

Лабораторная работа №8

Zayan Ahamed Ally - студент группы НКНбд-01-18

08.12.2021

Элементы криптографии.

Однократное гаммирование

Элементы криптографии. Шифрование (кодирование) различных исходных текстов одним ключом

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

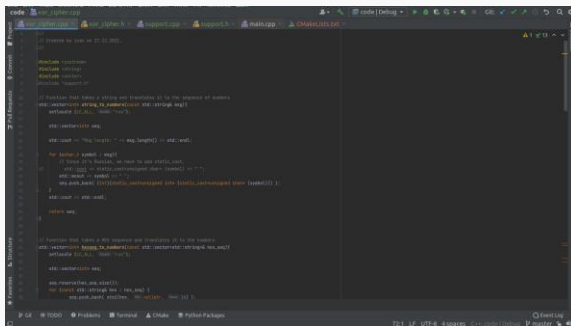
Написать приложение, которое кодирует и декодирует сообщения с помощью хог-шифра. А также то, которое может декодировать второе сообщение, имея два зашифрованных сообщения и одно оригинальное сообщение.

Результаты выполнения лабораторной работы. Часть 1

Написал код, шифрующий и дешифрующий сообщение. Вывод.

[illegible]

Рис. 1: Код 1



```
code
new_project.cpp
new_project.h
support.cpp
support.h
main.cpp
CMAKELists.txt

// Created by Ivan on 27.11.2022.
//

#include <iostream>
#include <string>
#include <cstdlib>
#include <cstring>

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

    for (int i = 0; i < str.length(); i++)
    {
        if (str[i] < '0' || str[i] > '9')
        {
            std::cout << "Invalid character: " << str[i] << std::endl;
            return res;
        }
        int digit = str[i] - '0';
        std::cout << "Digit: " << digit << std::endl;
        res.push_back(digit);
    }

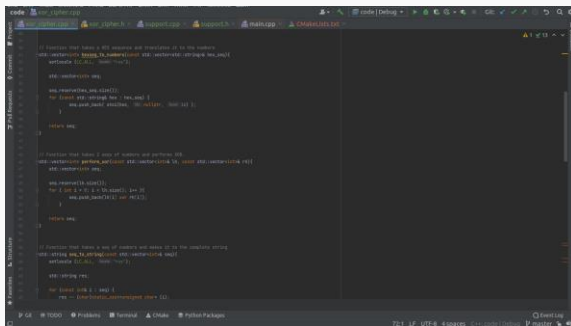
    return res;
}

// Function that takes a sequence of numbers and translates it to the number
std::vector<int> numberToNumber(const std::vector<int>& num)
{
    std::vector<int> res;
    std::cout << "Number length: " << num.size() << std::endl;

    int result = 0;
    for (int i = 0; i < num.size(); i++)
    {
        result = result * 10 + num[i];
    }

    return result;
}
```

Рис. 2: Код 2

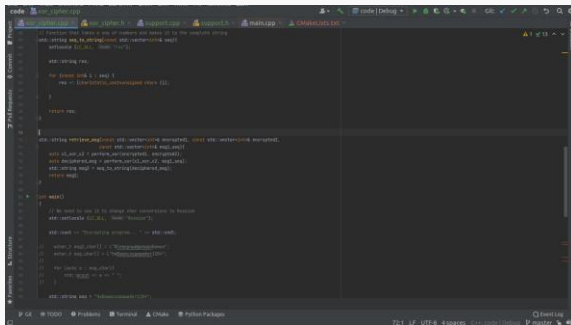


```
code
// ...
// Function that takes a 0/1 sequence and translates it to the numbers
void vectorMatrix_multiply_1D_vector(const unsigned long long* seq,
    vectorize_t(C, 0.5), const float*) {
    std::vector<float> seq;
    seq.reserve(1000000000);
    for (int i = 0; i < seq.size(); i++) {
        seq.push_back(seq[i] * 0.5);
    }
    return seq;
}

// Function that takes 2 sets of numbers and performs dot
void vectorMatrix_multiply_2D_vector(const unsigned long long* seq,
    const std::vector<float> seq) {
    std::vector<float> seq;
    seq.reserve(1000000000);
    for (int i = 0; i < seq.size(); i++) {
        seq.push_back(seq[i] * 0.5);
    }
    return seq;
}

// Function that takes a set of numbers and makes it to the numbers string
void string_vectorize(const unsigned long long* seq) {
    std::string res;
    for (int i = 0; i < seq.size(); i++) {
        res += std::to_string(seq[i]) + " ";
    }
}
```

