Spring 2021 Cryptography and Network Security

Homework 1

Release Date: 2021/3/9

Due Date: 2021/4/5, 23:59

Instruction

- **Submission Guide:** Please submit all your codes and report to NTU COOL. You need to put all of them in a folder named by your student id, compress it to hw1_{student_id}.zip. For example, hw1_r04922456.zip. The report must be in **PDF** format, and named report.pdf.
- You may encounter new concepts that haven't been taught in class, and thus you're encouraged to discuss with your classmates, search online, ask TAs, etc. However, you must write your own answer and code. Violation of this policy leads to serious consequence.
- You may need to write programs in the Capture The Flag (CTF) problems. Since you can use any programming language you like, we will use a pseudo extension code**.ext** (e.g., code.py, code.c) when referring to the file name in the problem descriptions.
- This homework set are worthy of 120 points, including bonus.
- You are recommended to provide a brief usage of your code in **readme.txt** (e.g., how to compile, if needed, and execute). You may lose points if TAs can't run your code.
- In each of the Capture The Flag (CTF) problems, you need to find out a flag, which is in `CNS{...}` format, to prove that you have succeeded in solving the problem.
- Besides the flag, you also need to submit the code you used and a short write-up in the report to get full points. The code should be named **code{problem_number}.ext**. For example, code3.py.
- In some CTF problems, you need to connect to a given service to get the flag. These services only allow connections from 140.112.0.0/16, 140.118.0.0/16 and 140.122.0.0/16.

Handwriting

1. CIA (10%)

Please explain three major security requirements: confidentiality, integrity and availability. For each security requirement, please give an example in the real world.

2. Hash Function (10%)

Please explain three properties of a cryptographic hash function: one-wayness, weak collision resistance and strong collision resistance.

For each property, please give an example applied in the real world.

3. ElGamal Scheme (15%)

In this problem, we want you to combine ElGamal public-key encryption and Shamir's secret sharing.
Lets recall the Shamir's Secret Sharing.

$$A(x) = a_0 + a_1 x + ... + a_{t-1} x^{t-1}$$

The secret is a_0, and the ith user will receive $(i, A(i)\ mod\ p)$, where p is a prime.

Then, let's recall the ElGamal encryption scheme.
`setup:`

$$\begin{aligned} \text{large prime} &: p \\ \text{generator} &: g \\ \text{secret key} &: sk_B = b \\ \text{public key} &: pk_B \equiv g^b (\text{mod } p) \end{aligned}$$

`encryption:`

$$\begin{aligned} \text{plaintext} &: m \\ \text{random value} &: x \\ \text{ciphertext1} &: c_1 \equiv g^x (\text{mod } p) \\ \text{ciphertext2} &: c_2 \equiv m(pk_B)^x (\text{mod } p) \end{aligned}$$

`decryption:`

$$\text{plaintext} : m = c_2 c_1^{-b} (\text{mod } p)$$

(a) (5%) How can Alice and Bob build a symmetric key by using ElGamal encryption scheme? Please explain clearly.

(b) (10%) Now you should revise the setting above to accomplish ElGamal threshold decryption, such that the ciphertext can be decrypted only if t out of n users collaborate.

Hint: Genarally, you only need to change the setup and decryption sections.

Capture The Flag

4. Simple Crypto (10%)

> "Welcome to the Crypto World. In this homework, you are going to play with some well known classical ciphers. Please solve all the classical cipher challenges yourself. Even though classical ciphers only used in the past and most of them can be practically computed and solved, I don't think you can figure it out that easily :P. Be careful and don't use classical ciphers to keep your secret!"

You can access the service by `nc cns.csie.org 17277`. If this is your first CTF challenge, highly recommend you to solve this challenge first.

5. RSA (10% + Bonus 5%)

After learning RSA in class, you may know that textbook rsa is insecure. To make it more insecure, I let $e = 3$. It's easy for you to get the flag, isn't it? You can access the system by `nc cns.csie.org 8503`. The challenge source is included in `hw1/rsa`.

Note: There are 4 flags and you don't need to solve them in order. But I think it may be easier for you to follow the order.

6. Rainbow Table (20%)

When it comes to cryptanalysis, space-time tradeoff is quite common. Rainbow table is a famous example, and we will use it to crack passwords in this problem. If you don't know what a rainbow table is, you should check it out in Wikipedia. For simplicity, the following problem description uses the naming borrowed from Wikipedia.

In the following problems, we assume that the combination of the hash function and the reduction function, i.e., $R \cdot H : P \longrightarrow P$, uniformly samples a password from P. However, if you give the same input to $R \cdot H$, it will always generate the same output.

(a) (5%) Expectedly, how many times should you call $R \cdot H$ to get $\frac{|P|}{4}$ unique passwords? (Let $|P| = 10^{10}$)

Hint: What is the expected number of rolls to get the last face of a fair dice if you already got 5 faces?

(b) (5%) Suppose you have generated 50000 chains of length 10000, and each chain has a unique starting point and a unique endpoint. Now, you choose another unique starting point and generate a new chain. What is the probability that this chain's endpoint is identical to an existing chain's endpoint? (Let $|P| = 10^{10}$)

Note: Each column uses a different reduction function.

(c) (10%) Implement your own rainbow table and solve the challenge provided in `hw1/RainbowTable`. You can access the server by `nc cns.csie.org 8505`. The size of your table should be smaller than 300 MB (without compression).

Hint: You can use https://tobtu.com/rtcalc.php to design your table.
Note: TA spent 5000 seconds to generate his table using pure Python.

7. Padding Oracle (20%)

Padding oracle is so fun, isn't it? There is a simple padding oracle server, however, only TA can print out the flag. Try to pass through the permission control to get it!
You can access the system by `nc cns.csie.org 8506`, and the server.py is also included in `hw1/cbc`.

Note: messages are hex encoded.

Hint: Find out the iv first.

8. Secret Exchange (10% + Bonus 10%)

Alice and Bob are trying to exchange a secret flag with each other, they decided to use symmetric cryptography to encrypt it.

(a) You can find **alice.py** and **bob.py** in `hw1/secretExchange/naive`, (2%) please draw a sequence diagram to explain how they exchange their secret. (2%) try to find out what the secret message is, and (6%) draw another sequence diagram to describe you attack and explain it.

Note: connect to Alice via `nc cns.csie.org 30001`
Note: connect to Bob via `nc cns.csie.org 30002`

(b) (Bonus) "Stop eavesdropping us, I'm done with you.", said Alice. She manage to get the public key from Bob in a secure way. Now can you break the system again? You can find **alice.py** and **bob.py** in `hw1/secretExchange/dsa`. Try to (Bonus 2%) find the secret message and (Bonus 8%) explain how you break the system in detail.

Note: connect to Alice via `nc cns.csie.org 30003`
Note: connect to Bob via `nc cns.csie.org 30004`
Node: You can write mathematics proof, strange phenomenon you observe, etc. The more detail you give, the more points you will get.

Hint: for sequence diagram, you can use any drawing tools, or some markdown language, reference: https://hackmd.io/@codimd/extra-supported-syntax#Sequence-Diagram