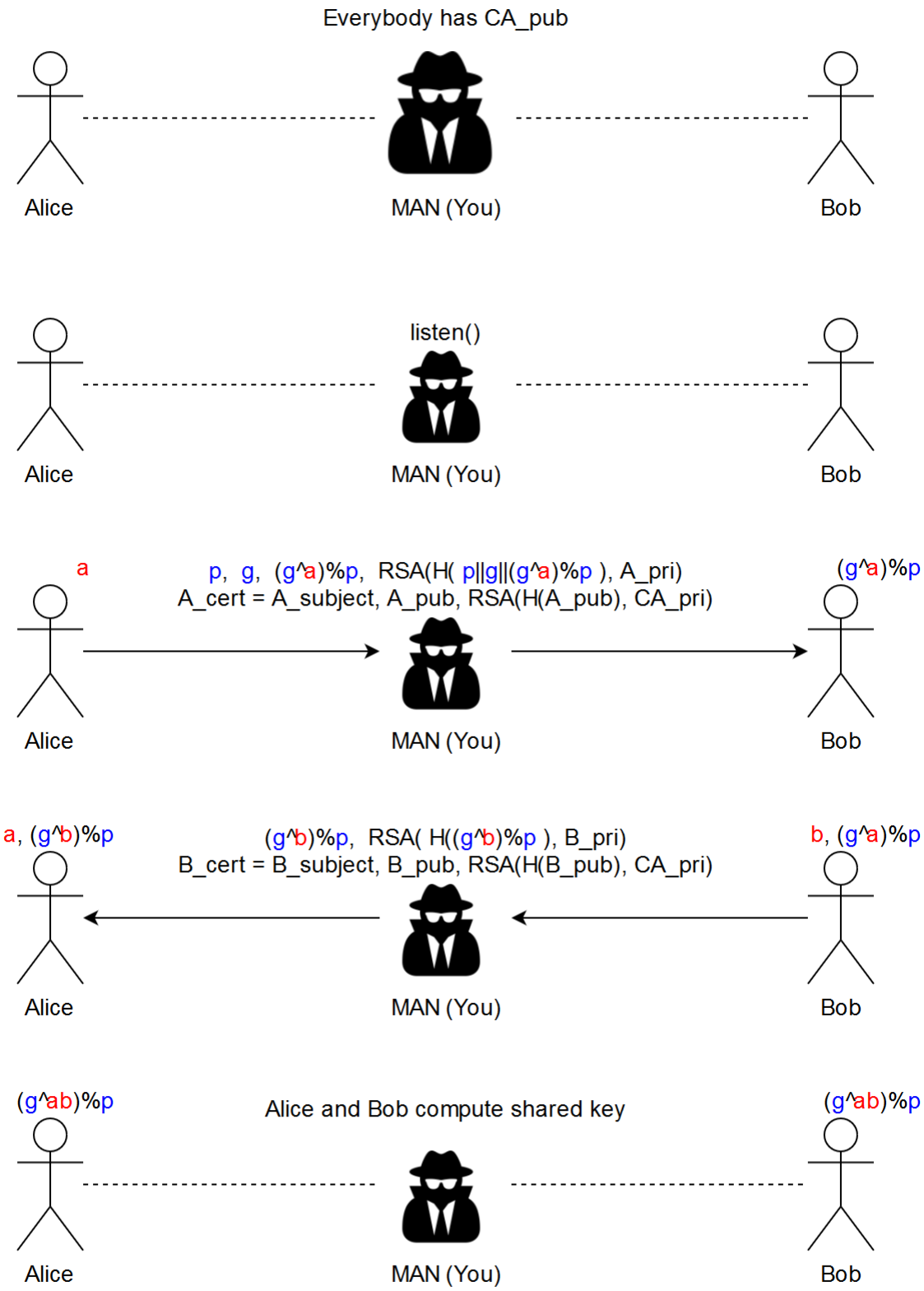
---

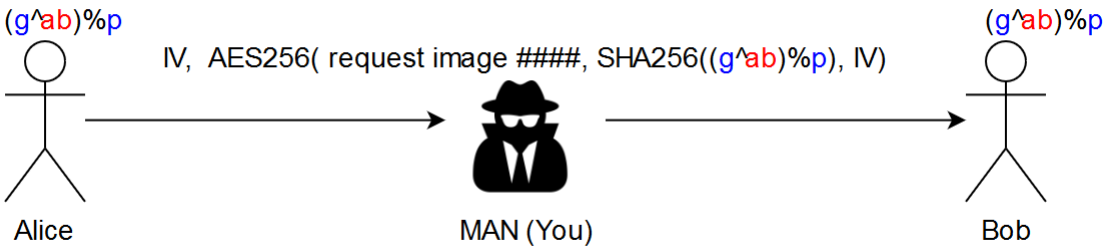
[1] Details here: https://www.linkedin.com/pulse/ephemeral-diffie-hellman-rsa-dhe-rsa-william-buchanan/

