南京郵電大學

《密码学》实验报告（一）

（ 2024 / 2025　学年 第 二 学期）

题　　目：转轮机的模拟实现

专　　　业	信息安全
学 号 姓 名	B230410
	B23041011
	于明宏
指　导　教　师	李琦
指　导　单　位	计算机学院、软件学院、网络空间安全学院
日　　　期	2025.3.8

转轮机的模拟实现

一、课题内容和要求

本实验的目标是实现一个简单的转轮机加密系统，以模拟经典的恩尼格玛（Enigma）机的基本工作原理。系统应具备：

1. 基本加密功能：输入明文（密文）后，经过转轮机加密（解密）后输出密文（明文）；
2. 转轮逻辑实现：采用多个转轮，输入字符后进行对应的旋转；
3. 反射器机制：通过反射器使得加密和解密使用同一个出入口；
4. 插线板功能：允许对字符进行单表代换；
5. 可调节初始位置：允许用户设定转轮初始位置。

二、实现分析

本系统的整体功能框架如下：

1. 转轮（Wheel）：通过映射不同的字符进行加密，并在每次输入后进行旋转。
2. 反射器（Reflector）：将信号反射回转轮，使其经过不同路径返回。
3. 插线板（Plugboard）：用于交换输入和输出的字符，提高加密强度。
4. 用户交互：用户可以设定转轮初始位置、输入待加密文本，并得到加密后的输出。

加密（解密）流程：

1. 输入字符 → 插线板转换 → 经过三层转轮加密（解密） → 反射器反射 → 逆向通过三层转轮 → 插线板转换 → 输出密文（明文）
2. 每次输入一个字符，转轮按一定规则旋转。

三、概要设计

1. 主要存储结构

系统采用数组和映射表实现字符映射转换：

```
class Wheel {
public:
    int left[26];   // 左侧字符映射
    int right[26]; // 右侧字符映射
};
class Reflector {
public:
    int reflect[26]; //  反射映射
};
class Plugboard {
public:
    map<char, char> wiring; //  插线板映射
};
```

