Cryptography and Network Security

Lab 1 (Due 2015/4/13 14:00)

- The purpose of this lab is to compensate the lectures and introduce useful security tools through hand-on experiments.
- 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 code and document. Violation of this policy may lead to serious consequences.
- You may need to code some programs in some problems. Since you can use any language you like, we will use pseudo extension name `.ext` when referring the file name.
- Please provide a brief usage of your code. If it is too hard to know how to use your code to get your answer, you may lose some points.
- The coding style and efficiency are parts of the grading criteria. If your code is too ugly or too slow, you may lose some points. In all problems, a reasonable solution should finish in 10 minutes unless you're asked to write a slow problem on purpose.
- If you use any tool, please clearly describe what tool you use, and how you use that tool. For example, if you use a CLI tool, you can show each command and explain each argument; if you use a GUI tool, you can show and explain by screenshots.
- Zip all your files for this homework in `lab1_[student_id].zip` and submit it on CEIBA. For example, `lab1_r03922456.zip`.

1 AES

AES is a commonly used symmetric encryption algorithm. The following ciphertexts are encrypted by AES. Please write a small program to decrypt them. You can use any library that provides the AES function to solve this problem. Since we didn't talk about initialization vector, mode of operation, or padding in the class, if you encounter these parameters in the library of your choice, feel free to ignore initialization vector, use ECB mode and disable padding.

1.1 Decrypt It (10%)

Please decrypt this hex encoded ciphertext `"4e17b6ffd22c19db31a47df663d31763"` by the key `"CNS~CNS!CNS?CNS!"` (without quotes). Save your code for this problem as `code1-1.ext`, and the origin text as `ans1-1.txt`.

1.2 Decrypt It Again (15%)

Please decrypt this hex encoded ciphertext `"18897416b14e8d539a8ffc23490a39b8"`. This time you don't have the key, but know that the key consist of characters in `"CNS"`, the key length is 128 bits, and the origin text is a printable string starts with `"CNS"`. Save your code for this problem as `code1-2.ext`, and the origin text as `ans1-2.txt`.

2 Discrete Logarithm

Given p, g, x, find a such that $g^a \bmod p = x$. We have learned in class that this simple problem is hard to solve, so we make it as your homework. . . ? In this problem, you should code your own program to solve them. Please do not use any powerful external library or any powerful builtin function. If you are not sure about whether you can use a specific function, contact the TAs.

2.1 Easy One (10%)

There are some instances of the discrete logarithm problem in `prob2-1.csv`. Save your program as `code2-1.ext`, and your answer as `ans2-1.csv`.

2.2 Not That Easy One (15%)

There are some instances of the discrete logarithm problem in `prob2-2.csv`. Save your program as `code2-2.ext`, and your answer as `ans2-2.csv`.

3 DoS on Hash Table

You must have learned hash table somewhere, most likely in a data structure class. It is common to use a hash table to store user input, like query strings in URL. But we are just too optimistic and expect it runs in constant time for each operation. Please design an algorithm that generate special testdata, such that our hash tables below would degenerate to $\Theta(n^2)$ for n operations.

3.1 C++ (10%)

Here is a small program `prob3-1` written in C++ that runs fast in most cases. You can check the source code at `prob3-1.cpp`. Please write a program that generate 50000 numbers in $[0, 2^{30}]$, such that `prob3-1` runs slowly. Your evil input should make `prob3-1.cpp` at least 10x slower than random input. Save your program as `code3-1.ext`, and your evil input as `input3-1.txt`.

Here is the environment that will be used to judge the performance of your evil input:

- Ubuntu 14.04.2 LTS (64bit)
- gcc version 4.8.2 (Ubuntu 4.8.2-19ubuntu1)
- Compile with `g++ -std=c++11 prob3-1.cpp -o prob3-1`

3.2 Python (15%)

Here is a small python program `prob3-2.py` that runs fast in most cases. Please write a program that generate 50000 numbers in $[0, 2^{30}]$, such that `prob3-2.py` runs slowly. Your evil input should make `prob3-2.py` at least 10x slower than random input. Save your program as `code3-2.ext`, and your evil input as `input3-2.txt`.

Here is the environment that will be used to judge the performance of your evil input:

- Ubuntu 14.04.2 LTS (64bit)
- Python 2.7.6 (The builtin python version of Ubuntu 14.04)

4 Timing Attack

4.1 Not That Hard One (10%)

Here is a strange program `prob4-1` that requires a password. The censored source code is in `prob4-1.cpp`. Please show us the good password, and how to fix the timing attack vulnerability. Since this problem is not that hard, you're encouraged to get the password without timing attack. You need to fix the vulnerability regardless of how you attack it. Save the fixed code as `fixed4-1.cpp`, and if you have any code for this problem, save it as `code4-1.ext`.

4.2 Hard One (15% + 10%)

Here is a strange program `prob4-2` that requires a password. Please do a timing attack on it and show us the good password. Save your attack script for this problem as `code4-2.ext`.

Since this problem is the hard one, if you can get password without timing attacks, show us and you will get extra bonus up to 10%, but you still need to solve it by timing attack.

Hint

SPOILER ALERT! Here are some keywords that may help you, but reading them may decrease your enjoyment. All hints are encoded by rot13.

1. BcraFFY, ClPelcgb
2. onol-fgrc tvnag-fgrc
3. `qvpgbowrpg.p`
4. cres