VIETNAM NATIONAL UNIVERSITY
HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

COURSE ASSIGNMENT

ADVANCED CRYPTOGRAPHY AND CODING THEORY

Ho Chi Minh City, October 2025

Contents

1 Introduction

This project offers a comprehensive, practical introduction to a subset of the real-world responsibilities performed by information security professionals—particularly those specializing in cryptography. The assignment is divided into three parts. The first part covers **theory**: a cryptography specialist must possess a solid foundational understanding and be able to rigorously demonstrate the correctness of core concepts. The second part bridges **theory and practice**: beyond understanding cryptographic principles, students will design and implement a tool capable of identifying and exploiting weaknesses in weak cipher systems. Such a tool is intended to serve both penetration testers and cryptographers by automating the discovery and exploitation of vulnerable schemes. The third and final part focuses on **exploitation**: you will simulate a man-in-the-middle attack from initial reconnaissance through discovery of a specific vulnerability, and produce a proof-of-concept demonstrating the issue. While these problems do not exhaust the full scope of an information security expert's work, they provide a clear, hands-on view of the tasks involved and are designed to inspire deeper interest in cybersecurity.

2 Research and Development

This section covers the first two parts in the assignments. Each group needs to complete these problems which includes 2 problems divided and scored as below.

Problem	**Description**	**Points**
1	Research	3
2	Development	2

GROUP 1

Problem 1 (Lattices):

Definition 2.1. (Subfields and Extension Fields) A **subfield** K of field L is a subset $K \subseteq L$ that is a field. If K is a subfield of field L, then L is the **extension field** of field K, and we denote L/K (we read "L over K") as a **field extension**.

Definition 2.2. (Minimal Polynomial). Let $\alpha \in K$, where F is a field and L/K is a field extension. A **minimal polynomial** of α is a monic polynomial of lowest degree in $F[x]$ where α is the root.

a) (1 points). Read in [HPS08] (Chapter 6) and present in the report the following terms: *vector spaces, linear combinations, independence, bases, orthogonal and orthonormal basis, lattices, fundamental domains, the Shortest Vector Problem (SVP), the Closest Vector Problem (CVP), the Euclidean ball, the Gaussian expected shortest length, the LLL algorithm.*

b) (1 points). Given a value $\alpha = 21 + \sqrt[4]{11}$, with an approximation β of α to 10 decimal places, find a minimal polynomial $f(x)$ of α given the approximation β by reformulating this into a lattice problem.
(Hint: What is your initial guess of the degree of the minimal polynomial that you want to find? What can we tell about $f(\beta)$ given $f(\alpha) = 0$? Find a suitable lattice basis such that a certain vector in the basis is small and can be efficiently found (You can test your finding by comparing the length of that vector to the Gaussian expected shortest length of the lattice).)

c) (1 points). Suppose that you know the first d-digits after the decimal place of $\sqrt{X}$. Show that you can find X by reformulating this problem into a lattice problem. Students should do b) first to understand the mindset of approaching a lattice problem.

Problem 2 (Attacking the Vigenere Cipher): It is understood that most classical ciphers are now broken, including the **Vigenere Cipher**. In this problem, students need to write an attack script (in Python or any other programming languages) to attack a long ciphertext encrypted using the Vigenere Cipher and explain the core

principles behind the attack. To test the attack, a challenge ciphertext is given here without the secret key, student should show the correct plaintext by attacking the ciphertext. A sample encryption program is also listed here.

GROUP 2

Problem 1 (Lattices):

Definition 2.3. (Subfields and Extension Fields) A **subfield** K of field L is a subset $K \subseteq L$ that is a field. If K is a subfield of field L, then L is the **extension field** of field K, and we denote L/K (we read "L over K") as a **field extension**.

Definition 2.4. (Minimal Polynomial). Let $\alpha \in K$, where F is a field and L/K is a field extension. A **minimal polynomial** of α is a monic polynomial of lowest degree in $F[x]$ where α is the root.

a) (1 point). Read in [HPS08] (Chapter 6) and present in the report the following terms: *vector spaces, linear combinations, independence, bases, orthogonal and orthonormal basis, lattices, fundamental domains, the Shortest Vector Problem (SVP), the Closest Vector Problem (CVP), the Euclidean ball, the Gaussian expected shortest length, the LLL algorithm.*

b) (1 point). Given a value $\alpha = 4 + \sqrt[3]{7}$, with an approximation β of α to 10 decimal places, find a minimal polynomial $f(x)$ of α given the approximation β by reformulating this into a lattice problem.
(Hint: What is your initial guess of the degree of the minimal polynomial that you want to find? What can we tell about $f(\beta)$ given $f(\alpha) = 0$? Find a suitable lattice basis such that a certain vector in the basis is small and can be efficiently found (You can test your finding by comparing the length of that vector to the Gaussian expected shortest length of the lattice).)

c) (1 point). Suppose that you know the first d-digits after the decimal place of $\sqrt{X}$. Show that you can find X by reformulating this problem into a lattice problem. Students should do b) first to understand the mindset of approaching a lattice problem.