2. 主要函数流程

（1）转轮旋转机制

通过 turn()方法，使得转轮每次加密（解密）后旋转。

（2）字符加密流程

通过 getIndex()和 getOriginalIndex()方法，获取字符在转轮中的映射位置。

（3）反射器功能

通过 reflectIndex()方法对字符进行反射处理。

（4）插线板功能

通过 setPlugboard()设定插线板映射，并用 swap()方法交换字符。

四、源程序代码

```
#include <iostream>
#include <string>
#include <cctype>
#include <map>
using namespace std;

class Wheel {
public:
    int left[26];
    int right[26];

    Wheel(int a[], int b[]) {
        for (int i = 0; i < 26; i++) {
            left[i] = a[i];
            right[i] = b[i];
        }
    }

    void turn() {
        int temp1 = left[25];
        int temp2 = right[25];
        for (int i = 25; i > 0; i--) {
            left[i] = left[i - 1];
            right[i] = right[i - 1];
        }
        left[0] = temp1;
        right[0] = temp2;
    }

    int getIndex(int x) {
        for (int i = 0; i < 26; i++) {
            if (left[x] == right[i]) {
                return i;
            }
        }
        return -1;
    }

    int getOriginalIndex(int x) {
        for (int i = 0; i < 26; i++) {
            if (right[x] == left[i]) {
                return i;
            }
        }
        return -1;
    }

    void setInitialPosition(int shift) {
        for (int i = 0; i < shift; i++) {
            turn();
        }
    }
};

class Reflector {
public:
    int reflect[26];

    Reflector() {
        int mapping[26] = { 25, 24, 23, 22, 21, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2,
1, 0 };
        for (int i = 0; i < 26; i++) {
            reflect[i] = mapping[i];
        }
    }
    int reflectIndex(int x) {
        return reflect[x];
    }
};

class Plugboard {
public:
    map<char, char> wiring;

    Plugboard() {
        for (char c = 'A'; c <= 'Z'; c++) {
            wiring[c] = c;
        }
    }

    void setPlugboard(string pairs) {
        for (size_t i = 0; i < pairs.length(); i += 2) {
            if (i + 1 < pairs.length()) {
                char a = toupper(pairs[i]);
                char b = toupper(pairs[i + 1]);
                if (a != b && wiring[a] == a && wiring[b] == b) {
                    wiring[a] = b;
                    wiring[b] = a;
                }
            }
        }
    }

    char swap(char c) {
        return wiring[c];
    }
};

int main() {
    int slowl[26] = { 24, 25, 26, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23 };
    int slowr[26] = { 21, 3, 15, 1, 19, 10, 14, 26, 20, 8, 16, 7, 22, 4, 11, 5, 17, 9, 12, 23, 18, 2, 25, 6, 24, 13 };
    int midl[26] = { 26, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25 };
    int midr[26] = { 20, 1, 6, 4, 15, 3, 14, 12, 23, 5, 16, 2, 22, 19, 11, 18, 25, 24, 13, 7, 10, 8, 21, 9, 26, 17 };
    int fastl[26] = { 26, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25 };
    int fastr[26] = { 14, 8, 18, 26, 17, 20, 22, 10, 3, 13, 11, 4, 23, 5, 24, 9, 12, 25, 16, 19, 6, 15, 21, 2, 7, 1 };

    string letters = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

    while (true) {
        int shift1, shift2, shift3;
```

```
cout << "Enter three initial rotor offsets (0-25), separated by spaces (-1 to exit): ";
cin >> shift1;
if (shift1 == -1) {
        cout << "Program exited." << endl;
        break;
}
cin >> shift2 >> shift3;
cin.ignore();

Plugboard plugboard;
string plugboardPairs;
cout << "Enter plugboard pairs (e.g. AB CD EF, press enter to skip): ";
getline(cin, plugboardPairs);
plugboard.setPlugboard(plugboardPairs);

Wheel wheel1(slowl, slowr);
Wheel wheel2(midl, midr);
Wheel wheel3(fastl, fastr);
Reflector reflector;

wheel1.setInitialPosition(shift1);
wheel2.setInitialPosition(shift2);
wheel3.setInitialPosition(shift3);

string input;
cout << "Enter text: " << endl;
getline(cin, input);

string output = "";
int count1 = 0, count2 = 0, count3 = 0;

for (char c : input) {
        if (isdigit(c)) {
                output += c;
        }
        else if (isalpha(c)) {
                char uppercase = toupper(c);
                uppercase = plugboard.swap(uppercase);
                int index = letters.find(uppercase);
                if (index != string::npos) {
                        index = wheel1.getIndex(index);
                        index = wheel2.getIndex(index);
                        index = wheel3.getIndex(index);
                        index = reflector.reflectIndex(index);
                        index = wheel3.getOriginalIndex(index);
                        index = wheel2.getOriginalIndex(index);
                        index = wheel1.getOriginalIndex(index);
                        char encryptedChar = letters[index];
                        encryptedChar = plugboard.swap(encryptedChar);
                        output += encryptedChar;
                        count3++;
                        wheel3.turn();
                        if (count3 == 26) { count3 = 0; count2++; wheel2.turn(); }
                        if (count2 == 26) { count2 = 0; count1++; wheel1.turn(); }
                    }
                }
            }
            cout << "Converted text: " << endl << output << endl;
        }
        return 0;
}
```

五、测试数据及其结果分析

```
Enter three initial rotor offsets (0-25), separated by spaces (-1 to exit): 6 12 18
Enter plugboard pairs (e.g. AB CD EF, press enter to skip): AZ BY CX MN
Enter text:
```

