Security Fundamentals and Development (H7SFD)

BSHC3, BSHC3B

Rohit Verma
Kamil Mahajan

Project Description – Group Submissions

Administrative Data

This project will be marked out of 100. **This project is worth 50% of the final grade awarded**. As published in the School portal, the deadline for submission of the assignment is:

15/11/2024 (Week 9) 11:00 PM

Extension/Re-run

Should any student miss the assessment deadline with a valid reason, the student can apply for an extension/rerun using the Extension/Re-run Form online, via NCI360.

N.B.

All submissions will be electronically screened for evidence of academic misconduct (plagiarism and collusion).

Instructions on AI plagiarism: 1. The use of AI tools for generating written content or code is strictly prohibited. Students must rely on their own knowledge, skills, and research to complete the assessment. 2. The assessment requires students to demonstrate their own original thinking and writing. Any attempt to use AI tools to produce or generate written content will be regarded as a breach of academic honesty and result in severe penalties. 3. If you are using any AI tool for the grammar correction, please acknowledge it in the submission.

1. Project Introduction:

The knowledge and skills gained from this module are expected to contribute to the project. Specifically, this project is designed to assess the following learning outcomes of this module:

#	Learning Outcome Description
LO1	Identify a range of security threats and examine technologies, regulations, standards, and practices to protect individuals and organisations from cyber-attacks.
LO2	Identify threats and formulate responses to mitigate risk through the application of appropriate tools and technologies.
LO4	Demonstrate an in-depth knowledge of cryptographic mechanisms and the ability of applying these mechanisms to the achievement of security services.

2. Project/CA Summary:

The **CA Group Project** for the **Security Fundamentals and Development** module is a collaborative assignment where students work in groups to develop and implement a cryptographic software solution, alongside researching and presenting a high-profile cybersecurity attack or threat.

You are a group of security software developers responsible for designing and developing software systems that are resilient against cyber threats and vulnerabilities. In this project, your group will select a specific use case scenario for cryptography from Table 1. Your project involves utilising your knowledge of coding and encryption to build a robust cryptography software solution. You will also investigate a recent threat/high-profile cyber-attack and prepare a presentation. Each group is limited to minimum 3 and maximum 4 people.

The project has two key components:

1. Group Presentation:
Each group is tasked with researching a specific **cybersecurity attack or threat** and presenting their findings during a scheduled lab session. The presentation must cover the details of the attack, its impact, and the defensive strategies used to mitigate the threat. The **presentation slides** are a mandatory submission alongside the oral presentation. The quality of the slides, the content's depth, and the group's ability to answer questions during the Q&A session will be assessed. The presentation accounts for **20 marks**.

2. Programming Group Project:
The second part of the project involves developing a cryptography-based software solution. These use cases may cover various applications of cryptography, such as secure messaging, secure file storage, and digital signatures. Groups are required to implement cryptographic mechanisms (e.g., encryption, decryption, hashing) and demonstrate the functionality of their application.
The deliverables for this component include: A **detailed project report** outlining the purpose, design, and technical implementation of the software. The report must also include a description of the cryptographic algorithms used and each group member's contribution. A **GitHub link** to the project's source code. A **5-minute demo video** showing the functionality of the application and a walkthrough of the code. This programming project is worth **80 marks**.

3. Submissions required for CA:

The CA submission consists of the following components:

1) **Slides** of group presentation about threats/high-profile cyber-attacks (in **Powerpoint**/**PDF**). The first slide must list the names and student IDs of the group members who contributed to the work. The **filename must include the GroupID**.
2) **Report** for the Cryptography Programming Group Project (in **PDF**). Make sure to acknowledge any original sources of your investigation as appropriate, including the use of AI. You need to **include the cover page,** providing this is applicable. The **filename must include the GroupID.**
3) **Source code** for Cryptography Programming Group Project (GitHub link included in the report template).
4) **Demo video** for Cryptography Programming Group Project (YouTube link included in the report template).

Note: Only one member of the group will submit all the artefacts to Moodle.

4. CA Description:

Group Presentation (20 marks)

1) Check your group number from your Moodle page. Each student has chosen or been allocated a group.
2) Each group must select a topic related to threats/high-profile cybersecurity attacks from the Excel sheet provided or propose a new one for their presentation.
3) The group will investigate the topic and prepare a set of slides, following the guidelines provided.
4) The group will present their topic and participate in Q&A during their corresponding lab slot.
5) Marks for group presentation will be awarded based on presenting the slides at your allocated slot, handling Q&As, and submitting the slides by the given deadline at the specified location. The quality of the content presented, presentation skills, and time management will also be considered.