Problem 2 (Attacking the Many-Time Pad): It is understood that most classical ciphers are now broken, including the **Multi-Time Pad**. In this problem, students need to write an attack script (in Python or any other programming languages) to attack a bunch of ciphertexts encrypted using the same key in One-Time Pad (hence,

the name Many-Time Pad) and explain the core principles behind the attack. To test the attack, another bunch of ciphertexts are given here without the secret key, student should show either the recovered plaintext, or the secret key by attacking the ciphertexts.

GROUP 3

Problem 1 (Security and Unpredictability of PRGs):

(a) (1 point). Read in [BS15] and clearly define and explain the following concepts in your report: *pseudo-random generator (PRG)*, *PRG security game*, *secure PRG*, *unpredictable PRG game*, and *unpredictable PRG*. If you use additional definitions or variations from other references, include them as well and provide proper citations.

(b) (1 point). Let G be a pseudo-random generator. Prove that if G is **secure**, then G is **unpredictable**. Formally, show that if there exists an adversary $\mathcal{A}$ that wins the *Unpredictable PRG Game* with non-negligible advantage, then there exists a reduction (adversary) $\mathcal{B}$ that wins the *PRG Security Game* with the same advantage as $\mathcal{A}$. Provide a clear construction of $\mathcal{B}$ and explain how the advantage is preserved.

(c) (1 point). Prove the **converse direction**: if G is **unpredictable**, then G is **secure**. Formally, construct an adversary $\mathcal{B}$ that wins the *Unpredictable PRG Game* given any adversary $\mathcal{A}$ that wins the *PRG Security Game*, and describe the relationship between their advantages. Conclude that the two notions, *security* and *unpredictability*, are equivalent for PRGs.

Problem 2 (Attacking the Vigenere Cipher): It is understood that most classical ciphers are now broken, including the **Vigenere Cipher**. In this problem, students need to write an attack script (in Python or any other programming languages) to attack a long ciphertext encrypted using the Vigenere Cipher and explain the core principles behind the attack. To test the attack, a challenge ciphertext is given here without the secret key, student should show the correct plaintext by attacking the ciphertext. A sample encryption program is also listed here.

GROUP 4

Problem 1 (Lattices):

Definition 2.5. (Subfields and Extension Fields) A **subfield** K of field L is a subset $K \subseteq L$ that is a field. If K is a subfield of field L, then L is the **extension field** of field K, and we denote L/K (we read "L over K") as a **field extension**.

Definition 2.6. (Minimal Polynomial). Let $\alpha \in K$, where F is a field and L/K is a field extension. A **minimal polynomial** of α is a monic polynomial of lowest degree in $F[x]$ where α is the root.

a) (1 point). Read in [HPS08] (Chapter 6) and present in the report the following terms: *vector spaces, linear combinations, independence, bases, orthogonal and orthonormal basis, lattices, fundamental domains, the Shortest Vector Problem (SVP), the Closest Vector Problem (CVP), the Euclidean ball, the Gaussian expected shortest length, the LLL algorithm.*

b) (1 point). Given a value $\alpha = 7 + \sqrt[4]{29}$, with an approximation β of α to 10 decimal places, find a minimal polynomial $f(x)$ of α given the approximation β by reformulating this into a lattice problem.
(Hint: What is your initial guess of the degree of the minimal polynomial that you want to find? What can we tell about $f(\beta)$ given $f(\alpha) = 0$? Find a suitable lattice basis such that a certain vector in the basis is small and can be efficiently found (You can test your finding by comparing the length of that vector to the Gaussian expected shortest length of the lattice).)

c) (1 point). Suppose that you know the first d-digits after the decimal place of $\sqrt{X}$. Show that you can find X by reformulating this problem into a lattice problem. Students should do b) first to understand the mindset of approaching a lattice problem.

