CT437 Assignment 2

Using and Benchmarking Block Ciphers with OpenSSL

Overview

In this assignment you will study OpenSSL, the de-facto crypto library for many operating systems and applications. The objectives are as follows:

1. Reinforce your understanding of block ciphers and their operation modes.
2. Deepen your understanding of the OpenSSL API.
3. Benchmark and contrast the performance of three cryptographic algorithms using different configurations.

You are required to write some code in C (or C++) and compile it using the gcc compiler, which is the standard compiler for Linux. You may use the same setup (i.e., Linux VM) as for assignment 1.

General guidelines to configure and call a cryptographic algorithm can be found under EVP Symmetric Encryption and Decryption - OpenSSLWiki
While you can use the source code sample Evp-symmetric-encrypt.c, you must implement a separate header file that contains all function prototypes, macros, #defines, etc. See also **HeaderTemplate.h** in Canvas.

For your benchmarking you must measure CPU Time rather than Wall Time, which can be achieved using the POSIX API. Detailed explanations including examples can be found in 8 Ways to measure execution time in C/C++
All your benchmarking results must be commented on and contrasted in tabular and diagram form.

Problem 1: Block Cipher Benchmarking [15 marks]
Determine and contrast the CPU time of the following encryption settings:

- AES, ARIA and Camellia Algorithm
- 128- and 256-bit key length
- ECB, CBC and CTR mode
- 100 MB and 1000 MB of data
- Encoding and decoding

In your answer comment on the performance difference between all three algorithms, the difference between encoding and decoding for each algorithm, and the impact of the key-length and mode.

Problem 2: Implementing and Benchmarking Triple-DES [5 marks]
Create a separate source code file that benchmarks Triple-DES for CBC, ECB and CTR mode. Compare and comment on the algorithms' performance in both modes when processing 100 MB and 1000MB of data, against your benchmarks in Problem 1.

Assignment Submission

Please submit a zipped folder to Canvas containing:

- Your source code and report for problem 1.
- Your source code and report for problem 2.

Both reports must contain at least two screenshots of running your benchmarks that show the shell with your name in the directory path (as done with assignment 1).

Marking Scheme

- Half the marks will be provided for a well-structured and commented source code that covers all aspects of each problem.
- Half the marks will be provided for a well written report that compares and comments on your findings.

Additional Hints:

1. Note that a summary of all supported algorithms can be found in evp.h
2. You can check your OpenSSL installation via "**openssl version**".
3. Your code can be compiled / linked via:
   "**gcc -o <FileName> <FileName>.c -lcrypto**"
4. Note that the POSIX library (needed for benchmarking) must be linked via the "**-lrt**" switch.
5. Make sure you don't do any file I/O operations while encoding / decoding data, as this will falsify your benchmark reults.
6. The code in the aforementioned example may give some nasty compiler warnings and even crash with a segmentation fault.
   You can fix this problem by changing the pointers *key, *iv, and *plaintext to arrays, e.g.
   unsigned char key[32] = "01234567890123456789012345678901";