Лабораторная работа №2

Подмогильный Иван Александрович - студент группы НПМмд-02-22

16.09.2022

Шифры перестановки

Прагматика выполнения

Умение пользоваться методами маршрутного, решеточного, Виженера шифрований

Цель выполнения лабораторной работы

Освоить на практике использование методов маршрутного, решеточного, Виженера шифрований

Задачи выполнения работы

Написать функции, которые реализуют шифрование маршрутного, решеточного, Виженера.

Результаты выполнения лабораторной работы. Написал код для зашивровки кодов Маршрутным шифрованием

```cpp
void CipherPath::cipher(const std::string &msg, const std::string &pwd, std::string &encrypted) {
    if (!encrypted.empty()){
        throw std::invalid_argument( "encrypted is not empty !" );
    }

    if (!msg.empty()){
        std::string pwdSorted = pwd;
        std::sort(pwdSorted.begin(), pwdSorted.end());

        // cols is actually the length of the pwd
        const uint cols = pwd.size();
        uint rows;
        std::string msgAligned = msg;
        if (msg.size() % cols == 0){
            rows = msg.size() / cols;
        }
        else{
            rows = msg.size() / cols + 1;
            // if the string is too small to complete the last block, add the random characters to the end.
            for (int i = 0; i < (cols - msg.size() % cols ); i++){
                msgAligned += "b";
            }
        }

        // for debugging purposes
        for (int i = 0; i < rows; i++){
            for (int j = 0; j < cols; j++){
                std::cout << msgAligned[i*cols + j] << " ";
            }
            std::cout << std::endl;
        }
        std::cout << std::string( n: cols*2, c: '-') << std::endl;
        for (auto c: pwd){
            std::cout << c << " ";
        }
        std::cout << std::endl << std::endl;
        // end debugging output

        // we actually don't have to create a big matrix of the message. we can just take the indexes corresponding to the
        // column of the password
        for (int i = 0; i < pwd.size(); i++){
            auto index = pwd.find(pwdSorted[i]);
            for (int j = 0; j < rows; j++){
                encrypted += msgAligned[j*cols + index];
            }
        }
```

Написал код для дешивровки кодов Маршрутным шифрованием

```cpp
void CipherPath::decipher(const std::string &msg, const std::string &pwd, std::string &decrypted) {
    if (!decrypted.empty()){
        throw std::invalid_argument( "decrypted is not empty!" );
    }

    if (!msg.empty()){
        std::string pwdSorted = pwd;
        std::sort(pwdSorted.begin(), pwdSorted.end());
        // we need to know what index of pwd corresponds to index in pwdSorted, because we're reconstructing the string
        // starting from the original first element of the password
        std::map<uint, uint> pwd2pwdSorted;
        const uint cols = pwd.size();
        const uint rows = msg.size() / cols;
        for (int i = 0; i < pwd.size(); i++){
            pwd2pwdSorted[i] = pwdSorted.find(pwd[i]);
        }

        // the formula was just derived from my observations.
        for (int row = 0; row < rows; row++){
            for (int col = 0; col < cols; col++){
                decrypted += msg[pwd2pwdSorted[col] * rows + row];
            }
        }
    }
}
```

