UNIVERSITY OF GUELPH
School of Computer Science

CIS*4510 **Computer Security Foundations** **Fall 2024**
Total Marks: 10 **Hassan Khan**

ASSIGNMENT 1

Assignment release date: ~~September 12~~ **September 17**
Assignment due date: **October 08 *3*:00 pm**

Overview

The purpose of this assignment is to introduce you to cryptographic libraries and to understand common pitfalls when using cryptographic building blocks. You will need the OpenSSL library and development package for this assignment (`https://www.openssl.org/`).

Please use CourseLink for all communication. Ask a private question if necessary.

What to submit? Responses to the questions should be provide in a PDF document with name `a1.pdf` (note that renaming a text or word document does not make it a PDF). Source code should be provided as `aes-encrypt.py` and `aes-decrypt.py`. Please make sure that PDF file and the source code files contain your name. zip all files together and name the zipped file as a1_your-first-name_your-last-name.zip. Unzipping should result in three files in the root: a PDF with your responses and two source files. Submit your files on CourseLink under the folder `A1_submissions`

Secret Key Cryptography

1. **Stream Ciphers** [1 mark]

   Alice has designed a mobile app for one-on-one chat among her friends in the class. However, the messages are transmitted over a channel that broadcasts to all devices in the class. Since Alice's friends like to gossip, it is important that only the intended recipient is able to read the messages. Alice has devised the following method to deal with the problem.

   Each user is assigned a 10-character long ID. Each message is at most 256-character long. The structure of messages is provided below:

Structure of the message: [10-character long sender name] [a character separator ':'] [10-character long receiver name] [a character separator ':'] [235 characters for the gossip]
Example message: alicexoxxo:bobko0oo0l: eve likes to snoop

Each message is XORed with a 256-byte long shared key between Alice and Bob. The output ciphertext is the same size as the message (i.e., 42-bytes for the above message from Alice to Bob). The key is changed every day. The app is updated each month over a secure communication channel (not the broadcast medium mentioned above) and it receives the keys for the relevant pair of users (i.e., Alice receives the shared key between Alice-Bob and Alice-Eve).

1.a ~~Use `openssl` CLI to generate 256 random bytes ("the key"). Please provide the command that you used and the generated key in hexadecimal. [0.25 mark]~~

1.b ~~Create a message with your name in gossip text from Alice to Bob (e.g., for the course instructor it will be: "alicexoxox:bobko0oo0l:hassan"). Provide the ciphertext using the key constructed in 1.a and the stream cipher construction discussed above [0.25 mark]~~

1.c Assume you are Eve and you want to obtain the shared key between Alice and Bob. You only have one encrypted message between Alice and Bob, and no information about the gossip in the message. Would you be able to recover the complete or partial key. Also provide the rationale for your answer [0.5 marks]

1.d Assume you are Eve and you want to obtain the shared key between Alice and Bob. You only have two 256-byte encrypted messages between Alice and Bob, and no further information about the gossip in the messages. Would you be able to recover the complete or partial key. Also provide the rationale for your answer [0.5 marks]

1.e ~~Assume you are Mallory and you know the plaintext and ciphertext pairs corresponding to the message "alicexoxox:bobko0oo0l: eve likes to snoop". You want to modify the message such that it reads "alicexoxox:bobko0oo0l: eve likes the calm". Is it possible? If yes, how and if no, why not? [0.5 marks]~~

2. **Block Ciphers** [8 marks]

You are required to implement two programs in Python — `aes-encrypt` and `aes-decrypt`. Each program could take up to five arguments: `key=⟨a file name containing 128-bit key as a hex string⟩`, `IV⟨a file name containing IV as a hex string⟩`, `mode=ecb|cbc`, `in=⟨the input file name⟩`, and `out=⟨the output file name⟩`. For padding, use PKCS7. For this task, you should use Python's `cryptography.hazmat`.

2.a Submit your implementation of `aes-encrypt` and `aes-decrypt`. You will be marked on the correctness of your solution. If you have used any code from the Internet, specify that in the response [3 marks]

2.b Create two messages — one with your first name and second with your last name in gossip texts from Alice to Bob (e.g., for the course instructor these will be: "alicexoxox:bobko0oo0l:hassan" and "alicexoxox:bobko0oo0l:khan"). Use `aes-encrypt` with `mode=ecb` and any random key. Provide the hex dump of first 20 bytes of both ciphertexts. Compare the ciphertexts. Do you see some similarity between the two? Why or why not? [1 mark]

2.c Assume that Alice chooses to use your implementation of `aes-encrypt` and `aes-decrypt` with `mode=cbc` and a counter IV. Does this construction provide confidentiality? Please explain your answer. [1 mark]

2.d Does the construction in 2.c provide integrity? Please explain your answer [1 mark]

2.e Add Galois Counter Mode (GCM) support to your `aes-encrypt` and `aes-decrypt` programs. For GCM, you can reuse `IV`. For GCM encryption and decryption, you should pass an additional parameter using `gcm_arg`, which takes the name of a file, which contains a parameter essential for authenticated encryption. What additional security properties GCM provides? [2 mark]

Public Key Cryptography

3. **RSA** [0.5 marks]

For this part of the assignment, you have been provided an X509 certificate (`F24.cert`) and corresponding key (`F24.key`).

3.a ~~Use `openssl` CLI to see the properties of the certificate? Write down the command that you used and the public key encryption algorithm that is used? [0.5 mark]~~

3.b Use the provided files to encrypt your first name only? Which key did you use and write the `openssl` command that you used for encryption [0.5 mark]

3.c ~~Is 1024-bit RSA safe to use over the Internet? What about 1024-bit Elliptic Curve keys? Why or why not? [1 mark]~~

Integrity and Authentication

4. **Digital Signatures and Hash Functions** [0.5 mark]

For this part of the assignment, use the provided an X509 certificate (`F24.cert`) and corresponding key (`F24.key`).

4.a ~~Your nemesis provides you Assignment 2 files for this course along with SHA-256 hash for it to verify that the assignment files were not modified by them. Should you trust the cryptography and believe them ? Why or why not? [0.5 mark]~~

4.b Use the provided files to hash-then-sign this PDF file. For hash, use SHA-256. Also verify the digital signature that you created. List the commands and the keys that you used along with the outcome [0.5 mark]

4.c ~~The provided files `F24.cert, F24.key` were used for both encryption (in 3.b) and sign-verify functions (in 4.b). What is the problem with this approach? [0.5 mark]~~