The technology for securely sharing data has grown extensively in recent years. Many users are willing to share their lightweight mobile device data via social networks or the cloud. A novel matchmaking encryption primitive was proposed in CRYPTO19, whose potential for privacy protection and data sharing security was introduced. However, matchmaking encryption technology faces challenges in flexibly realizing critical functions, such as one-to-many non-interactive scenarios, no key escrow problem, stronger security, lightweight computation and low communication overheads for mobile devices, which impede their widespread application. To achieve the above functions, we present a lightweight certificateless multi-user matchmaking encryption (LC-MUME) for mobile devices, which enhances security flexibly and performance based on standard hard assumptions and low-consumption pairing-free technology, while also avoiding one-by-one encryption for each user. The proposed LC-MUME scheme enjoys minor computation and communication overheads in a one-to-many non-interactive certificateless cryptosystem. We prove that our scheme achieves indistinguishability-based chosen-ciphertext attack (IND-CCA) security, the existential unforgeability under a chosen message attack (EU-CMA) security and anonymity-CCA security under the random oracle model. Our LC-MUME scheme outperforms the state-of-the-art schemes regarding efficiency and flexibility, as demonstrated by the performance comparison and analysis, and therefore is a practical

```
solution for resource-constrained mobile devices.

Converted text:

CIKEKUTMNSCXEDJXVYFBJGFMKLOQSCZJWJPBNCPBKUSPPBYLNQPNKXRCPRFBUPQSZHZB

PFZJOROEICABFSMFFELNGIPWBNTBBAQFVAIGQXAAUGTEKJTFIUZRVREJGMDVMGAPRBPUY

YBCOTLZDGATESSNOEJRQZQJYZDUQTLRMGMEUQYMTYJBTFLNCPMWHPGNKVWJBQHCAHB

KOBV19ZTXCXDEAFCXAJQKYXRTFMHRFLIHCVIWRWXXAAZYKDJMQFLEAVRBCMBYPYOZR

ONSZJGTPBUJXLNTGCVIAYHTLJZWXOUZHYWYZBPSPMDAXSSUHJWDNIRKMIHVZVPOPPZU

WJQYAVAPUPLQQOPPLAVJYNIVFYUURPOICULRDVLGAZNQCWCULOIPCWCMYUPMIMCMYPD

CUFKCFZWGEINYFSELCNAHXFSOPQMISYTPURMTASGNASKOZMSACBFYCJHRWBOQEVSTRY

OADUXXGZBDHXBBPCLTVTFYUQXGIMTUTQVAJFDHDVTGNYMLGJKYYCCFNTSOXXMFUTCA

ZKSHAWFXKVJQQKRQIFYGFLZRSHJZODBCVZBFBVKEYFFWNMISUMPTSSDXNLCWYCHQNBT

QOYSPUQVJWUGLYNJRNNYRKGVJMNLOWGRMEFARPJHREGEGMAOYSEVTNRDYXHDRCJLTC

OGUQHSDCZUUFYQAZQEASHQXDGNEKNPWRZRRIOYDJMPSHGRJTWHTVKUHJTEJTGKWFSO

UQEUEXTLEKVUFEZXJDGUVPKKBTPPDLYHRPLEOFMDZQCETADBDPGHJGTPYUSQNRNUFQY

ZGHMPGSXTKIFWMBEVKDHLKSEGPIGTKLFOGTWTERFTLUMOFEQMFZHERBJBPBUXKEPHPZ

UIHRWAWSYJWFJLECOPPCHEZPAKJCFCWMPSCVJFSWHUEJYASZGSPYJFMZZXHYIHHTDIKGY

GNZFTDQKOAXPPSZSNUVTUBNDYAEZCGCXKFHIKNXRFWAHHJWXBJZHVBMHURSHOHUGGV

IEJRUYOWREXLCJVKYYAGMNYQCTZWAUFRFHPRQJPJZRYPCPYBPSZOYTWAZWRTGYKQYOX

KJOBEDROOQAMRZXVVJRSKWVFDWYRHVRLGNHUDOBMFGAWESZURYDJWIUEVDJPISJSNY

XKCKBSRRQJFRSBKAHAWWLLIDAGPJRWKFYZUCROHNZTZOZAUAQJUNISJACNGHVEEOBOW

FGKLRXJJWLMEOZDQLLNCFCRDXGOTXHYSLNXCKZHJWDRCBJVPHFNOJGQWWEOSDSJBLA

XJVARORRPUIFTWDDIBACMCVKBAATIKKELDRTQKQMCQYAAKHKXKJDXZXYACYYAZJGXV

MIE

Enter three initial rotor offsets (0-25), separated by spaces (-1 to exit): 6 12 18

Enter plugboard pairs (e.g. AB CD EF, press enter to skip): AZ BY CX MN

Enter text:

CIKEKUTMNSCXEDJXVYFBJGFMKLOQSCZJWJPBNCPBKUSPPBYLNQPNKXRCPRFBUPQSZHZB

PFZJOROEICABFSMFFELNGIPWBNTBBAQFVAIGQXAAUGTEKJTFIUZRVREJGMDVMGAPRBPUY

YBCOTLZDGATESSNOEJRQZQJYZDUQTLRMGMEUQYMTYJBTFLNCPMWHPGNKVWJBQHCAHB

KOBV19ZTXCXDEAFCXAJQKYXRTFMHRFLIHCVIWRWXXAAZYKDJMQFLEAVRBCMBYPYOZR
```