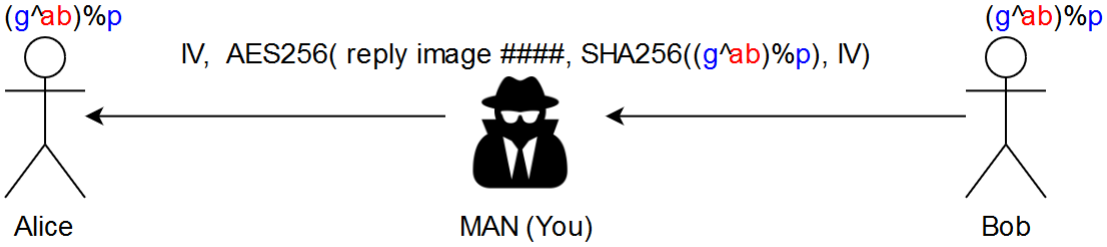
To get evidence, you want Bob to send a reply for image number 7258.

Hint: A fast prime factoriser is available here: https://www.alpertron.com.ar/ECM.HTM

The DHE-RSA protocol and Alice and Bob's conversation sequence is illustrated in the pictures below.




(g^ab)%p
IV, AES256(request image ####, SHA256((g^ab)%p), IV)
(g^ab)%p
Alice
MAN (You)
Bob


(g^ab)%p
IV, AES256(reply image ####, SHA256((g^ab)%p), IV)
(g^ab)%p
Alice
MAN (You)
Bob

Group: Cicada 3301
Members: Eric, Sidney, Xuexuan, Zheyu

Category: Exploit

Challenge name: You are THE MAN!

Solution:
The solution can be easily explained in the diagrams. The certificate authority's RSA key is 2048bits, but it is a perfect square. Thus, it can be factorised very easily.

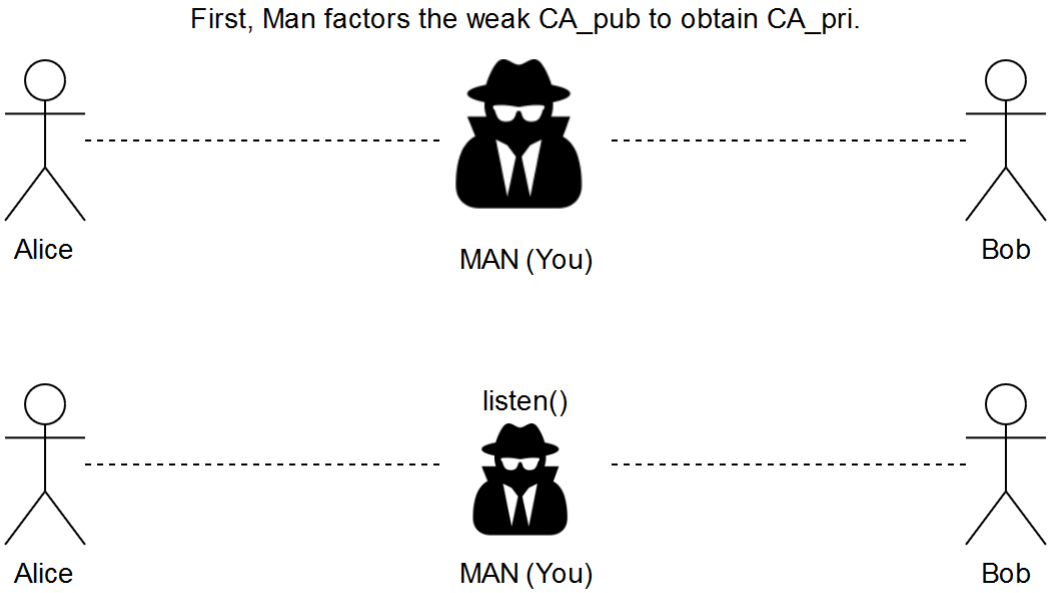
After factorisation, the euler totient $\phi(n) = p(p-1)$ can be calculated. It is $p(p-1)$ instead of $(p-1)(q-1)$ as n is a perfect square. Then the private key $d = e^{-1}\ mod\ \phi(n)$ can be computed. In cert.py, an inverse modulo function is linked from the rsa library for convenience.

