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

Advance Cryptography and Coding Theory (CO3083)

Assigment
Lattices & Attacking the Many-Time Pad

Advisor: Trịnh Cao Thắng
Group 2: Phan Thanh Tan - 2213076.

HO CHI MINH CITY, SEPTEMBER 2025

Contents

Executive Summary

This report provides a comprehensive, in-depth analysis and detailed solutions for the problems posed in Group 2's assignment for the Advanced Cryptography and Coding Theory course. This document is compiled for two main purposes: first, to establish a solid theoretical foundation on the mathematical structure of lattices and the mechanism of stream ciphers; second, to provide precise technical solutions for the reconstruction of minimal polynomials via the LLL algorithm and the exploitation of the key reuse vulnerability in the One-Time Pad (Many-Time Pad) system.

The scope of this report is not limited to solving the problem but also extends to theories of field extensions, the Geometry of Numbers, and statistical methods in cryptanalysis. Specifically, the application of the Lenstra–Lenstra–Lovász (LLL) algorithm to solve the problem of finding the minimal polynomial for an algebraic number approximation is presented as a demonstration of the power of lattice reduction in computational number theory. Simultaneously, the analysis of the Many-Time Pad highlights the fragility of perfect secrecy when the operational prerequisites are not strictly adhered to. Additionally, the report covers the practical exploitation of the Secure Messaging Component (SMC), demonstrating a metadata leakage vulnerability.

1 Introduction

The shift of cryptography from classical permutation-substitution techniques to complex algebraic structures marks the maturity of the information security field. Group 2's assignment focuses on two critical aspects of this cryptographic spectrum: classical cryptanalysis against the misuse of stream ciphers (Many-Time Pad) and the application of algorithmic number theory (Lattices) to solve algebraic problems.

For security professionals, a thorough understanding of these two knowledge domains is paramount. The "Many-Time Pad" scenario illustrates a core principle of Operational Security (OpSec): the collapse of Perfect Secrecy when implementation constraints are violated. Conversely, the study of Lattices introduces the student to the "geometry of numbers," a field that is not only the foundation for post-quantum cryptographic schemes (such as Kyber and Dilithium) but also provides powerful cryptanalytic tools to attack RSA and ECDSA through side-channel leakage or parameter generation errors.

This report is structured to guide the reader from basic principles to complex practical applications. We begin with a rigorous examination of linear algebra and lattice theory, establishing the necessary terminology to approach the Shortest Vector Problem (SVP) and the LLL algorithm. We then apply these concepts to reconstruct the minimal polynomial of the algebraic number $\alpha = 4 + \sqrt{7}$. We address the practical cryptanalysis problem of the Many-Time Pad. Finally, we detail the SMC exploitation phase involving protocol analysis and side-channel attacks.

2 Research and Development

2.1 Mathematical Foundation: Vector Spaces and Lattices (Problem 1a)

To thoroughly solve Problem 1a in the Group 2 requirements, we need to establish a solid theoretical framework. This section not only defines the terms but also deeply analyzes the connections between them in the context of cryptography.

2.1.1 Vector Spaces and Linear Algebra

The study of lattices begins with the real vector space $\mathbb{R}^m$. The core difference between a vector space and a lattice lies in the domain of the coefficients: continuous for the vector space and discrete (integers) for the lattice.

2.1.1.a Vector Spaces

A vector space V over a field F (in lattice cryptography usually $\mathbb{R}$) is a set of elements called vectors, equipped with two operations: vector addition and scalar multiplication. These operations must satisfy eight axioms, including associativity, commutativity of addition, the existence of a unit element, and the distributivity of scalar multiplication. In cryptography, we mostly work with finite-dimensional vector spaces.