Group Programming Project (80 marks)

Each group will implement an application that applies concepts of cryptography. A list of potential project descriptions is provided in **Table 1**. You have two options for your project:

1. **Starting from Scratch**:
    - You can choose to build your cryptography application from scratch, following the use case selected from the list.
2. **Extending an Existing Project**:
    - You may extend a project that you have previously worked on. If you choose this option, please note the following:
        - **Original Source**: The report must include details of the original project, such as its purpose, functionality, and existing features.
        - **New Contributions**: Clearly indicate the novel parts or contributions you are making for this assignment.
        - **Source GitHub Repository**:
            - If the original project is already hosted on GitHub, provide a link to the original repository in your report, in addition to the link for the repository containing your new contributions.
            - If the original project is not hosted on GitHub, create a repository for it before extending the project, and include that repository link in your report, along with the link for your new contributions.
            - This ensures that your work is transparent, and your contributions to the project are easily identifiable. You may include a README file in github -

Please note the following guidelines for the code implementation:

- You can use any programming language of your choosing, preferably Java or Python.
- Comment your code as appropriate (e.g., providing explanatory information about a function of the code).
- The application should compile & run, offer the main functionality chosen from Table 1, and offer a clear interface to enter the inputs and see the outputs.
- Each member of the group should be responsible for at least one use case or a distinct feature of the application and its security. This ensures that each member contributes meaningfully to the overall project *(The chosen use case should be significant enough that the member's contribution is clear and measurable)*.

Deliverables for this part:

1. A report: Using the template provided on Moodle, explain the main functionality provided by the application, the algorithms and technical details utilized in the implementation, instructions to download, run and test your application, and illustrate the application's operating process with a flowchart. You must also indicate the contribution of each member of the group **(20 marks)**
2. Source Code: upload your code to Github and include the link in your Report. **(50 marks)**
3. A 5-minute video demonstrating how the application works and a quick walkthrough of the code. Include the link to the video in your report. **(10 marks)**

Table 1 Use Case Scenarios for Cryptography

No.	Use Case Scenario for Cryptography
1	**Secure Instant Messaging App**: Create an application that encrypts messages for secure communication between users.
2	**Secure Email Client**: Develop an email client that uses end-to-end encryption to protect email content.
3	**Secure Voice Calls**: Design a VoIP application that encrypts voice calls to prevent eavesdropping.
4	**Secure Video Conferencing:** Create a video conferencing platform with end-to-end encryption to safeguard video and audio data.
5	**Secure Online Voting System:** Build an online voting system that ensures the confidentiality and integrity of votes through cryptography.
6	**Secure Cloud Storage:** Develop a cloud storage service where files are encrypted before uploading, and only the owner can decrypt them.
7	**Secure Healthcare Records:** Create a system for secure storage and sharing of health records using cryptographic techniques.
8	**Secure Supply Chain Tracking:** Implement a system for supply chain tracking with cryptographic hashing to verify product authenticity.
9	**Secure Document Signing:** Implement a digital signature system (like a document signing platform) that uses digital signatures (using hash functions) to verify the authenticity and integrity of electronic documents or messages.
10	**Secure Digital Identity:** Create a system for secure management and verification of digital identities using cryptography.
11	**Secure E-commerce Transactions:** Build an e-commerce platform with end-to-end encryption for payment transactions.
12	**Secure Exchange of Images:** Create an application that encrypts images to share them in a secure way between 2 parties.
13	**Historical Encryption Schemes:** Create an application that encrypts and decrypts text messages with historical encryption algorithms such as Caesar Cipher, Vigenère Cipher, Rail Fence cipher, etc.
14	**Password Manager:** Create an application that stores users' passwords safely that supports authentication by securely hashing and verifying user passwords to protect user accounts.
15	**File Integrity Verification:** Develop a tool to calculate and verify the hash values of files, ensuring their integrity during transfer or storage
16	**Data Deduplication:** Build a system that uses hashing to identify and eliminate duplicate data, thus optimizing storage resources.
17	**Anti-Virus Scanning:** Build an antivirus program that uses hash values to detect known malware by comparing file hashes with a database of known malicious hashes.
18	**Secure Data Backup:** Design a secure data backup service that encrypts data before storing it in the cloud.
19	**Content-Based File Retrieval:** Design a content-based file retrieval system that uses hashes to index and locate files based on their content
20	**Caching and Data Lookup:** Implement a caching system for a web application that uses hash-based data structures (e.g., hash tables) to optimize data retrieval and storage.

