Cryptography Tasks

Cryptography Tasks 1
Introduction 2
Notice 3
Arithmetic Operators in Python 3
Function Introduction 3
Fill Up Student Number Before Submission 4
Submission 4
Task 1 – Decryption (10 Points) 5
Coding (5 Points) 5
Report (5 Points) 5
Task 2 – Small Key Space Attack (10 + 10 Points) 6
Coding (10 Points) 6
Report (10 Points) 6
Task 3 – Ps & Qs Attack (15 + 15 Points) 7
Coding (15 Points) 7
Report (15 Points) 7
Task 4 – Broadcast RSA Attack (20 + 20 Points) 8
Coding (20 Points) 8
Report (20 Points) 8

Introduction

For all the tasks about cryptography, we choose RSA, a traditional public key cryptography system, as the core content. Being a challenger of all the tasks, you should read the task details and complete every single one of the tasks assigned.

To make it clear and easy to learn, you are required to finish the tasks in the order of difficulty; they are:

1. Task 1 – Decryption.
2. Task 2 – Small Key Space Attack.
3. Task 3 – Ps & Qs Attack.
4. Task 4 – Broadcast RSA Attack.

Please read the instructions below carefully and complete the four tasks. You should also notice that **NO third-party modules or libraries are allowed to help you finish the tasks!**

Notice

Arithmetic Operators in Python

Before getting started, you should learn some basic Python coding about the arithmetic operation:

- $a + b$, the sum of the variables a and b.
- $a - b$, the result of subtracting variable b from variable a.
- $a * b$, the product of the variables a and b.
- a/b, the division operation of variables a and b on the real number domain. For example, let $a == 23$ and $b == 5$, then $a/b == 4.6$.
- $a//b$, the result of variable a divides b evenly. For example, let $a == 23$ and $b == 5$, then $a//b == 4$. To perform division on integers, please always use //.
- $a ** b$, i.e., a to the power of b.
- $a\%b$, the remainder of dividing a by b. For example, let $a == 23$ and $b == 5$, then $a\%b == 3$.
- $pow(a, b, c)$, the result of $a^b\%c$. Although the result is the same as $(a ** b)\%c$, we strongly recommend you use $pow(a, b, c)$ for better performance.

The operators discussed above will be supported by Python3. Please refer to the link to install Python3 on your PC: https://www.python.org/downloads/.

Function Introduction

Some functions have been prepared for you in the code templates so that you do not have to be bothered by complicated implementations; that is, you can use them directly in your code:

- $gcd(a, b)$ is used to get the greatest common divisor of a and b. For example, let $a == 18$ and $b == 24$, then $gcd(a, b) == 6$.
- $gcde(a, b)$ implements Extended Euclidean Algorithm. Besides the greatest common divisor of a and b, it can also find a pair of x and y who satisfy $ax + by = gcd(a, b)$. For example, let $a == 18$ and $b == 24$, then $gcde(a, b)$ will return a triple tuple $(6, -1, 1)$, where 6 is the greatest common divisor of 18 and 24, and $-1 * 18 + 1 * 24 == 6$.
- $modinv(a, b)$ can help us get the modular inverse of b under modulus a. I.e., it returns x that $b * x \ mod \ a == 1$. For example, let $a == 7$ and $b == 17$, then $modinv(a, b) == 5$ because $17 * 5 \%7 == 1$.
- $root3(a)$ is used to get the cube root of a. For example, let $a == 27$, then $root3(a) == 3$.

Fill Up Student Number Before Submission

At the end of each task file, there is a function where you should fill in your student number.

```
def get_student_number():
    # TODO: Fill your student number here
    return ""
```

For example, if your student number is **3031234567**, it should be like:

```
def get_student_number():
    # TODO: Fill your student number here
    return "3031234567"
```

Submission

1. Please submit four Python code files.
    a. *decrypt_message.py* from task 1
    b. *short_key.py* from task 2
    c. *get_private_key.py* from task 3
    d. *broadcast_attack.py* from task 4

   DO NOT CHANGE THE FILE NAMES, as we will run auto-grading scripts to test your code.
2. Besides submitting the code files, you should also submit a PDF report to answer the questions in each task.

Please put all the code files and the PDF report under a folder named with your student ID, compress the folder to *.zip or *.tar, and submit the compressed file.

Task 1 – Decryption (10 Points)

Coding (5 Points)

At the very beginning, you should know how RSA works in encryption & decryption. Because of the symmetry, you can focus on the decryption only. You are given an RSA key pair (N, e) and d (that is, the decryption exponent), and a unique encrypted message c. You should get the decrypted message m.

