SYSC 4810: Introduction to Network and Software Security
Module 2 Assignment: Cryptographic Tools

Fall 2020

Dr. J. Jaskolka
Carleton University
Department of Systems and Computer Engineering

Posted: September 20, 2020

Due: October 11, 2020

Due on Sunday, October 11, 2020 by 11:55PM

This assignment contains 13 pages (including this cover page) and 8 problems. You are responsible for ensuring that your copy of the assignment is complete. Bring any discrepancy to the attention of your instructor.

Special Instructions:

1. **Do as many problems as you can.**
2. Start early as this assignment is much more time consuming than you might initially think!
3. The burden of communication is upon you. Solutions not properly explained will not be considered correct. Part of proper communication is the appearance and layout. If we cannot "decode" what you wrote, we cannot grade it as a correct solution.
4. You may consult outside sources, such as textbooks, but *any use* of *any source* **must** be documented in the assignment solutions.
5. You are permitted to discuss *general aspects* of the problem sets with other students in the class, but you must hand in your own copy of the solutions.
6. Your assignment solutions are due by 11:55PM on the due date and must be submitted on cuLearn.
   - Late assignments will be graded with a late penalty of 20% of the full grade per day *up to 48 hours past the deadline.*
7. You are responsible for ensuring that your assignment is submitted correctly and without corruption.

Problem	1	2	3	4	5	6	7	8	Total
Points:	10	10	5	10	5	10	10	10	70

In this assignment, you will participate in activities related to the operation and use of cryptographic tools and techniques. This assignment aims to assess your understanding of the basic principles underlying the main cryptographic concepts and technologies available today, including symmetric and asymmetric encryption, and digital signatures.

Acknowledgment

This assignment is based off the SEED Labs: "Secret-Key Encryption Lab" and "RSA Public-Key Encryption and Signature Lab" developed by Wenliang Du at Syracuse University.

Submission Requirements

Please read the following instructions very carefully and follow them precisely when submitting your assignment!

The following items are required for a complete assignment submission:

1. **PDF Assignment Report**: Submit a detailed report that carefully and concisely describes what you have done and what you have observed. Include appropriate code snippets and listings, as well as screenshots of program outputs and results. You also need to provide an adequate explanation of the observations that are interesting or surprising. You are encouraged to pursue further investigation beyond what is required by the assignment description.
2. **ZIP Archive of Source Code**: In addition to embedding source code listings in your assignment report, create and submit a ZIP archive of all programs that you write for this assignment. Please name each of your source code files with the problem number to which they correspond (e.g., for Problem 7(a), the source code file should be named `Problem7a.c`). Your source code must compile and run, producing the desired output. Also, please remember to provide sufficient comments in your code to describe what it does and why.
3. **ZIP Archive of Screenshot Image Files**: In addition to embedding screenshots of program outputs and results in your assignment report, create and submit a ZIP archive of all of the raw screenshot images that you capture for this assignment.

Grading Notes

An important part of this assignment is following instructions. As such, the following grade **penalties** will be applied for failure to comply with the submission requirements outlined above:

- Failure to submit an Assignment Report will result in a grade of 0 for the assignment.
- Failure to submit the Source Code files will result in deduction of 10% of the full grade of the assignment.
- Failure to submit the Screenshot Image files will result in deduction of 10% of the full grade of the assignment.
- Failure of Source Code to compile/run will result in a grade of 0 for the corresponding problem(s).
- Failure to submit any deliverable in the required format (PDF or ZIP) will result in deduction of 5% of the full grade of the assignment.

Part I Assignment Challenge

1 Introduction

Imagine that you are an employee of a computer security consulting firm. Your consulting firm has recently been approached and contracted by a company called RoadRunner Systems, Inc., which has requested recommendations to help them decide on the specific cryptographic mechanisms to employ in their new systems. You have been assigned as the lead investigator for this contract and are responsbile for preparing a report to fulfill the contractual obligations of your consulting firm with RoadRunner Systems, Inc. The details of these contractual obligations are provided in the sections below.

The different parts of this assignment are designed to guide your investigation into the client's concerns. At the end of the assignment, you will be required to summarize your findings and provide recommendations to RoadRunner Systems, Inc. addressing their concerns.

