Отчет по лабораторной работе 7

Элементы криптографии. Однократное гаммирование

Шалыгин Георгий Эдуардович

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Освоить на практике применение режима однократного гаммирования.

2 Теоретическое введение

Гамми́рование, или Шифр XOR, — метод симметричного шифрования, заключающийся в «наложении» последовательности, состоящей из случайных чисел, на открытый текст. Последовательность случайных чисел называется гамма-последовательностью и используется для зашифровывания и расшифровывания данных. Суммирование обычно выполняется в каком-либо конечном поле. Например, в поле Галуа $GF(2)$ суммирование принимает вид операции «исключающее ИЛИ (XOR)».

Для шифрования каждого нового сообщения нужно использовать новую гамму. Повторное использование гаммы недопустимо ввиду свойств операции «xor». Рассмотрим пример: с помощью одинаковой гаммы Y зашифрованы два открытых текста X_1 и X_2, получено две шифрограммы Z_1 и Z_2:

$Z_1 = X_1 \oplus Y$

$Z_2 = X_2 \oplus Y$

Выполним сложение двух шифрограмм, используя операцию xor:

$Z_1 \oplus Z_2 = (X_1 \oplus Y) \oplus (X_2 \oplus Y) = X_1 \oplus X_2$

Результат зависит от открытых текстов X_1 и X_2 и не зависит от гаммы Y. Ввиду избыточности естественных языков результат поддаётся частотному анализу, то есть открытые тексты можно подобрать, не зная гамму Y.

Для формирования гаммы (последовательности псевдослучайных чисел) нужно использовать аппаратные генераторы случайных чисел, основанные на физических процессах. Если гамма не будет случайной, для получения открытого текста потребуется подобрать только начальное состояние (англ. seed) генера-

тора псевдослучайных чисел. Длина гаммы должна быть не меньше длины защищаемого сообщения (открытого текста). В противном случае для получения открытого текста потребуется подобрать длину гаммы, проанализировать блоки шифротекста угаданной длины, подобрать биты гаммы.

Подробнее [2].

3 Выполнение лабораторной работы

1. Напишем функцию наложения гаммы (fig. 3.1).

```
string gamma(string dtext, string key){
    string etext = "";
    for(int i = 0; i < dtext.size(); i++){
        char c1 = dtext[i];
        char c2 = key[i];
        etext.push_back((c1 ^ c2));
    }
    return etext;
}
```

Рис. 3.1: Функция гаммирования

2. Для тестирования напишем следующий код, расшифровывающий текст и находящий ключ (fig. 3.2).

```
int main()
{
    string s = "Happy new year friends!";
    string s2 = "']W8R(IANRELAq%*FKC=!@";
    string key = "fF-'ArF,6n+ -3QCX/.-YRa";
    cout << "Text: " << s << '\n';
    cout << "Key: " << key << '\n';
    cout << "Open text: " << gamma(s, key) << '\n';
    cout << "Decoded: " << gamma(s2, key) << '\n';
    cout << "Find key: " << gamma(s, s2) << '\n';

}
```

Рис. 3.2: Тестирующий код

3. Убедимся в корректности результатов выполнения программы(fig. 3.3).

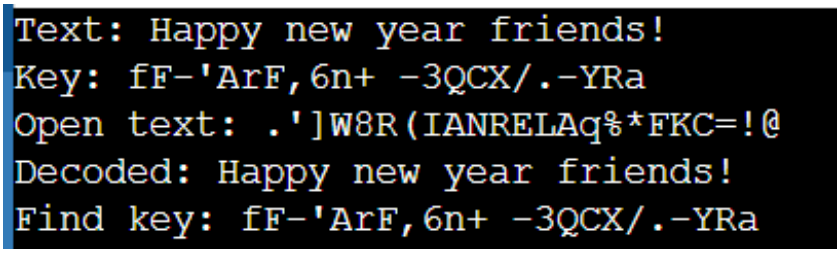

Рис. 3.3: Результаты выполнения

4 Выводы

В результате выполнения работы мы освоили на практике применение режима однократного гаммирования.

Список литературы

1. Ященко под ред. Введение в криптографию. Litres, 2017. 349 с.

2. Иванов В. Лекции о криптографии. . Яндекс, 2010.