ONSZJGTPBUJXLNTGCVIAYHTLJZWXOUZHYWYZBPSPMDAXSSUHJWDNIRKMIHVZVPOPPZU

WJQYAVAPUPLQQOPPLAVJYNIVFYUURPOICULRDVLGAZNQCWCULOIPCWCMYUPMIMCMYPD

CUFKCFZWGEINYFSELCNAHXFSOPQMISYTPURMTASGNASKOZMSACBFYCJHRWBOQEVSTRY

OADUXXGZBDHXBBPCLTVTFYUQXGIMTUTQVAJFDHDVTGNYMLGJKYYCCFNTSOXXMFUTCA

ZKSHAWFXKVJQQKRQIFYGFLZRSHJZODBCVZBFBVKEYFFWNMISUMPTSSDXNLCWYCHQNBT

QOYSPUQVJWUGLYNJRNNYRKGVJMNLOWGRMEFARPJHREGEGMAOYSEVTNRDYXHDRCJLTC

OGUQHSDCZUUFYQAZQEASHQXDGNEKNPWRZRRIOYDJMPSHGRJTWHTVKUHJTEJTGKWFSO

UQEUEXTLEKVUFEZXJDGUVPKKBTPPDLYHRPLEOFMDZQCETADBDPGHJGTPYUSQNRNUFQY

ZGHMPGSXTKIFWMBEVKDHLKSEGPIGTKLFOGTWTERFTLUMOFEQMFZHERBJBPBUXKEPHPZ

UIHRWAWSYJWFJLECOPPCHEZPAKJCFCWMPSCVJFSWHUEJYASZGSPYJFMZZXHYIHHTDIKGY

GNZFTDQKOAXPPSZSNUVTUBNDYAEZCGCXKFHIKNXRFWAHHJWXBJZHVBMHURSHOHUGGV

IEJRUYOWREXLCJVKYYAGMNYQCTZWAUFRFHPRQJPJZRYPCPYBPSZOYTWAZWRTGYKQYOX

KJOBEDROOQAMRZXVVJRSKWVFDWYRHVRLGNHUDOBMFGAWESZURYDJWIUEVDJPISJSNY

XKCKBSRRQJFRSBKAHAWWLLIDAGPJRWKFYZUCROHNZTZOZAUAQJUNISJACNGHVEEOBOW

FGKLRXJJWLMEOZDQLLNCFCRDXGOTXHYSLNXCKZHJWDRCBJVPHFNOJGQWWEOSDSJBLA

XJVARORRPUIFTWDDIBACMCVKBAATIKKELDRTQKQMCQYAAKHKXKJDXZXYACYYAZJGXV

MIE

Converted text:

THETECHNOLOGYFORSECURELYSHARINGDATAHASGROWNEXTENSIVELYINRECENTYEARSM

ANYUSERSAREWILLINGTOSHARETHEIRLIGHTWEIGHTMOBILEDEVICEDATAVIASOCIALNETW

ORKSORTHECLOUDANOVELMATCHMAKINGENCRYPTIONPRIMITIVEWASPROPOSEDINCRYPT

O19WHOSEPOTENTIALFORPRIVACYPROTECTIONANDDATASHARINGSECURITYWASINTRODU

CEDHOWEVERMATCHMAKINGENCRYPTIONTECHNOLOGYFACESCHALLENGESINFLEXIBLYRE

ALIZINGCRITICALFUNCTIONSSUCHASONETOMANYNONINTERACTIVESCENARIOSNOKEYESC

ROWPROBLEMSTRONGERSECURITYLIGHTWEIGHTCOMPUTATIONANDLOWCOMMUNICATION

OVERHEADSFORMOBILEDEVICESWHICHIMPEDETHEIRWIDESPREADAPPLICATIONTOACHIEV

ETHEABOVEFUNCTIONSWEPRESENTALIGHTWEIGHTCERTIFICATELESSMULTIUSERMATCHMA

KINGENCRYPTIONLCMUMEFORMOBILEDEVICESWHICHENHANCESSECURITYFLEXIBLYANDP

ERFORMANCEBASEDONSTANDARDHARDASSUMPTIONSANDLOWCONSUMPTIONPAIRINGFRE

ETECHNOLOGYWHILEALSOAVOIDINGONEBYONEENCRYPTIONFOREACHUSERTHEPROPOSED

```
LCMUMESCHEMEENJOYSMINORCOMPUTATIONANDCOMMUNICATIONOVERHEADSINAONET
OMANYNONINTERACTIVECERTIFICATELESSCRYPTOSYSTEMWEPROVETHATOURSCHEMEAC
HIEVESINDISTINGUISHABILITYBASEDCHOSENCIPHERTEXTATTACKINDCCASECURITYTHEEXI
STENTIALUNFORGEABILITYUNDERACHOSENMESSAGEATTACKEUCMASECURITYANDANONY
MITYCCASECURITYUNDERTHERANDOMORACLEMODELOURLCMUMESCHEMEOUTPERFORM
STHESTATEOFTHEARTSCHEMESREGARDINGEFFICIENCYANDFLEXIBILITYASDEMONSTRATED
BYTHEPERFORMANCECOMPARISONANDANALYSISANDTHEREFOREISAPRACTICALSOLUTION
FORRESOURCECONSTRAINEDMOBILEDEVICES
Enter three initial rotor offsets (0-25), separated by spaces (-1 to exit): -1
Program exited.
```

六、调试过程中的问题

调试过程中未出现问题。

七、课程总结

通过本次实验，深入理解了转轮机加密的工作原理，掌握了基于 C++的数据结构和算法的实现方法。实验表明，转轮机加密可以提供较强的加密能力，但由于其密钥空间有限，仍然容易被暴力破解。在现代密码学中，转轮机的原理仍然对于对称加密算法的设计有所启发。