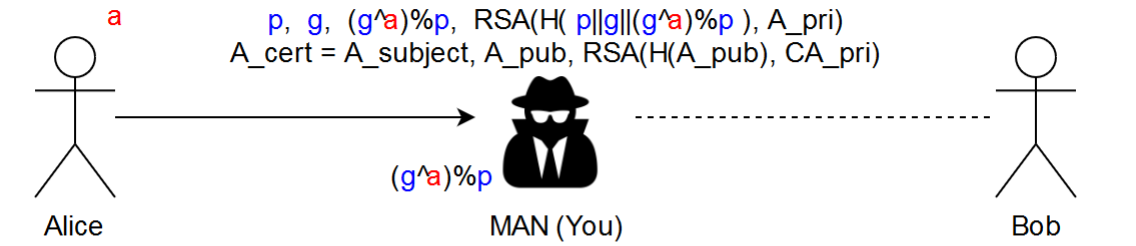
This allows MAN to create A'_cert and B'_cert, and thus pretend to be Alice and Bob respectively.

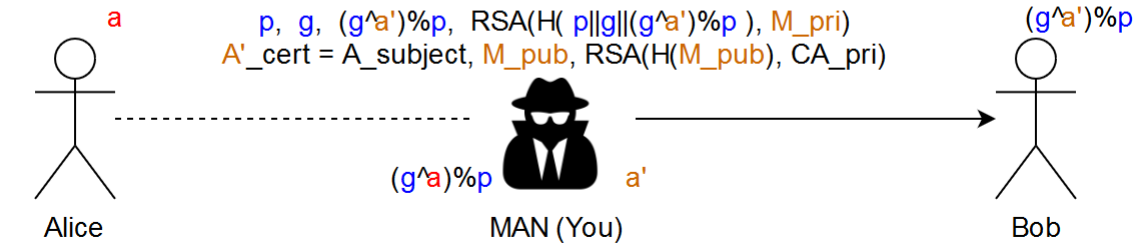
It should also be noted that once the format of A's request is known, A can be ignored and MAN simply communicates to Bob directly. (not shown in diagram)

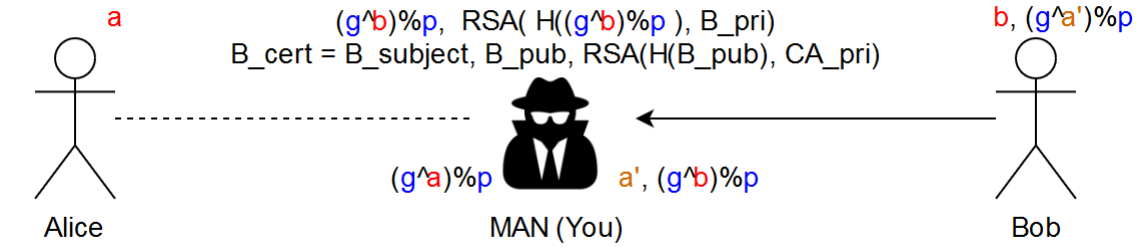
Running cert_maker.py will prompt for the factorisation of n, it can be computed in script (as a perfect square), but this is more generalised. Then, it will generate the fake A'_cert and B'_certs.

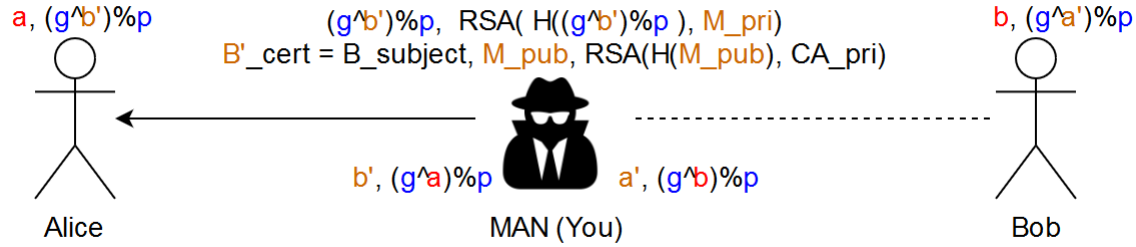
Runnig router_solution.py will execute the attack and print out the messages being sent.