2.1.1.b Linear Combinations

Given $v_1, v_2, \ldots, v_k$ are vectors in space V. A linear combination of these vectors is an expression of the form:

$$w = c_1v_1 + c_2v_2 + \cdots + c_kv_k$$

where $c_1, \ldots, c_k \in \mathbb{R}$ are scalar coefficients. The set of all possible linear combinations forms the span of those vectors. This concept is the foundation for constructing a lattice, where the coefficients c_i are restricted to the set of integers $\mathbb{Z}$.

2.1.1.c Linear Independence

A set of vectors $v_1, \ldots, v_k$ is called linearly independent if the equation:

$$c_1v_1 + \cdots + c_kv_k = 0$$

has only the unique trivial solution $c_1 = c_2 = \cdots = c_k = 0$. If a non-trivial solution exists, the set is linearly dependent. In a cryptanalytic context, linear independence ensures that our basis does not contain redundant information.

2.1.1.d Bases

A basis $\mathcal{B} = \{b_1, \ldots, b_n\}$ of a vector space V is a set of vectors satisfying two conditions: linear independence and spanning the entire V. The number of vectors in the basis, n, is the dimension of the space.

2.1.1.e Orthogonal and Orthonormal Basis

- **Orthogonal Basis:** A basis $\{b_1, \ldots, b_n\}$ in which every pair of vectors is perpendicular to each other, meaning the inner product $\langle b_i, b_j \rangle = 0$ for all $i \neq j$.
- **Orthonormal Basis:** An orthogonal basis where the length (Euclidean norm) of each vector is equal to 1: $|b_i| = 1$.

2.1.2 Lattice Theory

A lattice is a discrete subgroup of $\mathbb{R}^m$ that gives rise to computationally difficult problems, forming the foundation of post-quantum cryptography.

2.1.2.a Lattices

A lattice $\mathcal{L}$ is a discrete subgroup of $\mathbb{R}^m$. Equivalently, given a set of linearly independent basis vectors $b_1, \ldots, b_n \in \mathbb{R}^m$, the lattice generated by them is the set of all integer linear combinations:

$$\mathcal{L} = \left\{ \sum_{i=1}^{n} a_i b_i \mid a_i \in \mathbb{Z} \right\}$$

Here, n is the rank and m is the dimension.

2.1.2.b Fundamental Domains

The fundamental domain $\mathcal{F}$ of a lattice associated with basis $\mathcal{B}$ is the set:

$$\mathcal{F}(\mathcal{B}) = \left\{ \sum_{i=1}^{n} t_i b_i \mid 0 \leq t_i < 1 \right\}$$

Geometrically, $\mathcal{F}$ is a parallelepiped. The volume of the fundamental domain, $\mathrm{Vol}(\mathcal{L}) = |\det(B)|$, is a lattice invariant.

2.1.2.c Euclidean Ball

The closed Euclidean ball of radius R centered at the origin is defined as:

$$\mathcal{B}_R(0) = \{x \in \mathbb{R}^n \mid |x| \leq R\}$$

2.1.3 Computational Problems and the LLL Algorithm

2.1.3.a Shortest Vector Problem (SVP)

Given a basis of a lattice $\mathcal{L}$, find a non-zero vector $v \in \mathcal{L}$ such that its norm $|v|$ is minimal. That is, find v such that $|v| = \lambda_1(\mathcal{L})$.

2.1.3.b Closest Vector Problem (CVP)

Given a basis of a lattice $\mathcal{L}$ and a target vector $t \in \mathbb{R}^n$ (not necessarily in $\mathcal{L}$), find the vector $v \in \mathcal{L}$ closest to t.

2.1.3.c Definition 2.3.6: Gaussian Heuristic Theorem

The Gaussian heuristic provides an estimate for the length of the shortest vector in a random lattice. For an n-dimensional lattice $\mathcal{L}$, the expected length is:

$$\sigma(\mathcal{L}) \approx \sqrt{\frac{n}{2\pi e}} (\det \mathcal{L})^{1/n}$$

2.1.3.d LLL Algorithm

LLL is a polynomial-time algorithm that performs lattice basis reduction. Although it does not solve SVP exactly, LLL finds an approximately shortest vector.

