Spring 2019 Cryptography and Network Security

Homework 1

Release Date: 3/12/2019

Due Date: 4/7/2019, 23:59

Instruction

- **Submission Guide:** Please submit all your codes and report to CEIBA. 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 110 points including bonus. If you get more than 100 points, your score will still be counted as 100 points.
- 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 `BALSN{...}` 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. Threshold Signature (15%)

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.

Suppose $z = xy$, we call a function e is pairing if it satisfies:

$$e(g^x, g^y) = e(g, g)^{xy} = e(g, g)^z = e(g, g^z)$$

Now consider the following BLS signature scheme:
`Setup:`

$$\begin{aligned} \text{hash function} &: Hash(msg) \\ \text{large prime} &: q \\ \text{generator} &: g \\ \text{secret key} &: sk = x \\ \text{public key} &: pk \equiv g^x (mod\ q) \end{aligned}$$

`Signing:`

$$\begin{aligned} \text{plaintext} &: m \\ \text{hash value} &: h = Hash(m) \\ \text{signature} &: \sigma \equiv h^{sk} (mod\ q) \end{aligned}$$

`Verification:`

$$\text{Valid} : e(\sigma, g) = e(h, g^{sk})$$

You should revise the setting above to accomplish BLS threshold signature, such that a plaintext message m can be signed as a valid threshold signature only if t out of n users collaborate. Everyone should be able to verify the **user signature** and the **threshold signature** by performing the verification procedure, which means you should not split the message before signing as you can't ensure the integrity of splited messages. You should design the BLS threshold signature scheme which include *Setup, User Signature Signing, User Signature Verification, Threshold Signature Signing, Threshold Signature Verification*

procedure. Note that all the users will receive $(i, A(i)\ mod\ p)$ by a secure channel, and you should assume all other communication is done via an authenticated but not secret channel. Your solution should prevent the adversary (including some of the users) to forge a valid signature.

Hint1: $h^a \cdot h^b = h^{a+b}$
Hint2: $(h^a)^b = h^{ab}$
Hint3: Except Threshold Signature Signing, other procedures should similar to BLS signature scheme.

Capture The Flag

4. Babe crypto (10%)

> Welcome to the Crypto World. To prove you are ready for the journey in the Crypto World, please solve all the Classical cipher challenges yourself. Even though Classical cipher only used in the past and most of them can be practically computed and solved, I dont think you can figure it out that easily :P. Be careful and dont use Classical cipher to keep your secret!

You can access the service by `nc 140.112.31.96 10151`. If this is your first CTF challenge, highly recommend you to solve this challenge first.

5. OTP (15%)

(1) (5%) If a one-time pad XORs a message with a random key of same length, it achieves perfect security. Can you decrypt the perfect secret? Access the challenge server by `nc 140.112.31.96 10152`, and the server source code is provided in `otp-1.py` using Python 3.6.7.

(2) (10%) It seems that one-time pad is not secure enough so I use multiple random keys to encrypt my top secret flag. You can access the challenge server by `nc 140.112.31.96 10153`, and the server source code is provided in `otp-2.py` using Python 3.6.7

6. MD5 Collision (10% + Bonus 5%)

Checksum is used to ensure integrity of a given file. However, if the chosen hash function has collision, then the checksum is no longer trustworthy.

The service `nc 140.112.31.96 10150` requires two inputs, which are two base64-encoded python2 code. For instance, this is a python2 code: `print 'Hello'`, the base64-encoded python2 code will be `cHJpbnQgJ0hlbGxvJwo=`. You have to input **two base64-encoded python2 codes** that satisfied the following conditions:

(1) One of the code will output "MD5 is secure!" (without the quotes).

(2) Another code will output "Just kidding!" (without the quotes).

(3) Both codes has the same MD5 hash value.

(4) Of course, two codes should be different.

(5) The code size should be less than 500 bytes.

You will receive the **FLAG** if you succeed. You are required to attach both same MD5 codes in base64 encoding in the report (`code1: <insert base64-encoded code1>, code2: <insert base64-encoded code2>`), otherwise you will not receive any credit.

Bonus (5%): Try to find another hidden flag :) (No hint will be provided for bonus)

7. Flag Market (10% + Bonus 5%)

Welcome to Flag Market, we sell a lot of **Flags** here, you don't want to miss this! Oh, we only accept **BALSN Coin**, so please buy **BALSN Coin** from our cryptocurrency market.

You can access the service by `nc 140.112.31.96 10154`.

Bonus (5%): Try to find another hidden flag :) (No hint will be provided for bonus)

8. RSA (10%)

After learning RSA in class, I think it is a good encryption unless hackers have lots of computation resource. Therefore, I followed the operation in RSA encryption algorithm to keep my flag. You can ask for the cipher as many times as you want, but you will never know how to decrypt it without the private key! Go and find the flag by `nc 140.112.31.96 10155`.

9. The Backdoor of Diffie-Hellman (10%)

> "Hi, I am Alice. I used the script `DH_backdoor/DH.py` to send a top secret flag to my best friend, Bob. However, after sending the flag, I noticed that the generator g was modified by someone. It must be a backdoor of the Diffie Hellman algorithm. I have collected the parameters we used in the file `DH_backdoor/parameters`. Can you find out whether our flag is leaked?"

Decrypt the variable **cipher** in the file `DH_backdoor/parameters` to get the flag.

Hint: $g_{old}^{\frac{p-1}{691829}} \mod p = g_{backdoor}$ *is this simply a coincidence?*