

# **Информационная безопасность.**

**Лабораторная работа №8.**

Подмогильный Иван Александрович.

# Содержание

1. Цель работы	5
2. Задание	6
3. Выполнение лабораторной работы	7
4. Выводы	11

## Список иллюстраций

3.1. Вывод . . . . .	7
3.2. Код 1 . . . . .	8
3.3. Код 2 . . . . .	8
3.4. Код 3 . . . . .	9
3.5. Код 4 . . . . .	9
3.6. Код 5 . . . . .	10

## Список таблиц

# 1. Цель работы

Освоить на практике применение режима однократного гаммирования на примере кодирования различных исходных текстов одним ключом

## 2. Задание

Написать код для шифровки и дешифровки сообщений.

### 3. Выполнение лабораторной работы

Написал код для зашивровки и дешивровки кодов. Вывод программы. Для подробных объяснений обратитесь к скринкасту на youtube.

[illegible]

Рис. 3.1.: Вывод

Код:

```

code xor_cipher.cpp xor_cipher.h support.cpp support.h main.cpp CMakeLists.txt
// Created by Ivan on 17.12.2021.
//
#include <iostream>
#include <string>
#include <vector>
#include <support.h>

// Function that takes a string and translates it to the sequence of numbers
std::vector<int> string_to_numbers(const std::string& seq){
    std::vector<int> seq;
    std::cout << "seq length: " << seq.length() << std::endl;

    for (char c : seq) {
        // Since it's Russian, we have to add static_cast
        std::cout << static_cast<unsigned char>(c) << " ";
        seq.push_back( (int)(static_cast<unsigned char>(c)) );
    }
    std::cout << std::endl;

    return seq;
}

// Function that takes a HEX sequence and translates it to the numbers
std::vector<int> hexseq_to_numbers(const std::vector<std::string& hex_seq){
    std::vector<int> seq;
    seq.reserve(hex_seq.size());
    for (const std::string& hex : hex_seq) {
        seq.push_back( stoi(hex, nullptr, 16) );
    }
    return seq;
}

```

Рис. 3.2.: Код 1

```

code xor_cipher.cpp xor_cipher.h support.cpp support.h main.cpp CMakeLists.txt
// Function that takes a HEX sequence and translates it to the numbers
std::vector<int> hexseq_to_numbers(const std::vector<std::string& hex_seq){
    std::vector<int> seq;
    seq.reserve(hex_seq.size());
    for (const std::string& hex : hex_seq) {
        seq.push_back( stoi(hex, nullptr, 16) );
    }
    return seq;
}

// Function that takes 2 seqs of numbers and performs XOR
std::vector<int> perform_xor(const std::vector<int>& l1, const std::vector<int>& r1){
    std::vector<int> seq;
    seq.reserve(l1.size());
    for (int i = 0; i < l1.size(); i++) {
        seq.push_back( l1[i] ^ r1[i] );
    }
    return seq;
}

// Function that takes a seq of numbers and makes it to the complete string
std::string seq_to_string(const std::vector<int>& seq){
    std::string res;
    for (const int i : seq) {
        res += (char)static_cast<unsigned char>(i);
    }
    return res;
}

```

Рис. 3.3.: Код 2



```

code
xor_cipher.cpp
xor_cipher.h
support.cpp
support.h
main.cpp
CMakeLists.txt

// Function that takes a seq of numbers and saves it to the complete string
std::string seq_to_string(const std::vector<int>& seq) {
    std::string res;
    for (const auto& i : seq) {
        res += (char)std::stringstream::operator<<(i);
    }
    return res;
}

std::string retrieve_msg(const std::vector<int>& encrypted1, const std::vector<int>& encrypted2,
                        const std::vector<int>& key_seq) {
    auto c1_seq_c1 = perform_xor(encrypted1, encrypted2);
    auto deciphered_seq = perform_xor(c1_seq_c1, key_seq);
    std::string msg2 = seq_to_string(deciphered_seq);
    return msg2;
}

int main() {
    // We need to use it to change char conversions to Russian
    std::setlocale(LC_ALL, "ru_RU");

    std::cout << "Encrypting program... " << std::endl;

    auto c1_seq_char1 = "Kryptograficheskaya";
    auto c1_seq_char2 = "Kriptograficheskaya1234";

    for (auto& i : c1_seq_char1) {
        std::cout << i << " ";
    }
    std::cout << std::endl;

    std::string msg = "Kryptograficheskaya1234";
    std::cout << msg << std::endl;
    std::string msg2 = "Kriptograficheskaya";
    std::vector<int> seq_key = { 'K', 'R', 'I', 'P', 'T', 'O', 'G', 'R', 'A', 'F', 'I', 'C', 'H', 'E', 'S', 'K', 'A', 'Y', 'A', '1', '2', '3', '4' };
    auto msg_seq = string_to_numbers(msg);
    auto msg2_seq = string_to_numbers(msg2);
    auto key_seq = hexseq_to_numbers(seq_key);

    for (const auto& i : key_seq) {
        std::cout << i << " ";
    }
    std::cout << std::endl;

    // This is encrypted now
    auto encrypted = perform_xor(msg_seq, key_seq);
    auto encrypted2 = perform_xor(msg2_seq, key_seq);

    for (const auto& i : encrypted1) {
        std::cout << i << " ";
    }
    std::cout << std::endl;

    // This is deciphered now
    auto org_seq = perform_xor(encrypted, key_seq);

    std::string retrieved_msg2 = retrieve_msg(encrypted, encrypted2, key_seq);
    std::cout << "Retrieved msg2 without key: " << retrieved_msg2 << std::endl;

    for (const auto& i : org_seq) {
        std::cout << i << " ";
    }
    std::cout << std::endl;
}