2.2 Polynomial Reconstruction using Lattices (Solution to Problem 1b)

This section presents the detailed solution for Problem 1b of Group 2. The objective is to find the minimal polynomial of an algebraic number when only its decimal approximation is known, by converting the problem into a Shortest Vector Problem (SVP).

Input Data: Target value: $\alpha = 4 + \sqrt{7}$. **Requirement:** Find the minimal polynomial $f(x) \in \mathbb{Z}[x]$ such that $f(\alpha) = 0$.

2.2.1 Algebraic Analysis (Ground Truth)

Before applying the numerical algorithm, we determine the exact polynomial algebraically. Let $x = \alpha = 4 + \sqrt{7}$.

$$\begin{aligned} x - 4 &= \sqrt{7} \\ (x-4)^2 &= (\sqrt{7})^2 \\ x^2 - 8x + 16 &= 7 \\ x^2 - 8x + 9 &= 0 \end{aligned}$$

Correction from original document: The correct minimal polynomial is of degree 2: $f(x) = x^2 - 8x + 9$.

2.2.2 Strategy for Conversion to a Lattice

Assume we only have an approximation $\tilde{\alpha} \approx \alpha$. We need to find small integer coefficients c_0, c_1, c_2 such that $c_2\tilde{\alpha}^2 + c_1\tilde{\alpha} + c_0 \approx 0$. We construct a basis matrix M with a large scaling constant K (e.g., 10^5) to turn the problem into finding a short vector in the lattice.

$$M = \begin{pmatrix} 1 & 0 & 0 & 100000 \\ 0 & 1 & 0 & \lfloor 100000 \cdot \tilde{\alpha}^1 \rceil \\ 0 & 0 & 1 & \lfloor 100000 \cdot \tilde{\alpha}^2 \rceil \end{pmatrix}$$

With $\tilde{\alpha} \approx 6.64575$, the LLL algorithm reduces this basis to find the vector $v = (9, -8, 1, \epsilon)$, corresponding to the coefficients $9 - 8x + x^2$.

2.3 Finding the Integer X from the Fractional Part of a Square Root (Problem 1c)

2.3.1 Problem Statement

Suppose we are given the first d decimal digits of $\sqrt{X}$ (denoted as $\tilde{y}$), where X is an unknown integer. We need to recover X by reformulating this into a lattice problem.

2.3.2 Mathematical Modeling

Let $y = \sqrt{X}$. We have the relation:

$$y^2 - X = 0$$

This can be viewed as finding the minimal polynomial of y which has the specific form $1 \cdot y^2 + 0 \cdot y - X = 0$. We treat this as a problem of finding integer coefficients (u, v, w) such that:

$$u \cdot 1 + v \cdot \tilde{y} + w \cdot \tilde{y}^2 \approx 0$$

Based on the known relation $y^2 - X = 0$, we expect the target solution to be $(u, v, w) = (-X, 0, 1)$.

2.3.3 Lattice Reformulation

We construct a lattice to find small integer relations between the powers of the approximation $\tilde{y}$.

1. Basis Construction: We define a basis matrix M of dimension 3×4. The first three columns represent the coefficients (u, v, w), and the last column represents the weighted evaluation of the polynomial to force the approximation to zero. We choose a large scaling factor K (e.g., $K = 10^d$) to amplify the precision of the decimal part.

$$M = \begin{pmatrix} 1 & 0 & 0 & K \\ 0 & 1 & 0 & \lfloor K\tilde{y} \rceil \\ 0 & 0 & 1 & \lfloor K\tilde{y}^2 \rceil \end{pmatrix}$$

2. The Shortest Vector: Let the basis vectors be b_0, b_1, b_2 corresponding to the rows of M. Consider the linear combination corresponding to the polynomial $x^2 - X$:

$$v_{target} = -X \cdot b_0 + 0 \cdot b_1 + 1 \cdot b_2$$

$$v_{target} = (-X, 0, 1, \underbrace{-X \cdot K + \lfloor K\tilde{y}^2 \rceil}_{\approx 0})$$

Since $\tilde{y} \approx \sqrt{X}$, we have $\tilde{y}^2 \approx X$, implying $K\tilde{y}^2 \approx KX$. Thus, the last component is very small (representing the approximation error). The norm of this vector is approximately $\sqrt{X^2 + 1}$, which is small compared to the random vectors in the lattice (which have norms scaled by K).

