**CSE 345/545: Foundations of Computer Security**
**Assignment: 1**
**Maximum marks: 100**

**Instructions:**

- Follow the name convention and submission instructions for every question carefully.
- Your file names should follow the convention q0_<Roll no.>.py or q0_<Roll no>_report.pdf. Keep your code efficient, make sure output matches the requirements.
- Post your queries on Google Classroom.
- **Strict plagiarism checks will be conducted for each question. The assignment must be done individually**
- Deadline for submission: 20th September 2023, 11:59 PM

---

## 1. Cryptography — 25 marks

You need to create a program to perform AES encryption and decryption on a 128-bit plaintext using a 128-bit key and follow the AES algorithm with 10 rounds. This means you must implement every step of each of the 10 rounds in AES, including Substitute Bytes, Shift Rows, and so on. You should not rely on existing AES libraries but build these components from scratch in your program.

**a. Verify that the ciphertext, when decrypted text will yield the original plaintext (15 marks)**
**b. Verify that the output of 1st encryption round is the same as the output of the 9th decryption round (5 marks)**
**c. Verify that the output of the 9th encryption round is the same as that of the 1st decryption round (5 marks)**

You have to use the provided **aes.py** file as a template. **0 marks will be provided if the submission file is not in the same format.**

**Deliverable - Submit a python file with naming format : q1_firstname_rollno**

**One Possible Sample Test Case :**

**Input -**

**Input Message : "Two One Nine Two"**
**128-bit Key : "ABCDabcd12344321"**

**Output -**

**Cipher Text: 90f9ff2304c303443dfda3eca651c6c8**
**Decrypted Text: Two One Nine Two**
**Encryption after 1st round:**
**[['0x5d', '0x1a', '0xac', '0x68'], ['0xb1', '0x49', '0x19', '0x38'], ['0xa7', '0x32', '0xf7', '0xde'], ['0x28', '0x00', '0x8f', '0x79']]**
**Encryption after 9th round:**
**[['0xdb', '0x56', '0x40', '0xac'], ['0x9a', '0x11', '0x7a', '0xf7'], ['0xb6', '0xdc', '0x04', '0xa8'], ['0xf6', '0x0f', '0x7c', '0xca']]**
**Decryption after 1st round:**
**[['0xdb', '0x56', '0x40', '0xac'], ['0x9a', '0x11', '0x7a', '0xf7'], ['0xb6', '0xdc', '0x04', '0xa8'], ['0xf6', '0x0f', '0x7c', '0xca']]**
**Decryption after 9 th round:**
**[['0x5d', '0x1a', '0xac', '0x68'], ['0xb1', '0x49', '0x19', '0x38'], ['0xa7', '0x32', '0xf7', '0xde'], ['0x28', '0x00', '0x8f', '0x79']]**

## 2. Access Control Mechanism 20 Marks

You are the system administrator of a Linux server hosting sensitive financial data. Your task is to implement a comprehensive Access Control List (ACL) strategy for a directory named 'financial_data.'

1. Create an ACL and Document **(8 marks)**:
Answer each part with clear documentation and relevant commands, including screenshots
- a. Granting read and write access to the 'accounting' group for 'financial_data' and its contents(3 marks).
- b. Allow only read access for all other groups and users to the 'financial_data' directory and its contents. (2 marks).
- c. Ensure that any new files or directories created within 'financial_data' automatically inherit the ACL settings. (3 marks).

2. Demonstration with Screenshots **(7 marks):**
- a. Demonstrating restricted access for a test user outside the 'accounting' group (2 marks).
- b. Demonstrating read/write access for a user within the 'accounting' group (2 marks).
- c. Demonstrating the automatic inheritance of ACL settings for new files and directories (3 marks).

3. Granting Temporary Write Access **(5 marks)**:
- Documenting the steps to grant temporary write access to a specific subdirectory within 'financial_data' for a user from a new group with clear documentation and relevant commands, including screenshots

## 3. Authentication 25 marks

A “JSON Web Token”, or “JWT”, is a stateless method of authentication that has recently grown in popularity. The JWT string contains the signature of its payload, signed using a private key/secret and following a chosen algorithm – this ensures the token’s integrity. Read more about JWT here.

1. Define a Python function `verifyJwt(token, secret)` that takes in a JWT and a secret as arguments. Validate the token’s signature against the supplied secret. If it is valid, return the decoded payload. Otherwise, throw an exception. The function should implement checks for at least two symmetric algorithms of your choice. For testing, you can demonstrate by generating any jwt token from the jwt debugger in the above link.
   Note: **You are not allowed to use the “PyJWT” library, or any other library implementing JWT verification. (10 Marks)**

2. Refer to this sheet and find your corresponding jwt token. Your task is to retrieve the secret used to sign the JWT. Once retrieved, create a new JWT with the same secret, and payload updated with your roll number and emailID. Document and explain your steps. **(10 Marks)**
3. List down some world use cases of the jwt tokens and suggest some modifications in their architecture so as to improve their security. **(5 Marks)**

## 4. Networking Protocols and Security 30 Marks

### IPv6

1. Define IPv6 and explain why it was developed to replace IPv4. Highlight any key differences between IPv6 and IPv4. **(2 Marks)**

2. Discuss the security benefits and challenges associated with the adoption of IPv6, particularly in comparison to IPv4. **(3 Marks)**

3. You are a network administrator responsible for ensuring IPv6 security. Write a script that scans a network for IPv6-enabled devices and checks if they have any known security vulnerabilities (e.g., open ports or misconfigured settings). Describe the tools and libraries you used in your script and provide a code snippet illustrating how your script detects potential vulnerabilities. **(10 Marks)**

### TCP

1. Write a Python program that establishes a TCP connection with a remote server and sends a simple message "Hello, Server!" to it. Include error handling to deal with potential connection issues and exceptions. **(5 Marks)**

2. Extend your Python program to handle a response from the server. If the server responds with "Hello, Client!", print the response. Otherwise, display an error message. **(4 Marks)**

3. Implement a basic security feature in your program. Before sending the message, calculate its SHA-256 hash and send both the message and the hash to the server. The server should verify the message's integrity by calculating its hash and comparing it to the received hash. If they match, respond with "Message Verified"; otherwise, respond with "Message Tampered." **(6 Marks)**