Рис. 3: Код 3



```
code
// Create new node & insert at the beginning
void string_list_insert_at_begin(string_list_t* list, const char* str)
{
    struct node* new_node = (struct node*) malloc(sizeof(struct node));
    if (!new_node) {
        printf("Error: Memory allocation failed\n");
        return;
    }
    new_node->data = strdup(str);
    new_node->next = list->head;
    list->head = new_node;
}

// Insert new node at the end
void string_list_insert_at_end(string_list_t* list, const char* str)
{
    struct node* new_node = (struct node*) malloc(sizeof(struct node));
    if (!new_node) {
        printf("Error: Memory allocation failed\n");
        return;
    }
    new_node->data = strdup(str);
    new_node->next = NULL;
    if (!list->head) {
        list->head = new_node;
    } else {
        struct node* current = list->head;
        while (current->next != NULL) {
            current = current->next;
        }
        current->next = new_node;
    }
}

// Delete a node from the list
void string_list_delete(string_list_t* list, const char* str)
{
    if (!list->head) {
        printf("List is empty\n");
        return;
    }
    struct node* current = list->head;
    struct node* prev = NULL;
    while (current != NULL) {
        if (strcmp(current->data, str) == 0) {
            if (prev == NULL) {
                list->head = current->next;
            } else {
                prev->next = current->next;
            }
            free(current->data);
            free(current);
            return;
        }
        prev = current;
        current = current->next;
    }
    printf("Node not found\n");
}

// Print the list
void string_list_print(string_list_t* list)
{
    if (!list->head) {
        printf("List is empty\n");
        return;
    }
    struct node* current = list->head;
    while (current != NULL) {
        printf("%s\n", current->data);
        current = current->next;
    }
}

// Main function
int main()
{
    string_list_t list = {0};
    string_list_insert_at_begin(&list, "1");
    string_list_insert_at_end(&list, "2");
    string_list_insert_at_end(&list, "3");
    string_list_delete(&list, "1");
    string_list_print(&list);
    return 0;
}
```

Рис. 4: Код 4

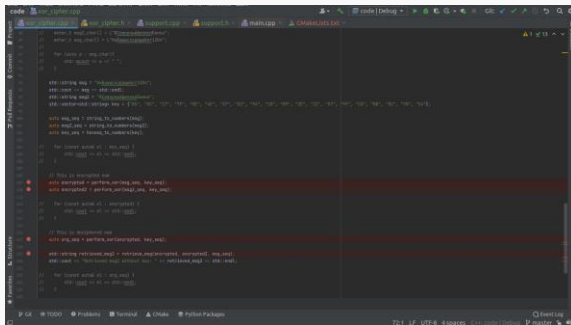


Рис. 5: Код 5

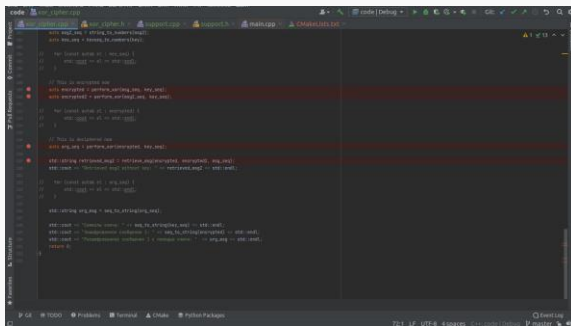


Рис. 6: Код 6

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