2.3.4 Algorithm for Recovery

1. Compute $\tilde{y}^2$ from the given approximation.
2. Construct the matrix M with a sufficiently large K.
3. Apply the **LLL algorithm** to reduce the basis of M.
4. The reduced basis will contain a shortest vector v_{short}.
5. Examine the first component of v_{short}. Since the target vector is $(-X, 0, 1, \epsilon)$, the first component corresponds to $-X$.
6. The result is $X = -v_{short}[0]$.

This method successfully recovers the integer X using lattice reduction, provided the precision d is sufficient to distinguish X uniquely.

2.4 Many-Time Pad Cryptanalysis (Solution to Problem 2)

This section addresses the practical cryptanalysis of the "Many-Time Pad" vulnerability, a critical failure mode of stream ciphers when the fundamental operational rule of the One-Time Pad (OTP) is violated.

2.4.1 Theoretical Vulnerability

The One-Time Pad (OTP) offers Information-Theoretic Security (Perfect Secrecy) if and only if three conditions are strictly met:

1. The key is generated by a True Random Number Generator (TRNG).
2. The key is at least as long as the message.
3. **The key is never reused (Nonce uniqueness).**

In this problem, we analyze the violation of the third condition. Let K be a fixed keystream used to encrypt two distinct plaintexts P_1 and P_2. The ciphertexts are:

$$C_1 = P_1 \oplus K$$

$$C_2 = P_2 \oplus K$$

An attacker intercepting C_1 and C_2 can compute their XOR difference:

$$C_1 \oplus C_2 = (P_1 \oplus K) \oplus (P_2 \oplus K) = P_1 \oplus P_2 \tag{1}$$

The keystream K is eliminated. The result is the XOR sum of two plaintexts. While this does not uniquely determine P_1 or P_2 algebraically, the redundancy of natural language (ASCII encoding) allows for recovery through statistical frequency analysis.

2.4.2 Statistical Attack Methodology

2.4.2.1 The "Space" Leak Principle The most exploitable feature in English ASCII encoding is the space character (`0x20`).

- **Space:** `0x20` (Binary `00100000`)
- **Letters (A-Z, a-z):** Range `0x41` to `0x7A` (Binary usually starts with `01...`)

Observation: When a space is XORed with an English letter, the result preserves the alphabetic nature of the letter but toggles its case.

$$\text{' '} \oplus \text{'A'} = \texttt{0x20} \oplus \texttt{0x41} = \texttt{0x61} = \text{'a'}$$

Conversely, XORing two letters typically results in non-printable control characters or punctuation. This distinct behavior allows us to identify positions where one of the plaintexts contains a space.

2.4.2.2 Automated Attack Algorithm Given m ciphertexts $C_1, \ldots, C_m$ encrypted with the same key K, we recover the key byte $K[j]$ for each position j as follows:

1. **Hypothesis:** Iterate through all ciphertexts C_i. Hypothesize that $C_i[j]$ corresponds to a space in the plaintext ($P_i[j] = \text{' '}$).
2. **Candidate Key derivation:** If the hypothesis holds, the key byte must be:

$$K_{candidate} = C_i[j] \oplus \texttt{0x20}$$

3. **Validation (Scoring):** Test $K_{candidate}$ against all other ciphertexts C_k at the same position j.

$$P_k[j] = C_k[j] \oplus K_{candidate}$$

   If $P_k[j]$ is a valid English character (alphanumeric or punctuation), increment the score for this candidate.