Figure 2: функция дешифрования Маршрутным шифрованием

Написал код для зашивровки кодов с помощью решеточного шифрования

```
CipherLattice::indexes_pwdAligned CipherLattice::cipher(const std::string &msg, const std::string& pwd,
                                    const int k, std::string &encrypted, const int tolerance, char fill) {
    if (!encrypted.empty()){
        throw std::invalid_argument( "encrypted is not empty! ");
    }
    const auto k_squared = (int)std::pow(k,  y: 2);
    auto diff = (int)pwd.size() - (int)std::pow(k,  y: 2);

    std::string pwdAligned = pwd;
    if ( abs(diff) > tolerance){
        std::string error = "tolerance is: " + std::to_string(tolerance) +
                " but the diff k^2 - pwd.size() is: " + std::to_string(diff);
        throw std::invalid_argument ( error );
    }
    // if diff > 0 => password size is bigger than k^2 => need to reduce
    else if (diff > 0){
        for (int i = 0; i < diff; i++){
            pwdAligned.pop_back();
        }
    }
    // if diff < 0 => password size if smaller than k^2 => need to add more
    else if (diff < 0){
        for (int i = 0; i > diff; i--){
            pwdAligned += fill;
        }
    }

    Eigen::MatrixXi block(k, k);
    // fill the block
    for (int i = 0; i < k; i++){
        for (int j = 0; j < k; j++){
            block(i, j) = (i*k + j) + 1;
        }
    }

    std::vector<Eigen::MatrixXi> table;
    Eigen::MatrixXi eigenTableInitial( x: k*2,  y: k*2);
    auto rotatedBlock = block;
    // fill the table
    for (int i = 0; i <= 4; i++){
//        std::cout << "push to table the block: " << std::endl << rotatedBlock << std::endl;
        table.push_back(rotatedBlock);
        rotate( & rotatedBlock);
    }
```

Часть 2

```
//    std::cout << "The constructed table is: " << std::endl << eigenTableInitial << std::endl;
    // by now rotatedBlock is the initial block (rotated by 360 degrees)

    std::random_device rd;
    std::mt19937 gen(rd());
    std::uniform_int_distribution<> distr( a: 0,  b: 3);
    std::vector< std::pair<int, int> > indexes;

    // this is a low-skilled fix to the problem of random initialization. of course you can come up with something better
    // for example, exit if after k^2 + 1 iterations the table is not filled.
    std::array<int, 4> chosen_parts = { _M_elems[0]: 1,  _M_elems[1]: 0,  _M_elems[2]: 0,  _M_elems[3]: 0};
    // compose a matrix of indexes
    Eigen::MatrixXi eigenTable = Eigen::MatrixXi::Zero( rows: k*2,  cols: k*2);
    for (int el = 0; el < k_squared; el++){
        // use predefined table instead of random generation.
        uint chosen_part = chosen_parts[el];
//        uint chosen_part = distr(gen);
        // search for the element in the block
        // Probably the function find_element is not working correctly. Test and debug the function.
//        std::cout << "chosen table: " << std::endl << table[chosen_part] << std::endl;
        std::pair<int, int> rowCol;
        if (chosen_part == 2){
            rowCol = find_element( &: table[chosen_part+1],  element: el+1);
        }
        if (chosen_part == 3){
            rowCol = find_element( &: table[chosen_part-1],  element: el+1);
        }
        else{
            rowCol = find_element( &: table[chosen_part],  element: el+1);
        }

//        std::cout << "found element is: " << rowCol.first << " " << rowCol.second << std::endl;
        // convert found index to the global index.
        if (chosen_part == 1){
            // if it's the second part, the row remains the same, but the column is skewed
            rowCol.second += k;
        }
        else if (chosen_part == 2){
            // if it's the third part, the column remains the same, but the row goes down
            rowCol.first += k;
            rowCol.second += k;
        }
        else if (chosen_part == 3){
            // finally, if it's the fourth part, the column and the row are skewed
            rowCol.first += k;
        }
        eigenTable(rowCol.first, rowCol.second) = el + 1;
        indexes.push_back(rowCol);
    }
```

```
    Eigen::Matrix<char, Eigen::Dynamic, Eigen::Dynamic> filledTable = Eigen::Matrix<char, Eigen::Dynamic, Eigen::Dynamic>::Zero( rows: k*2,  cols: k*2);
    Eigen::MatrixXi mToRotate = eigenTable;
    rotate( & mToRotate);


    // rotate matrix of indexes until the filledTable is all filled (4 times)

    // fill table the first time
    for (int i = 0; i < k_squared; i++){
        filledTable(indexes[i].first, indexes[i].second) = msg[i];
    }
//    std::cout << "initial fill: " << std::endl << filledTable << std::endl;

    // rotate and fill the rest of the times
//    for (int i = 1; i < 4; i++) {
    int filled_table_iterator = 1;
    // while the table is not filled.

//    std::cout << "filledTable is: " << std::endl << filledTable << std::endl;
//    std::cout << "The result of filledTable: " << (filledTable.array() == 0) << std::endl;

    while ( (filledTable.array() == 0).any() ){
//        std::cout << "mToRotate on iteration: " << filled_table_iterator << std::endl << mToRotate << std::endl;
//        std::cout << "filledTable on iteration: " << filled_table_iterator << std::endl << filledTable << std::endl;
        for (int el = 0; el < k_squared; el++) {
            auto elIndexes = find_element( & mToRotate,  element: el + 1);
            if (filledTable(elIndexes.first, elIndexes.second) == 0){
                filledTable(elIndexes.first, elIndexes.second) = msg[filled_table_iterator * k_squared + el];
            }
        }
        rotate( & mToRotate);
        filled_table_iterator += 1;
    } // now the matrix is considered to be filled

//    std::cout << "after rotations: " << std::endl << filledTable << std::endl;

    // permute the columns according to the alphabetical order of the pwd
    std::string pwdSorted = pwdAligned;
    std::sort(pwdSorted.begin(), pwdSorted.end());
    std::map<uint, uint> pwdSorted2pwd;
    create_mapping(pwdSorted, pwdAligned,  & pwdSorted2pwd);
    auto permTable = filledTable;
```

