CSE 410/565 Spring 2023
Homework 2
Due on March 12, 2023, at 11:59pm
Total: 60 points
You need to submit a pdf generated from LaTeX. If you do not have a local LaTeXsetup, I suggest https://www.overleaf.com/. You need to write the math formulas correctly using LaTeX.
This homework should be finished independently. AI violations are taken seriously.

This exercise will allow you to obtain experience with working with cryptographic libraries. For this purpose, you are to write a program either in python, C, or Java using a standard cryptographic library and your program should run on the UB CSE Fall 2021 virtual machine image (Ubuntu). The VM image can be downloaded from `https://cse.buffalo.edu/~eblanton/misc/vm/`. The image can be used from any operating system using a suitable VM environment software such as, e.g., VirtualBox. Once you start the VM, you can run commands from a terminal and directly write your program in the VM (using an editor such as emacs). When finished, you would need to copy your program to one of CSE student machines for submission.

Implementation and execution tasks

Your program is to call different cryptographic functions and measure the speed of different cryptographic operations. For that purpose, it would need to create or read a small file of size 1KB and a large file of size 10MB. These can be randomly generated data or existing files of any type (of the specified size). The program is to implement and measure the runtime of the following functionalities:

(a) [4 points for 565; 6 for 410] Create a 128-bit AES key, encrypt and decrypt each of the two files using AES in the CBC mode. AES implementations need to be based on hardware implementation of AES, so ensure that your libraries are chosen or configured properly.

(b) [4 points for 565; 6 for 410] Repeat part (a) using AES in the CTR mode.

(c) [4 points for 565; 6 for 410] Repeat part (b) with a 256-bit key.

(d) [4 points for 565; 6 for 410] Create a 2048-bit RSA key, encrypt and decrypt the files above with PKCS #1 v2 padding (at least v2.0, but use v2.2 if available; it may also be called OAEP). This experiment can use a 1MB file for the second file size to reduce the runtime.

(e) [4 points for 565; 6 for 410] Repeat part (d) with a 3072-bit key. This experiment can use a 1MB file for the second file size to reduce the runtime.

(f) [4 points for 565; 6 for 410] Compute a hash of each of the files using hash functions SHA-256, SHA-512, and SHA3-256.

(g) (CSE 565 only; 4 points) Create a 2048-bit DSA key, sign the two files and verify the corresponding signatures. If creating a key takes two parameters, use 224 bits for the exponent size. If the hash function algorithm needs to specified separately, use SHA-256.

(h) (CSE 565 only; 4 points) Repeat part (g) with a 3072-bit DSA key (if the second parameter is required, use 256).

Include simple checking of correctness of your code, namely, that computed ciphertexts decrypt to the original data and that signed messages properly verify. (There is no need to test whether the library functions themselves work correctly.)

Your program will need to measure the following execution times:

1. [Each question 2 points for CSE 565/410] For each encryption experiment (a)–(e), measure (i) the time it take to generate a new key, (ii) the total time it takes to encrypt each of the two files, (iii) the total time it takes to decrypt each file, and also compute (iv) encryption speed per byte for both files and (v) decryption speed per byte for both files.

2. [4 points for CSE 565/410] For each hash function experiment listed in part (f), measure the total time to compute the hash of both files and compute per-byte timings.

3. (CSE 565 only; 4 points) For each signature experiment, measure (i) the key generation time, and (ii) the time to produce a signature for both files, (iii) the time to verify a signature on both of the files, and compute per-byte time for (iv) signing and (v) signature verification for both files.

Make sure that you only measure the operations as specified and not any other operations (such as, e.g., reading or creating files). Also make sure that you retrieve the times using sufficient precision (e.g., in microseconds or nanoseconds); reporting 0 is not valid.

Program interface

Your program needs to be written to produce a single executable that executes all of the experiments above and prints the specified information to standard output for each experiment. It should be non-interactive and not prompt for information such as files to use. Name your program `cryptotools` and make sure that we can execute it by typing `./cryptotools` in your main submission directory. That is, if the source code requires compilation, include a Makefile that compiles the program by typing `make`. If the program cannot be executed in the above form, create a shell script named `cryptotools` that properly invokes your program.

Programming language specific instructions

You would need to choose from one of the programming language and cryptographic library options below.

1. If you choose to use python, the VM should come with all necessary libraries. Make sure that you use `python3`.

2. If you would like to use C, OpenSSL is the cryptographic library to use for this assignment and its commands can either be invoked from a shell or be called from a program. The VM comes with command-line functions, but not programming interfaces. To install the rest of OpenSSL, run the following commands at a shell prompt:

```
sudo apt-get update
sudo apt-get install libssl-dev
```

3. If you choose to use Java, you will need to install the standard Java distribution that comes with a cryptographic library. To install the necessary libraries and programs, run the following commands at a shell prompt:

```
sudo apt-get update
sudo apt-get install default-jdk
```

Performance report

Create a report that you will turn in as your homework. The report needs to include:

I. The chosen programming language and library option above.

II. The timing results that your program measured as specified above.

III. For each performance aspect below, your comments about (i) the expected performance, (ii) whether the observed performance followed the expected performance, and (iii) if there was a difference, your justification of the difference:

1.[2 points for 410/565] how per byte speed changes for different algorithms between small and large files

2.[2 points for 410/565] how encryption and decryption times differ for a given encryption algorithm;

3.[2 points for 410/565] how key generation, encryption, and decryption times differ with the increase in the key size;

4.[2 points for 410/565] how hashing time differs between the algorithms and with increase of the hash size;

5.[2 points for 410/565] how performance of symmetric key encryption (AES), hash functions, and public-key encryption (RSA) compare to each other.

Submission instructions

Submit your report as a PDF document via UBlearns. To submit your code, pack all files in a single ZIP file named `hw2.zip` and submit via UBlearns. It is important that you verify that running unzip hw2.zip correctly unpacks all of your source code. If we are unable to unpack your ZIP file, you risk getting no credit for the assignment.