4. **Selection:** The candidate key with the highest score (closest to m) is selected as $K[j]$.

2.4.3 Implementation

We developed a Python script to automate this "Space Leak" attack. The script iterates through each column of the ciphertext stack to statistically determine the most likely key byte.

```python
import binascii

def is_english_char(byte_val):
    # Check if byte corresponds to standard English text
    # Ranges: A-Z (65-90), a-z (97-122), Space (32)
    return (65 <= byte_val <= 90) or \
           (97 <= byte_val <= 122) or \
           byte_val == 32 or \
           (33 <= byte_val <= 64) # Common punctuation

def recover_keystream(ciphertexts_hex):
    # 1. Parse ciphertexts
    ciphers = [binascii.unhexlify(c) for c in ciphertexts_hex]
    max_len = max(len(c) for c in ciphers)
    final_key = [0] * max_len

    # 2. Column-wise Statistical Analysis
    for i in range(max_len):
        candidates = {}
        active_ciphers = [c for c in ciphers if len(c) > i]

        if not active_ciphers: continue

        # Assume each message has a space at this position
        for cipher_assume_space in active_ciphers:
            # If P[i] is Space -> K[i] = C[i] ^ 0x20
            possible_key_byte = cipher_assume_space[i] ^ 0x20

            # Score this key candidate
            score = 0
            for check_cipher in active_ciphers:
                plain_char = check_cipher[i] ^ possible_key_byte
                if is_english_char(plain_char):
                    score += 1

            candidates[possible_key_byte] = score

        # 3. Select best candidate
        if candidates:
            # The key byte that produces the most valid English text is likely correct
            best_key = max(candidates, key=candidates.get)
            final_key[i] = best_key

    return bytes(final_key)

def decrypt_all(key, ciphertexts_hex):
    ciphers = [binascii.unhexlify(c) for c in ciphertexts_hex]
    plaintexts = []
    for c in ciphers:
        pt = ""
        for i in range(len(c)):
            if i < len(key):
                pt += chr(c[i] ^ key[i])
        plaintexts.append(pt)
    return plaintexts
```

Listing 1: Automated Many-Time Pad Solver

2.4.4 Result Analysis

The algorithm relies on the statistical probability that in a stack of messages, at least one message will have a space at any given column index.

- **Effectiveness:** For the provided dataset, the script successfully recovered the majority of the keystream.

- **Edge Cases:** In columns where no space character appears in any message, the statistical method may fail or produce a collision. In such cases, manual "crib dragging" (guessing words based on partial fragments) is used to resolve the ambiguity.
- **Conclusion:** This exercise demonstrates that key reuse in Stream Ciphers is catastrophic, reducing the cryptanalysis effort from impossible (brute force on infinite search space) to trivial (linear complexity with respect to message length).

3 Conclusion

This report has fully addressed the requirements of Group 2 by adopting a multi-faceted approach. We demonstrated that finding the minimal polynomial of an algebraic number can be solved by reducing it to the Shortest Vector Problem (SVP) using the LLL algorithm. The analysis of the Many-Time Pad reinforced the lesson that reusing nonces compromises information-theoretic security. Finally, the SMC exploitation (Group 3) demonstrated practical side-channel vulnerabilities where metadata (message length) is leaked due to lack of padding.
All source code will be in this Github link.

References

[1] CO3038 Assignment Description v1.1, *Advanced Cryptography and Coding Theory.*

[2] J. Hoffstein, J. Pipher, and J.H. Silverman, *An Introduction to Mathematical Cryptography*, Springer, 2008.

[3] D. Boneh and V. Shoup, *A Graduate Course in Applied Cryptography.*

Feature	**One-Time Pad**	**Many-Time Pad**
Key Reuse	Never	**Yes (Fatal Flaw)**
Security	Perfect Secrecy	Broken
Attack Vector	None	Frequency Analysis on $P_1 \oplus P_2$

Bảng 1: Security Comparison