2 Context

RoadRunner Systems, Inc. specializes in developing roadside computer system units that consist of traffic cameras and sensors to automatically detect a variety of traffic violations such as speeding, failure to stop, illegal turns, etc. as shown in Figure 1. When an traffic violation is identified, a picture of the license plate responsible for the infraction is taken and stored in a central database, along with a message indicating the nature of the detected violation so that appropriate fines can be issued by local traffic authorities to the owners of the infracting vehicles.

Figure 1: Example of a RoadRunner Systems, Inc. roadside unit

Because RoadRunner Systems, Inc. is capturing, transmitting, and storing sensitive license plate information, it requires a cryptographic solution to protect the confidentiality of the contents of the images. However, they are currently undecided of the specific encryption algorithms and associated parameters. Furthermore, RoadRunner Systems, Inc. needs to ensure that the messages that are received from the roadside units are authentic and that they have not been altered in transit. This must be done to avoid any issues related to incorrectly assessing a fine to an innocent vehicle owner. For this purpose, they have considered using digital signatures, but are unsure whether this solution will be suitable for satisfying their security requirements.

RoadRunner Systems, Inc. has expressed the following requirements and constraints of their system, which must be considered in the eventual choices of cryptographic mechanisms.

1. Captured images of license plates must be transmitted to the central database. These transmissions must be encrypted.
2. Messages received by the central database must be authenticated to ensure that they are from a trusted roadside unit and that the contents of the message have not been altered in transit.
3. Because of the limited processing power on the roadside units, the images are transmitted for processing at the central site.
4. Performance is important, as the roadside units must be able to capture and send numerous images and messages to the central database.
5. The systems must be able to operate in all weather conditions including wind, rain, snow, and sun.
6. Key distribution and management is not an issue and will be determined based on the recommendations and eventual selections of the cryptographic mechanisms.

3 Obligations

At the end of this assignment, you will be required to address the following concerns of RoadRunner Systems, Inc.:

1. Provide a recommendation for a cryptographic solution to protect the confidentiality of their data (images of license plates). The recommendation should include a type of encryption, algorithm, key lengths, cipher modes, etc. Your recommendation must be justified with experimental results.
2. Provide a recommendation for message authentication. The client has expressed interest in using digital signatures. You should explain the suitability of this choice by discussing the potential issues with this choice. If you deem the choice of digital signatures to be unsuitable, recommend an alternative solution. In any case, your recommendation must be justified with experimental results, or a detailed discussion of the strengths and limitations of your recommendation.

Part II Environment Setup

This assignment will be conducted using a pre-built virtual machine (VM) image. All of the necessary tools, software, and libraries that are needed for the assignment have been installed on the virtual machine image. We will assume that you already have a virtual machine set up from the Module 1 Assignment. If you have not yet completed the Module 1 Assignment, you will need to do so before continuing with this assignment.

Important Note It is essential that you set up the virtual machine as early as possible to ensure that you have time to address any technical difficulties that you may face. The instructor and the TA will not be able to provide adequate technical support close to the assignment due date.

Part III Symmetric Cryptography

1 Introduction

The most commonly used symmetric encryption algorithms such as DES, 3DES and AES are block ciphers. Block ciphers process plaintext input in fixed-size blocks and produce a block of ciphertext of equal size for each plaintext block. In the case of DES and 3DES, the block length is 64 bits, and for AES the block size is 128 bits. For longer amounts of plaintext, it is necessary to break the plaintext into blocks (padding the last block if necessary). To apply a block cipher in a variety of applications, five modes of operation have been defined and are intended to cover virtually all the possible applications of encryption for which a block cipher could be used. These modes are intended for use with any symmetric block cipher.

In this part of the assignment, you will becomes familiar with fundamental concepts of symmetric encryption, including symmetric encryption algorithms and symmetric block cipher modes using `openssl`.

2 Background

The `openssl enc` command can be used to encrypt and decrypt files.

```
$ openssl enc -ciphertype -e -in plain.txt -out cipher.txt -K key -iv intial_vector
```

- `-ciphertype` stands for the cipher and mode to be used. Examples: `-aes-128-cbc`, `-bf-cbc`, `-aes-128-cfb`, etc.
- `plain.txt` is the input file to be encrypted
- `cipher.txt` is the output file containing the ciphertext resulting from the encryption
- `key` is the key used for encryption (in hexadecimal). Example: `00112233445566778899AABBCCDDEEFF`
- `initial_vector` is the initialization vector to be used (in hexadecimal). Example: `0102030405060708`

Some common options for the `openssl enc` command are provided below:

```
-in <file>        input file
-out <file>       output file
-e                encrypt
-d                decrypt
-K/-iv            key/initialization vector in hexadecimal
-[pP]             print the key/initialization vector (then exit -P)
```

You can find the meaning of the command-line options and all of the supported cipher types by typing `man enc` or `man openssl`.