Assessment Criteria and Grading Rubric

Group Presentation

0-39 (Fail)	The presentation lacks structure and coherence. Visual aids (slides, visuals) are either missing or poorly designed. The presentation is difficult to follow or understand. Poor communication skills, such as speaking too fast or too softly. The investigation is incomplete or lacks depth. There is a significant lack of relevant information.
40-49 (Marginal)	The presentation has some structure but lacks clarity. Visual aids are present but may not effectively enhance the presentation. The presentation is somewhat difficult to follow or understand. Communication skills could be improved. The investigation is somewhat complete but lacks depth. Relevant information is missing or insufficient.
50-59 (Adequate)	The presentation is organized but may need improvement in terms of flow. Visual aids are included and reasonably supportive of the content. The presentation is generally clear and understandable. Communication skills are decent but could be more engaging. The investigation is reasonably complete and somewhat in-depth. Most relevant information is included.
60-69 (Good)	The presentation is well-structured and coherent. Visual aids effectively support the content and enhance understanding. The presentation is clear, engaging, and easy to follow. Communication skills are good and engage the audience. The investigation is thorough and detailed. All relevant information is included.
70+ (Excellent)	The presentation is exceptionally well-structured. Visual aids are outstanding, enhancing content comprehension and engagement. The presentation is exceptionally clear, engaging, and highly informative. Exceptional communication skills, such as a strong presence and audience engagement. The investigation is exceptionally thorough, comprehensive, and highly detailed. All relevant information is not only included but also exceptionally well-researched.

Group Programming Project

0-39 (Fail)	The source code is incomplete, non-functional, or severely flawed. Cryptographic algorithms are incorrectly implemented or not used. Code is poorly organized and lacks documentation. The report is missing essential sections. Technical details are missing or incorrect. The flowchart is missing or not related to the application. The report contains multiple grammatical or spelling errors. The demo video is missing or provides no meaningful information. The video lacks clarity and coherence. Key aspects of the application are not demonstrated.
40-49 (Marginal)	The source code is somewhat functional but contains errors or omissions. Cryptographic algorithms are partially implemented with some issues. Code organization and documentation need improvement. The report includes most essential sections but lacks detail. Technical details are somewhat accurate but incomplete. The flowchart is present but lacks clarity or detail. Some grammatical or spelling errors are present in the report. The demo video provides some information but lacks clarity. Key aspects of the application are demonstrated but with gaps or issues.
50-59 (Adequate)	The source code is functional but may contain some minor errors. Cryptographic algorithms are correctly implemented but could be more efficient. Code organization and documentation are reasonable. The report covers all essential sections with adequate detail. Technical details are accurate but may lack depth. The flowchart is clear but could provide more detail. Minor grammatical or spelling errors are present in the report. The demo video provides a reasonable demonstration of key aspects of the application. Clarity and coherence are improved, but there may be some gaps or minor issues.
60-69 (Good)	The source code is functional and well-structured. Cryptographic algorithms are correctly implemented. Code organization and documentation are well-done. The report is comprehensive and well-detailed in all sections. Technical details are accurate and sufficiently explained. The flowchart is clear and illustrative. Minor grammatical or spelling errors are present in the report. The demo video provides a clear and coherent demonstration of all key aspects. Clarity, coherence, and coverage are good, with minimal gaps or issues.

70+ (Excellent)	The source code is exceptional in functionality and structure. Cryptographic algorithms are expertly implemented. Code organization and documentation are exemplary. The report is outstanding, covering all aspects in-depth and with clarity. Technical details are excellently explained. The flowchart is excellent in clarity and detail. The report is free from grammatical or spelling errors. The demo video is exceptional in clarity, coherence, and coverage. It provides an outstanding demonstration of all key aspects. There are no gaps or issues in the video.

Annexure:

Remember the 4 main goals of Cryptography:

- Confidentiality,
- Integrity,
- Authentication,
- Non-Repudiation

The text below was taken from https://www.professormesser.com/security-plus/sy0-501/cryptography-use-cases/

*One of the biggest reasons we use encryption is to ensure our data remains **confidential**. It is a secret and private method of communication that only the intended recipient can see. It's common to use file-level encryption, drive-level encryption, or even encryption over email to maintain this confidentiality.*

*There may be times when we send information to someone, and we want to be sure that the information they receive is exactly what we originally sent. This is called **integrity**, and it prevents someone from modifying the data as it is being transmitted. It's common to use **hashes** to provide this integrity. You would take a hash of the data before sending it, and the recipient would perform the same hashing function on the received data to compare the two hashes. This ensures that nothing has changed during transmission.*

This method is commonly used with file transfers to verify that the transfer was successful. It is also used to store passwords in a way that does not reveal the original password, while still allowing a check to ensure everyone is authenticating properly.

*We can also use cryptography to hide data through a process known as **obfuscation**. Modern malware takes advantage of obfuscation by encrypting data and transferring it onto your system. Since the data is encrypted, it remains hidden from any antivirus scanners that might be on your system. Once the malware executes, it decrypts itself and begins infecting the computer.*

*Cryptography is also commonly used with **authentication**. For instance, we hash passwords to store them on a system for later comparison. Often, we combine passwords with a random **salt** and create a hash of both the salt and the password. This way, even if someone gains access to the hashed password list, the passwords will appear unique, even if some users are sharing the same password.*

*Another useful feature of cryptography is **non-repudiation**. This ensures that any information received from a third party truly came from that third party. By using **digital signatures**, we can provide both integrity and non-repudiation of the data we send.*