

Project 3: Crypto – Have fun with RSA

CS4235/6035 : Prof. Wenke Lee (wenke@cc.gatech.edu)

February 27, 2018

1 Intro – RSA

RSA is one of the most widely used public key cryptosystems in the world. It's composed of three algorithms: key generation (**Gen**), encryption (**Enc**), and decryption (**Dec**). In RSA, the public key is a pair of integers (e, N) , and the private key is an integer d .

Gen The key pair is generated by the following steps:

1. Choose two distinct big prime numbers with the same bit size, say p and q .
2. Let $N = p * q$, and $\phi(N) = (p - 1) * (q - 1)$.
3. Pick up an integer e , such that $1 < e < \phi(N)$ and $\gcd(e, \phi(N)) = 1$.
4. Get the modular inverse of e : $d \equiv e^{-1} \pmod{\phi(N)}$ (i.e., $d * e \equiv 1 \pmod{\phi(N)}$).
5. Return (N, e) as public key, and d as private key.

Enc To encrypt integer m with public key (N, e) , the cipher integer $c \equiv m^e \pmod{N}$.

Dec To decrypt cipher integer c with private key d , the plain integer $m \equiv c^d \pmod{N}$.

2 Task1 – Get Familiar with RSA (10 points)

The goal of this task is to get you familiar with RSA.

You're given a RSA key pair (N, e) and d , and an unique encrypted message c . You're required to get the decrypted message m . Each student's key pair and cipher text can be found in *"keys4student.json"*.

You're only required to submit your decrypted message in hex format.

3 Task2 – Attack Small Key Space (20 points)

In the real world, a commonly used RSA key size is 1024 bits, which makes it hard for attackers to traverse the whole key space with limited resources. Now, you're given an unique RSA public key, of which the key size is pretty small (64 bits), your goal is to get the private key. All public keys can be found in *"keys4student.json"*.

You're required to write some code in *"get_pri_key.py"* to get the private key:

- TODO1: implement function `get_factors`, n is the given public key (64 bits), your goal is to get its factors. You can cheat on this subtask (i.e., search engine, pencil and paper), as long as you can get the right p and q .

```
def get_factors(n):  
    p = 0  
    q = 0  
  
    # your code starts here  
  
    # your code ends here  
    return (p, q)
```

- TODO2: implement function `get_key` to get the private key.

```
def get_key(p, q, e):  
    d = 0  
  
    # your code starts here  
  
    # your code ends here  
    return d
```

You're required to submit: (1) your unique private key in hex format; (2) the *"get_pri_key.py"* file; (3) a brief description about your steps to get the private key.

4 Task3 – Where Is Waldo? (30)

Read the paper "Mining Your Ps and Qs: Detection of Widespread Weak Keys in Network Devices" (<https://factorable.net/weakkeys12.extended.pdf>).

You're given an unique RSA public key, and the RNG (random number generator) used in the key generation is vulnerable. Also, all your classmates' public keys are generated by the same RNG on the same system. Your goal is to get your unique private key. All keys can be found in *"keys4student.json"*.

You're required to complete some code in *"find_waldo.py"* to get the private key:

- TODO1: implement function `is_waldo`, *n1* is your own key, *n2* is one of your classmate's key, try to find out whether this classmate is Waldo.

```
def is_waldo(n1, n2):
    result = False

    #your code starts here

    #your code ends here

    return result
```

- TODO2: since you've successfully found your Waldo among your classmates, now you have to implement function `get_private_key` to get your own unique private key. *n1* is your public key, *n2* is Waldo's public key.

```
def get_private_key(n1, n2, e):
    d = 0

    #your code starts here

    #your code ends here

    return d
```

You're required to submit: (1) your unique private key in hex format; (2) your classmate's (Waldo) name; (3) the *"find_waldo.py"* file; (4) your understanding about the weak key problem caused by Ps and Qs; (5) a simple description about your steps to get the private key.

5 Task4 – Broadcasting RSA Attack (40)

A message was encrypted with three different 1024 bit RSA public keys, all of them have the public exponent $e = 3$, resulting in three different encrypted messages. You're given the three pairs of public keys and encrypted messages, please recover the original message. You're required to implement the 'recover_msg' function in *"recover.py"*:

```
def recover_msg(n1, n2, n3, c1, c2, c3):
    m = 42
    # your code starts here: to calculate the original message - m
```

```
# Note 'm' should be an integer

# your code ends here

# convert the int to message string
msg = hex(m).rstrip('L')[2:].decode('hex')
return msg
```

You're required to submit the original message, your understanding about the attack and a brief explanation about how you recover the message.

6 Submission

Note that all students' keys are different, so don't copy and paste answers from your classmates. In total, please submit the following files:

1. "studentid_submission.json".
2. "get_pri_key.py", "find_waldo.py" and "recover.py".
3. "report.pdf": a brief explanation about how you get the private key in task2; your understanding about the Ps and Qs problem in task3; a brief explanation about how you get the private key and waldo in task3; your understanding about the attack in task4; a brief explanation about how you recover the message in task 4.

Please submit the files separately, don't archive them!