3 Problems and Tasks

Problem 1 [10 points]

Symmetric Encryption using Different Ciphers and Modes: This experiment will enable you to the behaviour of different ciphers and modes of poperation when using symmetric encryption algorithms. To complete this problem, do the following:

(a) [1 point] **Create a plaintext file**: You can populate the file with any contents you would like. For example, the contents of the file may be: "This is a confidential message intended only for my friend." Do not forget to describe the file and its contents in your report.

(b) [4 points] **Encrypt the file**: Using the `openssl enc` command, encrypt the file that you created. Do this using at least THREE (3) different cipher types but use the same key and initialization vector. Be sure to clearly state the ciphers and the chosen modes that you have selected.

(c) [5 points] **Verify the output**: Once the files are encrypted, most of the data in the file will not be printable. To observe the contents of the output file, use the command-line hex viewing tool `xxd` as follows:

```
$ xxd cipher.txt
```

Explain your findings and clearly identify what you notice about each of the ciphertexts that are generated.

Problem 2 [10 points]

Encryption Mode: ECB vs. CBC: RoadRunner Systems, Inc. needs to store encrypted license plate images in its central database. The files `plate1.bmp` and `plate2.bmp` each contain a sample license plate image and can be downloaded from the assignment resources for this assignment on cuLearn. We would like to encrypt these images, so people without the encryption keys cannot know what is in the image. To complete this problem, do the following:

(a) [2 points] **Encrypt the file**: Using the `openssl enc` command as in Problem 1, encrypt `plate1.bmp` using the ECB (Electronic Code Book) and CBC (Cipher Block Chaining) modes using the AES-128 bit cipher.

(b) [1 point] **Replace the encrypted headers**: We need to treat the encrypted image as an image, and use an image viewing software to display it. However, for the `.bmp` file, the first 54 bytes contain the header information about the image. This header must be set correctly so the encrypted file can be treated as a legitimate `.bmp` file. We will replace the header of the encrypted image with that of the original image. We can use the `bless` hex editor tool (already installed on the virtual machine) to directly modify binary files. We can also use the following commands to get the header from `p1.bmp`, the data from `p2.bmp` (from offset 55 to the end of the file), and then combine the header and data together into a new file as follows:

```
$ head -c 54 p1.bmp > header
$ tail -c +55 p2.bmp > body
$ cat header body > new.bmp
```

(c) [2 points] **View the encrypted images and draw conclusions**: Display the encrypted image using an image viewing program; an image viewer program called `eog` is installed on the virtual machine. Can you derive any useful information about the original image from the encrypted picture? Explain your observations.

(d) [5 points] Repeat Parts (a)-(c) using `plate2.bmp` and report your observations. Do you notice anything different about the results with respect to the original image?

Part IV Asymmetric Cryptography

1 Introduction

RSA (Rivest Shamir Adleman) is one of the first public-key cryptosystems and is widely used for secure communication. The RSA algorithm first generates two large random prime numbers, and then use them to generate public and private key pairs, which can be used to do encryption, decryption, digital signature generation, and digital signature verification. The RSA algorithm is built upon number theories, and it can be quite easily implemented with the support of libraries.

In this part of the assignment, you will gain hands-on experience with asymmetric cryptography and the RSA algorithm by generating public/private keys, performing encryption/decryption, and signature generation/verification. Essentially, you will be implementing the RSA algorithm using the C programming language.

2 Background

The RSA algorithm involves computations on large numbers. These computations cannot be directly conducted using simple arithmetic operators in programs, because those operators can only operate on primitive data types, such as 32-bit integer and 64-bit long integer types. The numbers involved in the RSA algorithms are typically more than 512 bits long. For example, to multiply two 32-bit integer numbers a and b, we just need to use $(a \times b)$ in our program. However, if they are big numbers, we cannot do that anymore; instead, we need to use an algorithm (i.e., a function) to compute their products.

There are several libraries that can perform arithmetic operations on integers of arbitrary size. In this assignment, we will use the Big Number library provided by `openssl`. To use this library, we will define each big number as a `BIGNUM` type, and then use the APIs provided by the library for various operations, such as addition, multiplication, exponentiation, modular operations, etc.

