Project 3 :

All Things Cryptography

Summer, 2019

The goals of this project :

Students will advance their knowledge of cryptography and hashing by working through example exercises and then trying to exploit some vulnerable systems.

Intro :

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.

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 $\varphi(N) = (p-1) * (q-1)$.

3. Pick up an integer e, such that $1 < e < \varphi(N)$ and $gcd(e, \varphi(N)) = 1$.

4. Get the modular inverse of e: $d \equiv e^{-1} \bmod \varphi(N)$ $(i.e., d * e \equiv 1 \bmod \varphi(N))$.

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

Task 1 – Get Familiar with RSA - (5 points)

The goal of this task is to get you familiar with RSA. You are given an RSA key pair (N, e) and d, and a unique encrypted message c. You are required to get the decrypted message m. Each student's key pair and cipher text can be found in `keys4student_task_1.json`

```
def decrypt message (self, N, e, d, c):
    m = 0
    return hex(m).rstrip('L')
```

Task 2 – Can I Have Some Salt With My Password? (10 points)

By now we all know that it is important not to use common passwords. Hackers have long utilized rainbow tables, which contain hashes of common passwords, to crack vulnerable passwords. Passwords are often salted before they are hashed in order to thwart a rainbow table attack. Assuming that your weak password is salted before it is hashed, **how much more protection does this offer you?**

Let's find out...

You are given the **SHA256** hash of a password randomly selected from a list of the 1,000 most commonly-used passwords on the Internet. A salt value was added to the password before it was hashed. Your job is to decode the original password and salt value from the hash. (To make your job a little easier, the salt value is another random password from the same list of common passwords.)

- All password hashes can be found in `hashes4student_task_2.json`.
- The complete list of common passwords can be found in `top_passwords.txt`.

TODO: In the provided `crypto_proj.py` file, implement the function `crack_password_hash :`

```
def crack password hash (self, password hash , weak password list ):
    password = 'abc'
    salt = '123'
    hashed_password = hashlib.sha256(password .encode () +
                                     salt.encode ()).hexdigest()

return password, salt
```

Reflection

In your essay address the following questions:

- Is the hash of the salted password still vulnerable? Why or why not?
- What steps could be taken to enhance security in this situation?

Task 3 – Attack A 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 a unique RSA public key, with a fairly small key size (**64 bits**).

Your goal is to get the private key. All public keys can be found in `keys4student_task_3.json`.

TODO: In the provided `crypto_proj.py` file, implement the function `get_factors`. n is the given public key (**64 bits**), and your goal is to get its factors.

```
def get_factors (self, n):
    p = 0
    q = 0
    return p, q
```

TODO: In the provided `crypto_proj.py` file, implement the function `get_private_key_from_p_q_e` to get the private key.

```
def get_private_key_from_p_q_e(self, p, q, e):
    d = 0
    return d
```

Reflection

In your essay address the following questions:

- What steps did you follow to get the private key?

Task 4 – Where's Waldo (30 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. All keys can be found in `keys4student_task_4.json`.

TODO: In the provided `crypto_proj.py` file, implement the function `is_waldo`. $n1$ is your own key, $n2$ is one of your classmate's key. Try to determine whether this classmate is Waldo.

```
def is waldo (self, n1 , n2):
    result = False
    return result
```

TODO: Since you've *successfully* found Waldo amongst your classmates, you now have to implement the function **`get_private_key_from_n1_n2_e`** to get your own unique private key. $n1$ is your own key, $n2$ is Waldo's public key.

```
def get_private_key_from_n1_n2_e(self, n1 , n2 , e):
    d = 0
    return d
```

Reflection

In your essay address the following questions:

- What makes the key generated in this situation vulnerable?
- What steps did you follow to get the private key?

Task 5 – Broadcast RSA Attack (35 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. Your job is to recover the original message.

All keys and messages can be found in `keys4student_task_5.json`.

TODO: In the provided `crypto_proj.py` file, implement the function **`recover_message`**.

```
def recover_msg (self, N1 , N2 , N3 , C1 , C2 , C3):
    m = 42
    # Note that 'm' should be an integer
    return m
```

Reflection

In your essay address the following questions:

- How does this attack work?
- What steps did you follow to recover the message?

Important Notes :

You must change the student ID within the class constructor in `crypto_proj.py` to **your own student ID**!! This has to be correct.

```
def __init__(self):
      # TODO Change this to your Georgia Tech student ID!!!
      # Note that the ID below is NOT your 9-digit Georgia Tech ID
      self.student_id = 'bdornier3'
```

The crypto proj.py file has all of the modules that you will need imported for you. You are NOT allowed to alter the import list. You will lose substantial points if you do so.

Your entire submission must run in 10 minutes or less.

You are also given two unit testing files (`test_crypto_proj_1.py & test_crypto_proj_2.py`) to help you test your program. We encourage you to read up on Python unit tests, but in general, the syntax should resemble either:

```
python -m unittest test_crypto_proj_1
```

or:

```
python test_crypto_proj_2.py
```

The provided files are written in Python 3 and will be tested on a Python 3 interpreter.

HINT (as a reward for reading this far):

The answers in the `test_crypto_proj_1.py` and `test_crypto_proj_2.p` unit test files are **CORRECT** for the student IDs `bdornier3` and `ctaylor`. However, keep in mind that passing the unit tests does NOT guarantee that your code will pass the autograder!

The final deliverables:

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. `crypto_proj.py`
2. `project_report.pdf` : An essay with all of your answers to the reflection questions.

When writing your report *please* preface each section with the associated task (i.e. Task 2, Task 3, etc). There is no need to reproduce the prompts. There is no page count or word count for the report, but in the past, **highly successful reports have been between 2-5 pages**. **Please submit the files separately**, **don't archive them**! **All submissions must have ALL files.**

Good luck!