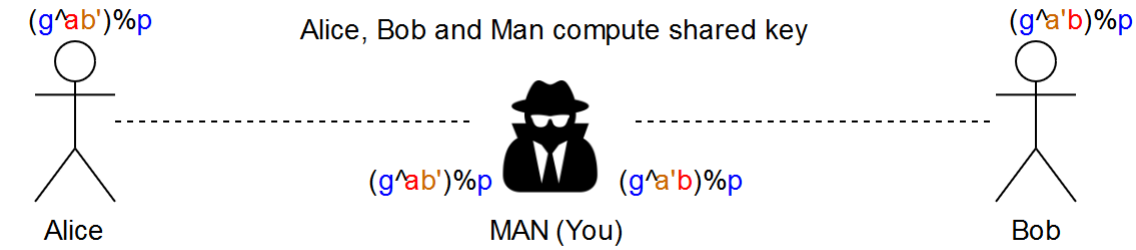


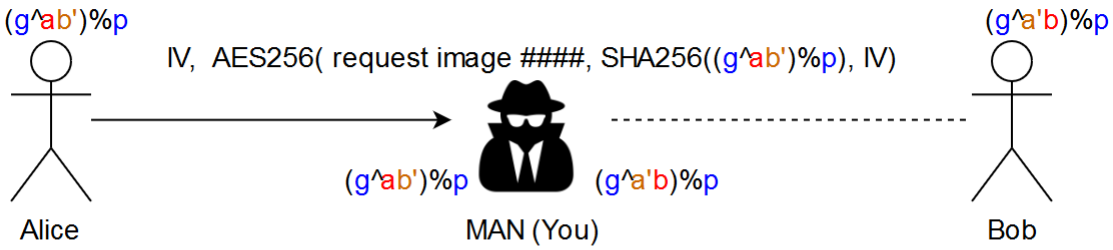

a
p, g, (g^a)%p, RSA(H(p||g||(g^a)%p), A_pri)
A_cert = A_subject, A_pub, RSA(H(A_pub), CA_pri)
(g^a)%p
Alice
MAN (You)
Bob

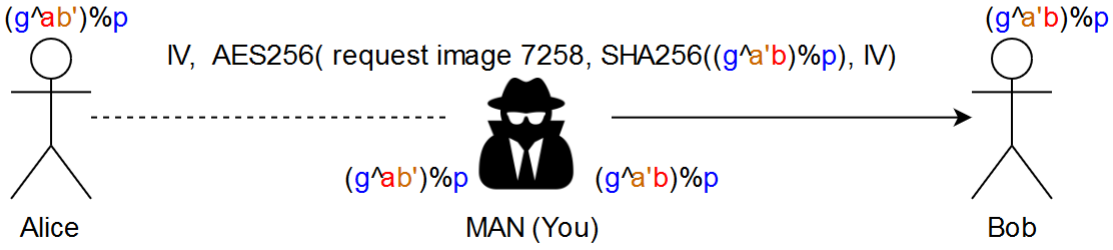

a
p, g, (g^a')%p, RSA(H(p||g||(g^a')%p), M_pri)
A'_cert = A_subject, M_pub, RSA(H(M_pub), CA_pri)
(g^a')%p
(g^a)%p
a'
Alice
MAN (You)
Bob

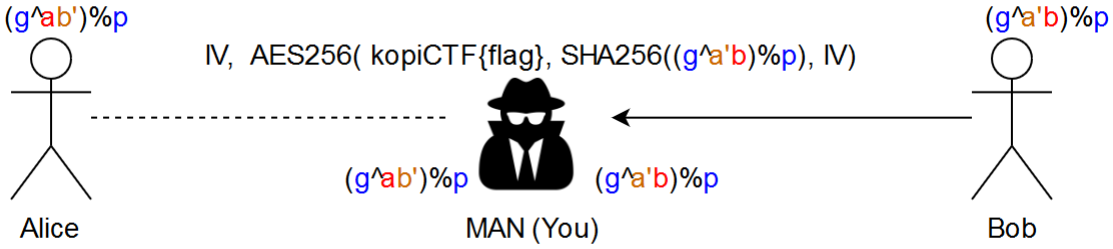

a
(g^b)%p, RSA(H((g^b)%p), B_pri)
B_cert = B_subject, B_pub, RSA(H(B_pub), CA_pri)
b, (g^a')%p
(g^a)%p
a', (g^b)%p
Alice
MAN (You)
Bob


a, (g^b')%p
(g^b')%p, RSA(H((g^b')%p), M_pri)
B'_cert = B_subject, M_pub, RSA(H(M_pub), CA_pri)
b, (g^a')%p
b', (g^a)%p
a', (g^b)%p
Alice
MAN (You)
Bob


(g^ab')%p
Alice, Bob and Man compute shared key
(g^a'b)%p
(g^ab')%p
(g^a'b)%p
Alice
MAN (You)
Bob


(g^ab')%p
(g^a'b)%p
IV, AES256(request image ####, SHA256((g^ab')%p), IV)
(g^ab')%p
(g^a'b)%p
Alice
MAN (You)
Bob


(g^ab')%p
(g^a'b)%p
IV, AES256(request image 7258, SHA256((g^a'b)%p), IV)
(g^ab')%p
(g^a'b)%p
Alice
MAN (You)
Bob


(g^ab')%p
(g^a'b)%p
IV, AES256(kopiCTF{flag}, SHA256((g^a'b)%p), IV)
(g^ab')%p
(g^a'b)%p
Alice
MAN (You)
Bob