Please complete the function *decrypt_message* in the given Python file ***decrypt_message.py***.

```
def decrypt_message(N, e, d, c):
    m = 0
    # TODO: Implement this function for Task 1
    return hex(m).rstrip('L')
```

To check whether your code is correct or not, please first fill your student ID in the function *get_student_number*. Then, you can run the Python file "test.py". If it prints out the message "**The decryption is correct!**", your code is correct; otherwise, please fix the code according to the error information.

Report (5 Points)

Please explain the following questions:

1. How does RSA encrypt a message? (2 Points)
2. How can we decrypt the message if we have the private key? (3 Points)

Task 2 – Small Key Space Attack (10 + 10 Points)

Coding (10 Points)

RSA is often under the threat of attacks, especially when the length of the key is short. In this task, you are given a unique RSA public key with a fairly small key length (64bits). Your goal is to get the private key.

This task is split into two steps, each worth 5 points. First, n is given, which is the public key, and your goal is to get the factors p and q. Finish the function ***get_factors*** in ***short_key.py***.

```
def get_factors(n):
    p, q = 0, 0
    # TODO: Implement this function for Task 2
    return p, q
```

After that, you are required to implement the function ***get_private_key_from_p_q_e*** to get the private key.

```
def get_private_key_from_p_q_e(p, q, e):
    d = 0
    # TODO: Implement this function for Task 2
    return d
```

To check whether your code is correct, please first fill your student ID in the function *get_student_number*. Then, you can run the Python file "test.py". If it prints out the message "**The p & q are correct!**", your implementation of ***get_factors*** is correct, and if it prints out the message "**The private key is correct**", your code is correct.

Report (10 Points)

Please explain the following questions:

1. How do you get the factors p and q from the given public key? (5 Points)
2. How do you generate the relative private key from the factors? (5 Points)

Task 3 – Ps & Qs Attack (15 + 15 Points)

Coding (15 Points)

Read the paper "Mining Your Ps and Qs: Detection of Widespread Weak Keys in Network Devices", which can be found at https://factorable.net/weakkeys12.extended.pdf.

You are given a unique RSA public key, but the RNG (Random Number Generator) used in the key generation is vulnerable. In addition, all of your classmates' public keys were generated by the same RNG on the same system. Your goal is to get your unique private key.

The first step is worth 5 points, in which you should implement function ***is_waldo*** in file ***get_private_key.py***. $n1$ is your key, and $n2$ is one of your classmate's key. Try to determine whether this classmate is Waldo. In other words, this function should return True if $n1$ and $n2$ share the same prime number; otherwise, it returns False.

```
def is_waldo(n1, n2):
    result = False
    # TODO: Implement this function for Task 3
    return result
```

The second step worth 10 points, that since you have successfully found Waldo amongst your classmates, you should implement ***get_private_key_from_n1_n2_e*** to get your own unique private key, while $n1$ is your key and $n2$ is your classmate's public key.

```
def get_private_key_from_n1_n2_e(n1, n2, e):
    d = 0
    # TODO: Implement this function for Task 3
    return d
```

To check whether your code is correct, please first fill your student ID in the function *get_student_number*. Then, you can run the Python file "test.py". If it prints out the message "**Your waldo is correct!**", your implementation of ***is_waldo*** is correct, and if it prints out the message "**The private key is correct**", your code is correct.

Report (15 Points)

Please discuss the following questions in the report:

1. What is the workflow of Ps and Qs attack? (10 points)
2. How can we prevent Ps and Qs attack? (5 points)

Task 4 – Broadcast RSA Attack (20 + 20 Points)

Coding (20 Points)

A message was encrypted with three different 1024-bit RSA public keys, resulting in three different encrypted messages. All of them have the public exponent $e = 3$.

You are given the three pairs of public keys and associated encrypted messages. You need to recover the original message.

Implement the function ***recover_msg*** in file ***broadcast_attack.py***.

```
def recover_msg(N1, N2, N3, C1, C2, C3):
    m = 0
    # TODO: Implement this function for Task 4
    return m
```

To check whether your code is correct, please first fill your student ID in the function *get_student_number*. Then, you can run the Python file "test.py". If it prints out the message "**The plain text is correct!**", your code is correct.

Report (20 Points)

Please discuss the following questions in the report:

1. What is the workflow of the broadcast attack? (15 Points)
2. Can we recover the message with two ciphertexts instead of using three? (5 Points)