Презентация по лабораторной работе №7

Элементы криптографии. Однократное гаммирование

Ле Тиен Винь

Содержание

I.Цель работы 1
II. Задание 1
III. Выполнение задания 1
Код приложения 1
Анализ кода 2
Результат программы 4
IV. Вывод 4

I.Цель работы

Освоить на практике применение режима однократного гаммирования.

II. Задание

Нужно подобрать ключ, чтобы получить сообщение «С Новым Годом, друзья!». Требуется разработать приложение, позволяющее шифровать и дешифровать данные в режиме однократного гаммирования. Приложение должно:

1. Определить вид шифротекста при известном ключе и известном открытом тексте.
2. Определить ключ, с помощью которого шифротекст может быть преобразован в некоторый фрагмент текста, представляющий собой один из возможных вариантов прочтения открытого текста.

III. Выполнение задания

Код приложения

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

```
string xorOperator(const string &input, const string &key) {
    string output = input;
    for (size_t i = 0; i < input.size(); ++i) {
        output[i] = input[i] ^ key[i % key.size()];
    }
    return output;
}

string determineKey(const string &ciphertext, const string &known_plaintext) {
    string key = "";
    for (size_t i = 0; i < ciphertext.size() && i < known_plaintext.size(); ++i) {
        key += ciphertext[i] ^ known_plaintext[i];
    }
    return key;
}

int main() {
    string plaintext;
    cout << "Import text: "; cin >> plaintext;
    string key;
    cout << "Import key: "; cin >> key;

    string ciphertext = xorOperator(plaintext, key);

    cout << "Ciphertext (Hex): ";
    for (char c : ciphertext) {
        cout << hex << static_cast<int>(c) << " ";
    }
    cout << endl;

    string decryptedtext = xorOperator(ciphertext, key);
    cout << "Decrypted Text: " << decryptedtext << endl;

    string example_ciphertext = ciphertext;
    string example_plaintext_fragment = plaintext;

    string extracted_key = determineKey(example_ciphertext, example_plaintext_fragment);
    cout << "Extracted Key: " << extracted_key << endl;

    return 0;
}
```

Анализ кода

- Мы используем метод шифрования: Выполнение операции сложения по модулю 2 (XOR). Поскольку такой метод шифрования является симметричным, так как двойное прибавление одной и той же величины по модулю 2 восстанавливает исходное значение, а шифрование и расшифрование выполняется одной и той же программой.

```
string xorOperator(const string &input, const string &key) {
    string output = input;
    for (size_t i = 0; i < input.size(); ++i) {
        output[i] = input[i] ^ key[i % key.size()];
    }
    return output;
}
```

- Функция преобразует каждый элемент введенного текста в новый элемент, зашифрованный на основе ключа, с помощью операцией сложения по модулю 2 (XOR): Ci = Pi + Ki (1).

- Где Ci — i-й символ получившегося зашифрованного послания, Pi — i-й символ открытого текста, Ki — i-й символ ключа, i = 1, …, m.

```
string determineKey(const string &ciphertext, const string &known_plaintext) {
    string key = "";
    for (size_t i = 0; i < ciphertext.size() && i < known_plaintext.size(); ++i) {
        key += ciphertext[i] ^ known_plaintext[i];
    }
    return key;
}
```

- Функция определяет ключ, когда известен открытый текст и зашифрованный текст, на основе XOR: Ki = Ci + Pi.

- Если известны шифротекст и открытый текст, то задача нахождения ключа решается также в соответствии с (1), а именно, обе части равенства необходимо сложить по модулю 2 с Pi: Ci + Pi = Pi + Ki + Pi = Ki.

```
int main() {
    string plaintext;
    cout << "Import text: "; cin >> plaintext;
    string key;
    cout << "Import key: "; cin >> key;

    string ciphertext = xorOperator(plaintext, key);

    cout << "Ciphertext (Hex): ";
    for (char c : ciphertext) {
        cout << hex << static_cast<int>(c) << " ";
    }
    cout << endl;

    string decryptedtext = xorOperator(ciphertext, key);
    cout << "Decrypted Text: " << decryptedtext << endl;

    string example_ciphertext = ciphertext;
    string example_plaintext_fragment = plaintext;
```

```
    string extracted_key = determineKey(example_ciphertext, example_plaintext_fragment);
    cout << "Extracted Key: " << extracted_key << endl;

    return 0;
}
```

- В main мы будем собирать данные с клавиатуры.
- Использовать функцию “xorOperator” для генерации зашифрованного текста и вывода зашифрованного текста на экран.
- Использовать функцию “xorOperator”, чтобы расшифровать зашифрованный текст и вывести исходный текст на экран.
- Использовать ранее созданный зашифрованный текст и исходный текст, чтобы найти ключ и вывести исходный текст на экран.

Результат программы

```
PS C:\Users\DELL\Desktop> .\xor_cipher.exe
Import text: HappyNewYear,Friend!
Import key: HappyNewYear,YouToo!
Ciphertext (Hex): 0 0 0 0 0 0 0 0 0 0 0 0 0 1f 1d 1c 31 1 b 0
Decrypted Text: HappyNewYear,Friend!
Extracted Key: HappyNewYear,YouToo!
PS C:\Users\DELL\Desktop>
```

IV. Вывод

После лабораторной работы я получил практические навыки по применение режима однократного гаммирования.