Spring 2020 Cryptography and Network Security

Homework 1

Release Date: 2020/3/17

Due Date: 2020/4/6, 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. Semantic Security (15%)

Let $\mathcal{E} = (E, D)$ be a cipher defined over $(\mathcal{K}, \mathcal{M}, \mathcal{C})$. Assume that one can efficiently generate messages from the message space $\mathcal{M}$ at random. We define an attack game between an adversary $\mathcal{A}$ and a challenger as follows. The adversary selects a message $m \in \mathcal{M}$ and sends m to the challenger. The challenger then computes:

$$b \xleftarrow{R} \{0,1\}, k \xleftarrow{R} \mathcal{K}, m_0 \leftarrow m, m_1 \xleftarrow{R} \mathcal{M}, c \xleftarrow{R} E(k, m_b)$$

and sends the ciphertext c to $\mathcal{A}$, who then computes and outputs a bit $\hat{b}$. That is, the challenger encrypts either m or a random message, depending on b. We define $\mathcal{A}$'s advantage to be $|Pr[\hat{b} = b] - 1/2|$, and we say the $\mathcal{E}$ is *real/random semantically secure* if this advantage is negligible for all efficient adversaries.

Show that $\mathcal{E}$ is real/random semantically secure if and only if it is semantically secure in the ordinary sense.

Note: $x \xleftarrow{R} \mathcal{X}$ denotes the process of assigning to the variable x a random, uniformly distributed element of from the space $\mathcal{X}$.

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 10200`. If this is your first CTF challenge, highly recommend you to solve this challenge first.

5. Find The Secret (10%)

My friends and I built a Shamir's Secret Sharing scheme using a polynomial $A(x) = a_0 + a_1 x + a_2 x^2$, where the secret is a_0. The ith user receives $D_i = (i, A(i) \mod q)$, where q is a prime. I have collected the secret shares from 1st, 2nd and 3rd users (D_1, D_2, D_3). However,

some bad guys forged lots of faked secret shares, trying to prevent us from retrieving our secret. Can you help me find the secret?

My friend gave me some hints to find out the true secret shares, $c_i = g^{a_i} \mod p, 0 \leq i \leq 2$, where p is also a prime and $q|(p-1)$, $g \in Z_p^*$ and g is an element of order q. The data is included in `hw1/secret_sharing`.

6. Cute Baby Cat (20%)

Are you cats lover? There is an organization owns the best cat collection, however, you have to pass through layers of permission control to get them!
You can access the system by `nc cns.csie.org 10202`, and the challenge is also included in `hw1/cbc`.

Note: Each flag worths 5%

Hint: Encoding is not trivial.

7. Resourceful Secret Agent (10% + Bonus 10%)

You are a secret agent. You sneaked into a company to find out evidence of the company secretly selling surgical masks illegally. However, the company uses an encryption system to communicate to each other. Can you find out all evidence? You can access the system by `nc cns.csie.org 10201`. The challenge source is included in `hw1/rsa`.

Note: FLAG1 and FLAG2 are 5% each

Bonus Note: You may take a lot of time to get FLAG3

8. Wired Equivalent Privacy (15% + Bonus 10%)

Alice is a selfish person. She doesn't like to share anything unless you are her close friends. Recently, Alice got a new WiFi and she only gives the password to Bob. You notice that the WiFi uses the WEP Protocol.

(1) (15%) Can you sniff some packets and figure out what they are sharing? There is a secret in their conversation.

(2) (Bonus 10%) Can you find out what the key is?

You can sniff the packet by `nc cns.csie.org 10203`. The challenge is included in `hw1/wep`.

Note: This challenge is a simulation of the WEP Protocol, and there might be some parts not following the original spec of the original protocol.

Hint1: You can assume all the messages (packets) are valid HTTP requests.

Hint2: sha3-256(key) = 8803c3c361025d29b56c7f18074082b61dcb83944fe03c6aac54696ccb85d15e.