Privacy-Preserving and Trustworthy Cyber-Systems CIS 4212/6214 (Spring 2026)

Instructor: **Dr. Attila A. Yavuz**
TA: **Saif E. Nouma**

December 30, 2025

Homework 1 (Due: $January.\ 29^{th}, 2026 - 23:59$)[100 Points + 10 Extra-Credit]

(Upload your solutions to the CANVAS)
Requirements: HW groups are 2-3 people. Each student should contribute all parts of the answers. If one group member drops, the others are equally responsible (each HW can be handled by a single student if needed, so no excuse for missing a partner (if you are doing in group of 2), see syllabus for breaking-up with pair programming partner). Directly borrowing (e.g., copy-paste) from any material and putting in solutions (e.g, from online solutions, Wikipedia, or research papers) is plagiarism (see Syllabus for its corresponding actions). Please cite very carefully each resource you use but citing a solution does not give a license to directly put it as an answer. **All of your answers must be in your own words and interpretations. All your solutions will be passed through MOSS and Turnitin for plagiarism check.**
Note: Many of these questions require you to recall explanations/discussions during the class (this is why attendance is important), and also make research on the internet to find good answers. Please kindly note that very brief and/or informal explanations will potentially be not sufficient.
Remark: HW should be prepared by a text editor (e.g., Microsoft Word or Latex). Handwritten submissions are not accepted. In your answer key, please indicate the point of each question as it is done in this HW document. Preferably, take questions from here, plug them into your answer key with their sub scores, and provide your answer below them. This will facilitate the grading.
As indicated in Syllabus, **no partial credit is possible for coding answers that does not produce correct results with respect to test vectors, deviate from the description and rubric, are coded on platforms other than specified, or do not work on the grader's machine.**

0.1 Programing Assignment – Toy Stream Cipher Implementation [100 pt]

This exercise will put basic cryptographic tools and toy stream cipher that were discussed during the class into action. Specifically, consider the toy stream cipher built from a Pseudo Random Number Generator (PRNG) as described in "Basics.ppt", Slide 11. In this exercise, we will utilize the deterministic PRNG function ChaCha20 from OpenSSL. Moreover, as alluded to in Slide 12, to avoid repeating keys and information leakage, a PRNG must have a "seed".

Alice:

1. Alice reads the message from the "Message.txt" file. The message size must be equal or greater than 32 bytes. (Read the message as unsigned char so that you could use the functions provided in in-class exercises.)

2. Alice reads the shared seed from the "SharedSeed.txt" file. The seed is 32 Bytes (Read the message as unsigned char so that you could use the functions provided in in-class exercises.)

3. Alice generates the secret key from the shared seed based on utilizing the PRNG function from OpenSSL. The key size must match the message length.

4. Alice writes the Hex format of the key in a file named **“Key.txt”**.

5. Alice XORs the message with the secret key to obtain the ciphertext: $(Ciphertext = Message \bigoplus Key)$.

6. Alice writes the Hex format of the ciphertext in a file named **“Ciphertext.txt”**.

7. Once Bob has processed the message, Alice reads Bob’s computed hash from **”Hash.txt”**.

8. If the comparison is successful, Alice can be confident that Bob has received the accurate message. She then writes ”Acknowledgment Successful” in a file called **”Acknowledgment.txt.”** Conversely, if the comparison fails, she records **”Acknowledgment Failed.”**

Bob:

1. Bob reads the ciphertext from the **”Ciphertext.txt”** file.

2. Bob reads the shared seed from the ”SharedSeed.txt” file. The seed is 32 Bytes (Read the message as unsigned char so that you could use the functions provided in in-class exercises.)

3. Bob generates the secret key from the shared seed based on utilizing the PRNG function from OpenSSL. The key size must match the message length.

4. Bob XORs the received ciphertext with the secret key to obtain the plaintext: $(plaintext = ciphertext \bigoplus key)$.

5. Bob writes the decrypted plaintext in a file named **“Plaintext.txt”**.

6. Bob hashes the plaintext via SHA256 and writes the Hex format of the hash in a file named **”Hash.txt”** for Alice to verify.

NOTES:

- The program will be run as two separate executables - ”Alice” and ”Bob” - that communicate through text files.

- In grading at our side, the seed, message, and lengths will be chosen at random and everything must work.

- The code that throws errors or does not successfully compile/run on our side, receives zero credit (**It doesn’t matter if it works on your computer!**).

- You require two files: ”Message.txt” and ”SharedSeed.txt”.

- For the correctness of reading from a file and XORing, please use unsigned char.

- We have provided a script and test files that you could use to test the correctness of your final codes (alice.c and bob.c). Please, only submit your code if it is functional.

- You can check your answer with each provided test file manually or you can use the script to verify your solution for the whole HW1 programming question.

- In order to use the script, just put ”alice.c”, ”bob.c”, the rest of the provide files along with the ”VerifyingYourSolution1.sh” in one folder and run the following command in the terminal:
  bash VerifyingYourSolution1.sh

- Compile your codes with the following commands:
  gcc alice.c -lcrypto -o alice
  gcc bob.c -lcrypto -o bob

- Run your codes for the first test files with the following commands:
  ./alice Message1.txt SharedSeed1.txt
  ./bob SharedSeed1.txt

- **You Must submit 3 files to canvas:**
  alice.c ; bob.c ; HW1Group(GroupNumber).pdf

- Follow the Instructions and all the specific names mentioned above, otherwise there will be 20% deduction for inconvenience. If you followed the above steps and the "secret key", "ciphertext", the "decrypted plaintext", "hash", and the "Acknowledgment" are correct, you will receive full credit.

Rubric:

- The **"Key.txt"** file **[20 pt]**
- The **"Ciphertext.txt"** file **[25 pt]**
- The **"Acknowledgment.txt"** file **[10 pt]**
- The **"Plaintext.txt"** file **[25 pt]**
- The **"Hash.txt"** file **[20 pt]**

Extra-Credit Question [10 points]

0.2 Analysis and Comparison of the Toy Stream Cipher Concept

A. **Potential Vulnerabilities:** In the above toy stream cipher application, assume that Alice uses your software as is, and after encrypting n blocks of data, her computer completely crashes and then re-start just code to encrypt the rest of the data. Does this create a specific vulnerability in the encryption process? Describe an attack that can exploit the current state of the implementation in this event (the answer should have ideally 5-6 sentences, backed up with some basic equations to support the arguments). [5 pt]

B. **Potential Remedies:** Can you suggest at least two ways to mitigate the negative impacts of these potential risks? Your solution should address the root cause of this vulnerability (remember in-class discussions). You can consider an approach only local to the machine, and another that gets benefits from the internet connection. [5 pt]