```

Рис. 3.4.: Код 3

```

code
xor_cipher.cpp
xor_cipher.h
support.cpp
support.h
main.cpp
CMakeLists.txt

// Function that takes a seq of numbers and saves it to the complete string
std::string seq_to_string(const std::vector<int>& seq) {
    std::string res;
    for (const auto& i : seq) {
        res += (char)std::stringstream::operator<<(i);
    }
    return res;
}

std::string retrieve_msg(const std::vector<int>& encrypted1, const std::vector<int>& encrypted2,
                        const std::vector<int>& key_seq) {
    auto c1_seq_c1 = perform_xor(encrypted1, encrypted2);
    auto deciphered_seq = perform_xor(c1_seq_c1, key_seq);
    std::string msg2 = seq_to_string(deciphered_seq);
    return msg2;
}

int main() {
    // We need to use it to change char conversions to Russian
    std::setlocale(LC_ALL, "ru_RU");

    std::cout << "Encrypting program... " << std::endl;

    auto c1_seq_char1 = "Kryptograficheskaya";
    auto c1_seq_char2 = "Kriptograficheskaya1234";

    for (auto& i : c1_seq_char1) {
        std::cout << i << " ";
    }
    std::cout << std::endl;

    std::string msg = "Kryptograficheskaya1234";
    std::cout << msg << std::endl;
    std::string msg2 = "Kriptograficheskaya";
    std::vector<int> seq_key = { 'K', 'R', 'I', 'P', 'T', 'O', 'G', 'R', 'A', 'F', 'I', 'C', 'H', 'E', 'S', 'K', 'A', 'Y', 'A', '1', '2', '3', '4' };
    auto msg_seq = string_to_numbers(msg);
    auto msg2_seq = string_to_numbers(msg2);
    auto key_seq = hexseq_to_numbers(seq_key);

    for (const auto& i : key_seq) {
        std::cout << i << " ";
    }
    std::cout << std::endl;

    // This is encrypted now
    auto encrypted = perform_xor(msg_seq, key_seq);
    auto encrypted2 = perform_xor(msg2_seq, key_seq);

    for (const auto& i : encrypted1) {
        std::cout << i << " ";
    }
    std::cout << std::endl;

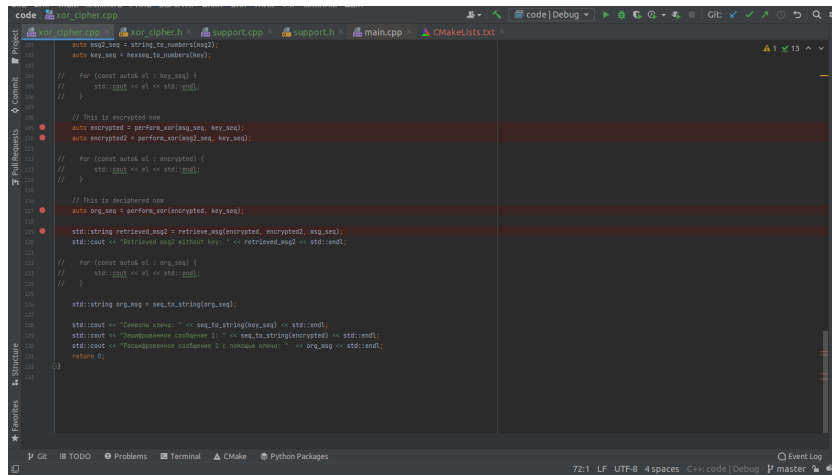
    // This is deciphered now
    auto org_seq = perform_xor(encrypted, key_seq);

    std::string retrieved_msg2 = retrieve_msg(encrypted, encrypted2, key_seq);
    std::cout << "Retrieved msg2 without key: " << retrieved_msg2 << std::endl;

    for (const auto& i : org_seq) {
        std::cout << i << " ";
    }
    std::cout << std::endl;
}

```

Рис. 3.5.: Код 4



```
code
xor_cipher.cpp
xor_cipher.h
support.cpp
support.h
main.cpp
CMakeLists.txt

100 auto msg_seq = string_to_numbers(msg);
101 auto key_seq = hexseq_to_numbers(key);
102
103 // For (const auto& el : key_seq) {
104 //     std::cout << el << std::endl;
105 // }
106
107 // This is encrypted now
108 auto encrypted = perform_xor(msg_seq, key_seq);
109 auto encrypted2 = perform_xor(msg2_seq, key_seq);
110
111 // For (const auto& el : encrypted) {
112 //     std::cout << el << std::endl;
113 // }
114
115 // This is deciphered now
116 auto org_seq = perform_xor(encrypted, key_seq);
117
118 std::string retrieved_msg2 = retrieve_msg(encrypted, msg_seq);
119 std::cout << "Retrieved msg2 without key: " << retrieved_msg2 << std::endl;
120
121 // For (const auto& el : org_seq) {
122 //     std::cout << el << std::endl;
123 // }
124
125 std::string org_msg = seq_to_string(org_seq);
126
127 std::cout << "Message name: " << seq_to_string(key_seq) << std::endl;
128 std::cout << "Decryption codebase 1: " << seq_to_string(encrypted) << std::endl;
129 std::cout << "Decryption codebase 2: " << org_msg << std::endl;
130 return 0;
131 }
132
133 }
```

Рис. 3.6.: Код 5

## 4. Выводы

Освоил на практике применение режима однократного гаммирования на примере кодирования различных исходных текстов одним ключом