Problem 2 (Attacking the Vigenere Cipher): It is understood that most classical ciphers are now broken, including the **Vigenere Cipher**. In this problem, students need to write an attack script (in Python or any other programming languages) to attack a long ciphertext encrypted using the Vigenere Cipher and explain the core

principles behind the attack. To test the attack, a challenge ciphertext is given here without the secret key, student should show the correct plaintext by attacking the ciphertext. A sample encryption program is also listed here.

GROUP 5

Problem 1 (Lattices):

Definition 2.7. (Subfields and Extension Fields) A **subfield** K of field L is a subset $K \subseteq L$ that is a field. If K is a subfield of field L, then L is the **extension field** of field K, and we denote L/K (we read "L over K") as a **field extension**.

Definition 2.8. (Minimal Polynomial). Let $\alpha \in K$, where F is a field and L/K is a field extension. A **minimal polynomial** of α is a monic polynomial of lowest degree in $F[x]$ where α is the root.

a) (1 point). Read in [HPS08] (Chapter 6) and present in the report the following terms: *vector spaces, linear combinations, independence, bases, orthogonal and orthonormal basis, lattices, fundamental domains, the Shortest Vector Problem (SVP), the Closest Vector Problem (CVP), the Euclidean ball, the Gaussian expected shortest length, the LLL algorithm.*

b) (1 point). Given a value $\alpha = 3 + \sqrt[3]{23}$, with an approximation β of α to 10 decimal places, find a minimal polynomial $f(x)$ of α given the approximation β by reformulating this into a lattice problem.
(Hint: What is your initial guess of the degree of the minimal polynomial that you want to find? What can we tell about $f(\beta)$ given $f(\alpha) = 0$? Find a suitable lattice basis such that a certain vector in the basis is small and can be efficiently found (You can test your finding by comparing the length of that vector to the Gaussian expected shortest length of the lattice).)

c) (1 point). Suppose that you know the first d-digits after the decimal place of $\sqrt{X}$. Show that you can find X by reformulating this problem into a lattice problem. Students should do b) first to understand the mindset of approaching a lattice problem.

Problem 2 (Attacking the Many-Time Pad): It is understood that most classical ciphers are now broken, including the **Multi-Time Pad**. In this problem, students need to write an attack script (in Python or any other programming languages) to attack a bunch of ciphertexts encrypted using the same key in One-Time Pad (hence,

the name Many-Time Pad) and explain the core principles behind the attack. To test the attack, another bunch of ciphertexts are given here without the secret key, student should show either the recovered plaintext, or the secret key by attacking the ciphertexts.

GROUP 6

Problem 1 (Security and Unpredictability of PRGs):

a) (1 point). Read in [BS15] and clearly define and explain the following concepts in your report: *pseudo-random generator (PRG)*, *PRG security game*, *secure PRG*, *unpredictable PRG game*, and *unpredictable PRG*. If you use additional definitions or variations from other references, include them as well and provide proper citations.

(b) (1 point). Let G be a pseudo-random generator. Prove that if G is **secure**, then G is **unpredictable**. Formally, show that if there exists an adversary $\mathcal{A}$ that wins the *Unpredictable PRG Game* with non-negligible advantage, then there exists a reduction (adversary) $\mathcal{B}$ that wins the *PRG Security Game* with the same advantage as $\mathcal{A}$. Provide a clear construction of $\mathcal{B}$ and explain how the advantage is preserved.

(c) (1 point). Prove the **converse direction**: if G is **unpredictable**, then G is **secure**. Formally, construct an adversary $\mathcal{B}$ that wins the *Unpredictable PRG Game* given any adversary $\mathcal{A}$ that wins the *PRG Security Game*, and describe the relationship between their advantages. Conclude that the two notions, *security* and *unpredictability*, are equivalent for PRGs.

Problem 2 (Attacking the Many-Time Pad): It is understood that most classical ciphers are now broken, including the **Multi-Time Pad**. In this problem, students need to write an attack script (in Python or any other programming languages) to attack a bunch of ciphertexts encrypted using the same key in One-Time Pad (hence, the name Many-Time Pad) and explain the core principles behind the attack. To test the attack, another bunch of ciphertexts are given here without the secret key, student should show either the recovered plaintext, or the secret key by attacking the ciphertexts.

3 SMC Exploitation

Overview

Burp Suite is used as the primary intercepting proxy for this assignment. By acting as a controlled man-in-the-middle, Burp captures every HTTP/HTTPS request and response exchanged between the Android client and the server during the key exchange and messaging phases. Capturing these traces allows us to: (1) inspect fields used in key derivation and authentication, (2) map network messages to source code locations, and (3) craft targeted replay or modification tests to verify and exploit implementation weaknesses.