Figure 5: функция шифрования решетками 3

```
//    std::cout << "pwdAligned: " << pwdAligned << std::endl;
//    std::cout << "pwdSorted: " << pwdSorted << std::endl;
//    std::cout << "pwdSorted2pwd: ";
//    for (int i = 0; i < pwdSorted.size(); i++){
//        std::cout << i << ", " << pwdSorted2pwd[i] << "  ";
//    }
//    std::cout << std::endl << std::endl;

//    std::cout << "perm table: " << std::endl << permTable << std::endl;
//    std::cout << "filled table: " << std::endl << filledTable << std::endl;


    // take the pwdSorted index, find it's corresponding index in the pwd and substitude
    // the column with that index to the index of pwdSorted
    // TODO: Create the function for permutations
    for (int i = 0; i < pwdSorted.size(); i++){
//        std::cout << "column from filledTable: " << std::endl << filledTable.col(pwdSorted2pwd[i]).eval() << std::endl;
        permTable.col(i) = filledTable.col(pwdSorted2pwd[i]).eval();
//        permTable.col(pwdSorted2pwd[i]) = filledTable.col(i).eval();
//        std::cout << "filledTable: " << std::endl << permTable << std::endl;
    }

//    std::cout << "permuted table: " << std::endl << permTable << std::endl;
    // now, when the columns are permuted we can write to the cipher message
    for (int col = 0; col < permTable.cols(); col++){
        for (int row = 0; row < permTable.rows(); row++){
            encrypted += permTable(row, col);
        }
    }

    CipherLattice::indexes_pwdAligned i_pwd(indexes, &pwdAligned);
    return i_pwd;
}
```

Figure 6: функция шифрования решетками 4

Написал код для зашивровки кодов с помощью таблицы Виженера

```
// considering there's only capital letters
// TODO: It seems that in order to support the russian letters it's better to use wstring, or the array of the russian
//  chars => it's better to create a template which accepts different types of input variables
void CipherVigenereTable::cipher(const std::string &msg, const std::string &pwd, std::string& encrypted) {
    std::string pwdAligned;
    for (int i = 0; i < msg.size(); i++){
        auto ind = i % pwd.size();
        pwdAligned += pwd[ind];
    }

    for (int i = 0; i < msg.length(); i++){
        if (msg[i] != ' '){
//            auto msg_size = msg.size() / sizeof(msg[0]);
            auto rowIndex = CipherHelper::engAlphabetLower.find(pwdAligned[i]);
            auto colIndex = CipherHelper::engAlphabetLower.find(msg[i]);
            encrypted += cipherEngTable[rowIndex][colIndex];
        }

    }
}
```