2.1 BIGNUM APIs

All of the big number APIs can be found from https://linux.die.net/man/3/bn. In the following, we describe some of the APIs that are needed for this assignment.

- Some of the library functions require temporary variables. Since dynamic memory allocation to create BIGNUMs is quite expensive when used in conjunction with repeated subroutine calls, a `BN_CTX` structure is created to hold `BIGNUM` temporary variables used by library functions. We need to create such a structure, and pass it to the functions that require it.

```
BN_CTX *ctx = BN_CTX_new();
```

- Initialize a `BIGNUM` variable

```
BIGNUM *a = BN_new();
```

- There are a number of ways to assign a value to a `BIGNUM` variable.

```
// Assign a value from a decimal number string
BN_dec2bn(&a, "12345678901112231223");
// Assign a value from a hex number string
BN_hex2bn(&a, "2A3B4C55FF77889AED3F");
```

```
// Generate a random number of 128 bits
BN_rand(a, 128, 0, 0);
// Generate a random prime number of 128 bits
BN_generate_prime_ex(a, 128, 1, NULL, NULL, NULL);
```

- Print out a big number.

```
void printBN(char *msg, BIGNUM * a) {
    // Convert the BIGNUM to number string
    char * number_str = BN_bn2dec(a);
    // Print out the number string
    printf("%s %s\n", msg, number_str);
    // Free the dynamically allocated memory
    OPENSSL_free(number_str);
}
```

- Compute $res = a - b$ and $res = a + b$:

```
BN_sub(res, a, b);
BN_add(res, a, b);
```

- Compute $res = a \times b$. It should be noted that a `BN_CTX` structure is needed in this API.

```
BN_mul(res, a, b, ctx);
```

- Compute $res = (a \times b) \bmod n$:

```
BN_mod_mul(res, a, b, n, ctx);
```

- Compute $res = a^c \bmod n$:

```
 BN_mod_exp(res, a, c, n, ctx);
```

- Compute modular multiplicative inverse, i.e., given a, find b, such that $(a \times b) \bmod n = 1$.

```
BN_mod_inverse(b, a, n, ctx);
```

2.2 A Complete Example

The following code sample shows a complete example where we initialize three `BIGNUM` variables, a, b, and n. We then compute $(a \times b)$ and $(a^b \bmod n)$.

```
/* bn_sample.c */
#include <stdio.h>
#include <openssl/bn.h>
#define NBITS 256

void printBN(char *msg, BIGNUM * a) {
    // Use BN_bn2hex(a) for hex string
    // Use BN_bn2dec(a) for decimal string
    char * number_str = BN_bn2hex(a);
    printf("%s %s\n", msg, number_str);
    OPENSSL_free(number_str);
}
```

```
int main () {
    BN_CTX *ctx = BN_CTX_new();
    BIGNUM *a = BN_new();
    BIGNUM *b = BN_new();
    BIGNUM *n = BN_new();
    BIGNUM *res = BN_new();

    // Initialize a, b, n
    BN_generate_prime_ex(a, NBITS, 1, NULL, NULL, NULL);
    BN_dec2bn(&b, "273489463796838501848592769467194369268");
    BN_rand(n, NBITS, 0, 0);

    // res = a*b
    BN_mul(res, a, b, ctx);
    printBN("a*b = ", res);

    // res = a^b mod n
    BN_mod_exp(res, a, b, n, ctx);
    printBN("a^b mod n = ", res);
    return 0;
}
```

Compilation

We can use the following command to compile `bn_sample.c`.

```
$ gcc bn_sample.c -lcrypto
```

NOTE: the character after - is the letter ℓ, not the number 1; it tells the compiler to use the `crypto` library.

3 Problems and Tasks

Problem 3 [5 points]

Deriving the Private Key: An important requirement for asymmetric cryptography is that it should be computationally infeasible for an adversary, knowing the public key, to determine the private key. In other words, the private key should remain private. This experiment will demonstrate how to derive the private key given, knowing the public key and having determined the two prime numbers p and q (in some way).