This assignment guides you through an end-to-end analysis and exploitation of the Secure Messaging Component (SMC) released in this Github link. The goal of this Problem is to apply protocol analysis, client reconstruction, and responsible exploit development to demonstrate practical weaknesses in a tested SMC implementation.

Interception & API Analysis (Use Burp Suite) (0.5 point)

Objective. Build a man-in-the-middle proxy to capture and document every API call exchanged between client and server from the Key Exchange phase through the message exchange phase.

Required tasks

1. Configure Burp Suite as an intercepting proxy for the Android client (real device or emulator). Ensure the client trusts Burp's CA certificate if traffic is TLS-protected.

2. Capture all HTTP/HTTPS requests and responses involved in:

   - Initial key exchange / handshake
   - Session establishment
   - Subsequent encrypted messaging

3. For each captured API call produce:

- A screenshot of the request and response as shown in Burp (include URL, headers, and body).
- A short, precise explanation of every field/parameter in the request and response.

4. Correlate each Burp trace with the repository source files and code paths that implement the API.

Deliverables

- Document containing: Burp screenshots, a table listing each API endpoint, request/response fields, and mapping to code locations.

Re-implementation & Protocol Reconstruction (1 point)

Objective. Act as a client: reconstruct the full protocol flow from scratch and provide a clear, step-by-step pseudocode implementation of the client logic (login → key exchange → secure channel → messaging).

Required tasks

1. Using Burp captures and the source code, enumerate the exact sequence of protocol messages exchanged: message types, ordering, and field semantics.
2. Produce clean pseudocode that shows message construction/parsing, key derivation and use, signature/verification steps, and state transitions.
3. Explain each step of the pseudocode so a reviewer can understand how the secure channel is intended to be built and used.
4. Provide a minimal, runnable client reimplementation in any programming language to demonstrate you can reproduce the protocol behavior. This will be helpful when developing the PoC.

Deliverables

- Document containing: reconstructed message sequence (diagram or ASCII), detailed pseudocode for the client side, and stepwise explanations of each operation.
- Runnable code, attach source file(s) and simple instructions to run them against the provided server.

Exploitation & Proof-of-Concept (3.5 point)

Objective. Based on the analysis from the above Parts, identify and demonstrate a vulnerability in the SMC implementation that impacts one of the following goals depending on your group assignment: recover server secrets, impersonate the server, or leak message-length information.

Group-specific objectives

- **group-1:** Find a server-side weakness that breaks confidentiality. The priority objective is to recover `ServerSecretKey` or otherwise demonstrate the ability to decrypt or reveal plaintext messages.
- **group-2:** Investigate server weaknesses that permit *server impersonation* (i.e., an attacker who either obtains the server's secret signing key) — design, implement, and demonstrate an exploit that causes the client to accept and interact with a forged server endpoint for the assigned user ID.
- **group-3:** Investigate server weaknesses that leak message metadata. For the purpose of this assignment it is sufficient to demonstrate that you can determine the **length** of a message sent by the user. Recovering additional message content is allowed but not required.

User ID distribution

Each group will be assigned a single user ID in this Google Sheet "Assignment". Use only the user ID allocated to your group during testing.

Client environment and demo requirements

- The client is an Android application. You may test using a real Android device or an emulator. **Strongly recommended: run tests on an emulator in an isolated environment.**
- The grading demo must show a live PoC on an **emulator** to ensure consistent evaluation across groups.
- In addition to the live demo, record a backup video (upload to Google Drive or YouTube) and include the link in your report.

Constraints and safety rules (mandatory)

- Limit your testing strictly to the provided test server. Do not attack external systems.
- The test server is free and rate-limited. Do not perform blind brute-force attacks. If brute-force appears necessary, obtain explicit permission from the TA before proceeding.
- Automated scripts must enforce a delay of at least **1 second** between any two consecutive requests. Failure to implement this delay will result in penalties.
- Notify the TAs immediately when you find a working exploit and provide PoC artifacts for verification, so they can validate findings and monitor the test environment.

Deliverables

- Exploit scripts (source) with a README describing how to configure and run them.
- PoC evidence: exploit output (stdout/logs), Burp traces (if relevant), screenshots, and the backup video link.
- A short report that includes: the exploited vulnerability class, high-level steps to reproduce the exploit, evidence that the exploit works (recovered secret / message length / impersonation result), and remediation recommendations (protocol fixes or specific code patches).