Figure 7: зашивровки кодов с помощью таблицы Виженера

Написал заголовочный файл для класса реализации CipherHelper2

```cpp
#ifndef LAB02_CIPHERHELPER2_H
#define LAB02_CIPHERHELPER2_H

#include "../../lab01/include/CipherHelper.h"
#include <vector>
#include <eigen3/Eigen/Core>
#include <map>

class CipherHelper2 : public CipherHelper {
public:
    static const std::string rusAlphabetLower;
    static const std::string rusAlphabetUpper;
};

class CipherPath : CipherHelper2 {
public:
    static void cipher(const std::string& msg, const std::string& pwd, std::string& encrypted);
    static void decipher(const std::string& msg, const std::string& pwd, std::string& decrypted);
};

class CipherLattice : CipherHelper2 {
public:
    // we need the pwd, and the array of indexes which contains the indexes which were cutted from the matrix.
    // to build the matrix we can use the additionally created function for reconstructing matrix from the msg and pwd
    // returns the indexes which were chosen to create the lattice.
    // tolerance is the number of cut/add characters function is allowed to cut/add to the pwd. If pwd length is 4, and
    // desired k is 3, therefore, k^2 = 9, we need to add 5 additional characters to pwd. The function will not do that
    // in case if tolerance = 2 and throw and error. Otherwise, it will fill with 'a' character by default (or cut)
    typedef std::pair<std::vector<std::pair<int, int>>, std::string&> indexes_pwdAligned;
    static indexes_pwdAligned cipher(const std::string& msg, const std::string& pwd, int k,
                                     std::string& encrypted, int tolerance = 2, char fill = 'a');
    static void decipher(const std::string& msg, const indexes_pwdAligned& i_pwd, std::string& decrypted);
```

Figure 8: заголовочный файл для класса реализации CipherHelper2 1

Написал заголовочный файл для класса реализации CipherHelper2. 2

```
private:
    // rotates clockwise by 90 degrees => reversed columns are now rows.
    static void rotate(Eigen::MatrixXi& input){
        Eigen::MatrixXi res = input;
//        std::cout << "Before reverse: " << std::endl << input << std::endl;
//        std::cout << "input before reverse: " << std::endl << input << std::endl;
        // read about Eigens' aliasing: https://eigen.tuxfamily.org/dox/group__TopicAliasing.html and why to use .eval()
        input = input.colwise().reverse().eval();
//        std::cout << "input after reverse: " << std::endl << input << std::endl;
//        std::cout << "After reverse: "<<  std::endl << input << std::endl;
//        std::cout << "res: " << std::endl << res << std::endl;
//        std::cout << "output: " << std::endl << input << std::endl;
        for (int i = 0; i < input.rows(); i++){
            res.row(i) = input.col(i);
        }
        input = res;
//        std::cout << "resulting matrix: " << std::endl << res << std::endl;
    }

    // bear in mind that the numbering in matrix is from 1. The function assumes that you've taken that into account
    static std::pair<int, int> find_element(Eigen::MatrixXi& input, int element){
        for (int row = 0; row < input.rows(); row++){
            for (int col = 0; col < input.cols(); col++){
                if ( input(row, col) == element ){
                    std::pair<int, int> res(row, col);
                    return res;
                }
            }
        }
    }

    static void create_mapping(const std::string& from, const std::string& to, std::map<uint, uint>& output){
        for (int i = 0; i < from.size(); i++){
            auto index = to.find(from[i]);
            output[i] = index;
        }
    }
};

class CipherVigenereTable : CipherHelper2 {
public:
    const static std::vector<std::string> cipherRusTable, cipherEngTable;
    static void cipher(const std::string& msg, const std::string& pwd, std::string& encrypted);
    static void decipher(const std::string& msg, const std::string& pwd, std::string& decrypted);

};
```

Figure 9: заголовочный файл для класса реализации CipherHelper2 1

Написал CMakeLists.txt.

