Spring 2018 Cryptography and Network Security

Homework 1

Release Date: 3/20/2018

Due Date: 4/9/2018, 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.
- 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. ElGamal Threshold Decryption (15%)

In this problem, we want you to combine ElGamal public-key encryption and Shamir's secret sharing.
First, 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:`

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

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. How2Crypto (10%)

> "Welcome to the Crypto World. I separated the flag into 6 pieces. If you want the flag, 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, but I don't think you can figure it out that easily :P. Be careful and don't use `Classical cipher` to keep your secret!"

You can access the service by `nc 140.112.31.96 10120`. If this is your first CTF challenge, highly recommend you to solve this challenge first.

5. Mersenne RSA (10%)

In the RSA encryption algorithm, two distinct large prime numbers p and q will be chosen to compute $n = pq$. In order to generate "very large" primes, we implement a *Mersenne Prime Generator* to generate p and q, such that both are large Mersenne primes. You can found that n is a 1128 bits number.

Please try to decrypt the flag. The parameters of RSA are all listed in `mersenne-rsa.txt`.

6. OTP (10%)

If a one-time pad XORs a message with a random key of same length, it achieves perfect security. However, it's impractical since the length of key is required to be the same length of the message.

In order to be more practical, we repeatly append the key to itself. For instance, suppose the key is "*abc*". "*helloworld*" will be encrypted to:

$$(abcabcabcab \oplus helloworld)$$

In this challenge, the plaintext is valid English sentences. It's XORed with the key. Please try to retrieve the flag. The ciphertext can be found in `otp.txt`.

*Hint: The plaintext is valid **English sentences**.*

7. Double AES (10%)

Do you know that encrypting plaintext once is not secure? Therefore, we create a new cipher named Double AES, which encrypts the plaintext twice. Addtionally, the key space is 2^{46}. Only ACM champions are able to break it with efficient brute-force algorithms. There are not many ACM champions in this class, aren't there?

In `2aes.txt`, we provide a pair of plaintext and ciphertext with a secret key. The same key is used to encrypted the flag as well. Please try to decrypt it. The code can be found in `2aes.py`.

Hint: Have you ever heard "Double DES"?

8. Time Machine (10%)

On your way to school, you ran into a strange old man, who claimed himself as a time traveler. He was holding a mysterious metal box, saying that it is a time machine. He said, to turn on the time machine, he needs to do some kind of Proof-of-Resource (PoR). This served as a proof that he comes from the future, since it is not easy to finish such a PoR task with the computing power in 2018.

The task is as follows:

- First, you need to find an input x such that the rightmost 24 bits in $SHA1(x)$ are the same as what the time machine says.
- Second, you need to find an input y such that $SHA1(x) = SHA1(y)$, where $x \neq y$.

Can you help the old man to go back to the future? You can access the challenge by `nc 140.112.31.96 10121`. Please explain how you find x and y. Note that you should send your x and y in `hex-format`.

9. Future Oracle (10%)

After helping the old time traveler, he gave you a USB as a gift and went back to the future. “Use it wisely.”, you heard him say. Opening the program in the USB, you found out that it is a future oracle system, and it can give you hints about what whould happen in the future. To use the future oracle function, you need to login as admin, or the program becomes a super mario game. What the old time traveler left for you was a guest account **guest** and the password **IT'SMEPASSWORD**. Can you login as **admin**?

The login system code is provided (future-oracle.py), with guest account ID and password but no admin's password. You can access the service by `nc 140.112.31.96 10122`. What is the flag? Explain the vulnerability and how you bypass the login system.

Hint: admin password length is smaller than 30

10. Digital Saving Account (15%)

`Digital Saving Account` or `DSA` provides a service for users to check their recent transactions. Everyone can register a new account for free! Since `DSA` is getting more popular, they decide to held a Bug Bounty Program and hope you can report as many vulnerabilities as you can. They provide you the source code `server.py` and you notice that something is suspicious. Can you try to find out what it is?

You can access the service by `nc 140.112.31.96 10123`. Please briefly explain the vulnerabilities.

Hint1: Have you heard about cut-and-paste attack?

Hint2: What is DSA?