4 Conclusions

Academic integrity. All work submitted for this assignment must be your own. Collaboration within your assigned group is allowed and encouraged, but sharing solutions across groups, posting exploit scripts or PoC details publicly in ways that enable misuse prior to grading, or otherwise attempting to gain an unfair advantage is strictly forbidden. Any evidence of cheating, plagiarism, or misuse of the test environment will be handled according to the course academic integrity policy and may result in penalties up to zero credit for the assignment or further disciplinary action. When in doubt, ask the TAs for guidance.

Responsible disclosure and safe testing. All testing must be restricted to the provided test server and the scope of this assignment. Do not target external systems or public services. If you discover a severe vulnerability that could affect other systems or users, notify the TAs immediately and follow responsible-disclosure guidance; do not publish details publicly until authorised.

Communication & support.

- For questions, hints, or clarification, please email the Teaching Assistants at `dangduongminhnhat2003@gmail.com` and `tcthang.sdh242@hcmut.edu.vn`. The TAs will reply within **one business day** (24 hours) wherever possible.
- If you prefer live discussion, you may schedule an online meeting with the TAs during weekday evenings. Please propose times in advance by email and the TAs will confirm availability.

Submission requirements.

1. **Source code:** All source code written for this assignment (client reconstructions, exploit scripts, helper tooling, reproducible scripts) must be placed in a public GitHub repository. The repository must be public at the time of submission and contain a clear `README.md` that describes how to reproduce your PoC, any required configuration, and how your scripts enforce the 1-second request delay constraint. Include any emulator images or configuration instructions required to run the demo.
2. **Report:** Submit a single PDF report that covers all problems. The report must include:

- A concise summary of your objectives and threat model for each part/group.
- For each problem, provide:
  - **Theory:** The theoretical background relevant to your approach (e.g., protocol design, cryptographic primitives, or vulnerability model).
  - **Detailed explanation:** Clear, step-by-step reasoning and methodology used in your analysis or exploit.
  - **Evidence:** Screenshots, logs, captured traffic, or other reproducible proof demonstrating your results.
- Properly formatted citations for all referenced papers, books, web pages, or online resources (use a consistent citation style such as IEEE, ACM, or APA).
- **Team contribution & work breakdown:** A clear table listing every group member and, for each member:
  - specific tasks they were responsible for,
  - what they actually completed,
  - percentage of completion for their assigned tasks,
  - a short note on any pending items or dependencies.

3. **Evidence backup:** Provide a link to one or more short videos (hosted on Google Drive or YouTube with restricted/unlisted visibility) demonstrating your PoC running on an emulator. Include these links in the PDF report and in the GitHub `README.md`.

4. **Submission method:** Upload the PDF report and provide the public GitHub repository link via the course submission system (or email both to the TA if no LMS submission portal is available). Ensure the GitHub link is accessible and the repo is public at the time of submission.

Formatting & citations. The report should be professional and reproducible: include clear step-by-step instructions, configuration parameters, and minimal commands to reproduce the PoC. Cite external sources correctly (paper/book/web) in a consistent style (e.g., IEEE or APA). When quoting or paraphrasing published material, include full bibliographic details.

Deadline. The submission deadline is **two months (60 days) from the assignment release date**. Submit your PDF report and the public GitHub link

by that deadline. Late submissions may be accepted only with prior approval from the instructor/TAs and may be subject to a grade penalty.

Encouragement. This assignment is challenging by design. Please be bold in asking questions and seek help early. The TAs are available to assist and will respond to legitimate requests. Good engineering practices, clear documentation, and responsible testing will be rewarded.

Presentation and demonstration. Each group is required to prepare a concise slide deck summarizing the key findings, methodology, and lessons learned from the project. The slides will be used for the final presentation session, which take place *offline*. During the presentation, each team must perform a clear, reproducible live demonstration (PoC) of their exploit or analysis results on an emulator environment. The presentation should emphasize understanding of the protocol, correctness of the exploit, and safe testing practices.

Final note. Thank you for your hard work and dedication throughout this project. Be confident, stay curious, and collaborate effectively. We wish all teams the very best of success in completing the assignment and delivering an insightful final presentation.

5 Change log

v1.1. Fix Definition of Minimal Polynomial. Add Definition of subfields and extension fields.

References

[BS15] Dan Boneh and Victor Shoup. *A Graduate Course in Applied Cryptography.* 2015.

[HPS08] J. Hoffstein, J. Pipher, and J.H. Silverman. *An Introduction to Mathematical Cryptography.* Undergraduate Texts in Mathematics. Springer, 2008.