Lab 6: Implementing Hill Cipher

Information Security and Cryptography (AI331)

Objective

The objective of this lab is to understand and implement the Hill Cipher, a polygraphic substitution cipher based on linear algebra. Students will learn how to perform encryption and decryption using modular arithmetic over matrices.

Background

The Hill cipher was invented by Lester S. Hill in 1929. It uses linear algebra to encrypt blocks of plaintext.

Encryption

Let the plaintext be represented as an n-dimensional vector P over Z_{26}, where $A = 0, B = 1, . . . , Z = 25$. The key is an invertible $n \times n$ matrix K over Z_{26}. Encryption is performed as:

$$C \equiv P \cdot K \pmod{26}$$

where C is the ciphertext vector.

Decryption

Decryption requires the inverse of the key matrix modulo

$$26: P \equiv C \cdot K^{-1} \pmod{26}$$

The matrix K must be chosen such that det(K) and 26 are coprime (so that K^{-1}exists in Z_{26}).

Tasks

1. Convert the given plaintext (only uppercase letters without spaces/punc tuation) into numeric form using $A = 0, B = 1, . . . , Z =$ 25.

2. Implement encryption using a given key matrix K.

3. Implement decryption by computing K^{-1}(mod 26) and applying it to the ciphertext.

4. Test your implementation on the following:

   - Key matrix K = 3 3 2 5

   - Plaintext = “HELP”

   Show the intermediate steps (numerical representation, block division, encryption, ciphertext, decryption).

5. Generalize your code to handle any n × n Hill cipher where $n = 2$ or 3.

Programming Guidelines

- You may use C++ or Python.
- Your program should read plaintext and key matrix from input.
- Write modular arithmetic and matrix inverse computations explicitly (do not use external cryptographic libraries).

Starter Code (C++ Skeleton)

Below is a skeleton code to help you get started. You need to complete the missing parts such as modular inverse and matrix multiplication.

```
#include <bits/stdc++.h>
using namespace std;

int modInverse(int a, int m) {
        // TODO: Implement Extended Euclidean Algorithm
}

vector<vector<int>> matrixInverse(vector<vector<int>> K, int mod ) {
        // TODO: Implement modular matrix inverse
}

vector<int> encryptBlock(vector<int> P, vector<vector<int>> K, int mod) {
        // TODO: Implement C = P * K mod 26
}

vector<int> decryptBlock(vector<int> C, vector<vector<int>> Kinv , int mod) {
        // TODO: Implement P = C * Kinv mod 26
}

int main() {
        string plaintext = "HELP";
        vector<vector<int>> K = {{3,3},{2,5}};
        int mod = 26;

        // TODO: Preprocess plaintext, encrypt and decrypt }
```

Submission

Submit the following:

- Source code file.
- A text file showing encryption and decryption of the given example.
- A short report (1-2 pages) explaining how you implemented matrix inverse modulo 26.