Let p, q, and e be three prime numbers. Let $n = p \times q$. We will use (e, n) as the public key. Write a C program to calculate the private key d. The hexadecimal values of p, q, and e are listed below.

```
p = B5E26E74EB5068C447D660920255A6EF
q = 3122EFFD6FF0246608195B81D3111B0B
e = 34CA9
```

It should be noted that although p and q used in this problem are quite large numbers, they are not large enough to be secure. We intentionally make them small for the sake of simplicity. In practice, these numbers should be at least 512 bits long (the ones used here are only 128 bits). In your report, you should explain why this is the case!

Problem 4 [10 points]

Encrypting a Message: Let (e, n) be the public key. Write a C program to encrypt the message "`Plate: ABCD 123; Failed Stop`" (the quotations are not included). We need to convert this ASCII string to a hexadecimal string, and then convert the hexadecimal string to a `BIGNUM` using the `hex-to-bn` API `BN_hex2bn()`. The following `python` command can be used to convert a plain ASCII string to a hexadecimal string.

```
$ python -c 'print("My ASCII string.".encode("hex"))'
4D79204153434949920737472696E672E
```

The public keys are listed below in hexadecimal. We also provide the private key d to help you verify your encryption result. Be sure to explain how you verified the result.

```
n = 22E929B981FFC200B13E601177E398F1B965B34FD419420DD8DB603B35286145
e = 9EF23
M = Plate: ABCD 123; Failed Stop
d = 07A0CC8B662312154E670D2B767A6E5FE4607BE3215AE3CCD710D1418828D507
```

Problem 5 [5 points]

Decrypting a Message: The public/private keys used in this problem are the same as the ones used in Problem 4. Write a C program to decrypt the following ciphertext C.

```
C = 19C308D172F3E6CB4A9E2D93FC75D662B97C6743FF94BE50CF5F788106C8AA71
```

You will need to convert the result back to a plain ASCII string. The following `python` command can be used to convert a hexadecimal string back to a plain ASCII string.

```
$ python -c 'print("4D79204153434949920737472696E672E".decode("hex"))'
My ASCII string.
```

Problem 6 [10 points]

Signing a Message: The public/private keys used in this problem are the same as the ones used in Problem 4.

(a) [5 points] Write a C program to generate a signature for the following message:

```
M = Plate: LMNO 456; Illegal Turn
```

You should directly sign this message, instead of signing its hash value.

(b) [5 points] Make a slight change to the message M, such as changing 'LMNO 456' to 'LMNO 457', and sign the modified message. Compare both signatures and describe what you observe.

Problem 7 [10 points]

Verifying a Signature: Suppose RoadRunner Systems, Inc.'s central systems receives a message $M =$"`Plate: GOOD 100; Clear Record`" from a roadside unit, with the signature S. We know that roadside unit's public key is (e, n).

(a) [5 points] Write a C program to verify whether or not the signature is indeed that of the roadside unit. The public key and signature (in hexadecimal) are listed below:

```
M = Plate: GOOD 100; Clear Record
S = 174BC865C9023B4978B807A2CF24B15F4F382D20050307E253373C5AAE246ADC
e = 9EF23
n = 22E929B981FFC200B13E601177E398F1B965B34FD419420DD8DB603B35286145
```

(b) [5 points] Suppose that the signature is corrupted, such that the last byte of the signature changes from `DC` to `DD`, i.e, there is only one bit of change. Repeat Part (a) of this problem and describe what happens to the verification process.

Part V Summary of Findings

1 Reminder: Obligations

You are required to address the following concerns of RoadRunner Systems, Inc.:

1. Provide a recommendation for a cryptographic solution to protect the confidentiality of their data (images of license plates). The recommendation should include a type of encryption, algorithm, key lengths, cipher modes, etc. Your recommendation must be justified with experimental results.
2. Provide a recommendation for message authentication. The client has expressed interest in using digital signatures. You should explain the suitability of this choice by discussing the potential issues with this choice. If you deem the choice of digital signatures to be unsuitable, recommend an alternative solution. In any case, your recommendation must be justified with experimental results, or a detailed discussion of the strengths and limitations of your recommendation.

2 Problems and Tasks

Problem 8 [10 points]

Recommendations: Write a summary of your recommendations. Write this summary as if you are going to submit it to RoadRunner Systems, Inc. This means that it should be clear and concise. It should address all concerns outlined in the contractual obligations above and also take into consideration the client's requirements and constraints.

HINT: You may want to refer to specific observations from your experiments obtained in the rest of the problems in this assignment to justify your recommendations.

END OF ASSIGNMENT