Файл который создаёт библиотеку из класса CipherHelper2 и бинарник main, и прилинковывает библиотеку CipherHelper.

```
cmake_minimum_required(VERSION 3.20)
project(lab02)

set(CMAKE_CXX_STANDARD 14)

find_package(Eigen3)
add_library(lab01 ../lab01/src/CipherHelper.cpp ../lab01/include/CipherHelper.h)
add_library(lab02 src/CipherHelper2.cpp include/CipherHelper2.h)
target_link_libraries(lab02)

add_executable(main src/main.cpp)
target_link_libraries(main lab02 lab01)

```

Figure 10: CMakeLists.txt файл

Написал main.cpp файл, в котором есть тесты реализованных функций.

```cpp
#include <iostream>
#include "../include/CipherHelper2.h"

int main() {
//    std::setlocale(LC_ALL, "RUS");
    std::string test1 = "nelzya nedooocenivat protivnika";
    std::string test2 = "hello world! Kulyabov is THE BEST";
    std::string test3 = "kriptographia seryoznaya nauka";
    std::wstring test4( s: reinterpret_cast<const wchar_t *>("КРИПТОГРАФИЯ СЕРЬЕЗНАЯ НАУКА"));
    std::wstring pwd4( s: reinterpret_cast<const wchar_t *>("МАТЕМАТИКА"));
    std::wstring enc4;
    std::string enc1, enc2, enc3;
    std::string dec1, dec2, dec3;

    CipherPath::cipher(test1, pwd: "parol", &: enc1);
    std::cout << enc1 << std::endl;
    CipherPath::decipher(enc1, pwd: "parol", &: dec1);
    std::cout << dec1 << std::endl << std::endl << std::endl;

    // we can't input the word which repeats at least one character inside
    CipherPath::cipher(test2, pwd: "pasword", &: enc2);
    std::cout << enc2 << std::endl;
    CipherPath::decipher(enc2, pwd: "pasword", &: dec2);
    std::cout << dec2 << std::endl << std::endl << std::endl;

    // NOTE: Works only for k = 2, and password with number k^2. Please refer to the function definition and search for:
    // "this is a low-skilled fix" to find out why. If you have time you can expand the function.
    enc1 = "";
    std::cout << "CipherLattice: " << std::endl;
    std::cout << "message: " << test1 << std::endl;
    CipherLattice::cipher(test1, pwd: "paroli", k: 2, &: enc1);
    std::cout << "encrypted: " << enc1 << std::endl << std::endl;

    std::cout << "Cipher Vigenere Table: " << std::endl;
    std::string pwd3 = "matematika";
//    std::cout << pwd3.size() << std::endl;
    CipherVigenereTable::cipher(test3, pwd3, &: enc3);
    std::cout << enc3 << std::endl << std::endl;

    return 0;
}
```

Figure 11: main.cpp файл

Результаты тестов.

```
/home/pi/education/pfur_masters/matOsnovyInfBez/labs/lab02/cmake-build-debug/main
n e l z y
a   n e d
o o c e n
i v a t
p r o t i
v n i k a
---------
p a r o l

e ovrnydn iazeettknaoipvlncaoi
nelzya nedoocenivat protivnika


h e l l o   w
o r l d !   K
u l y a b o v
  i s   T H E
  B E S T b b
-------------
p a s w o r d

erliBwKvEbo!bTThou    oHbllysElda S
hello world! Kulyabov is THE BESTbb


CipherLattice:
message: nelzya nedoocenivat protivnika
encrypted: ezinnaoyclee nod

Cipher Vigenere Table:
wrbtfozzkptiteekgyzzarentcua


Process finished with exit code 0
```

Выводы

Освоил на практике применение методов маршрутного, решеточного, Виженера шифрований