РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №3

Тема: Шифрование гаммированием

дисциплина: Математические основы защиты информации и информационной безопасности

Студент: Койфман Кирилл Дмитриевич

Группа: НФИмд-01-25

Введение

Цель работы

Получение практических навыков реализации алгоритмов, использующих гаммирование.

Задачи

1. Реализовать алгоритм шифрования гаммированием конечной гаммой

Теория

В основе функционирования шифров простой замены лежит следующий принцип: формируется m-разрядная случайная двоичная последовательность — ключ шифра. Отправитель производит побитовое сложение по модулю два (mod 2) ключа $$ k = k_1k_2...k_i...k_m $$ и m-разрядной

двоичной последовательности $$ p = p_1p_2...p_i...p_m, $$ соответствующей посылаемому сообщению: $$ c_i = p_i\oplus k_i, i = \overline{1,m}, $$ где p_i-i-й бит исходного текста, k_i-i-й бит ключа, $\oplus-$ операция побитового сложения (XOR), c_i-i-й бит получившейся криптограммы $$ c=c_1c_2...c_i...c_m. $$

Операция побитного сложения является обратимой, т.е. $(x\oplusy)\oplusy = x$, поэтому дешифрование осуществляется повторным применением операции $\oplus$ к криптограмме: $$ p_i=c_i\oplus k_i,i=\overline{1,m}, $$

Однако основным недостатком такой схемы является равенство объёма ключевой информации и суммарного объёма передаваемых сообщений. Данный недостаток можно убрать, использовав ключ в качестве "зародыша", порождающего значительно более длинную ключевую последовательность.

Гаммированием называют процедуру наложения при помощи некоторой функции F на исходный текст гаммы шифра, т.е. псевдослучайной последовательности (ПСП) с выходов генератора G. Псевдослучайная последовательность по своим статистическим свойствам неотличима от случайной последовательности, но является детерминированной, т.е. известен алгоритм ее формирования. Обычно в качестве функции F берется операция поразрядного сложения по модулю два или по модулю N (N число букв алфавита открытого текста).

Простейший генератор псевдослучайной последовательности можно представить рекуррентным соотношением: $$ \gamma_i=\alpha*\gamma_{i-1}+b\bmod(m), i=\overline{1,m}, $$

где γ_i-i-й член последовательности псевдослучайных чисел, a, γ_i, b — ключевые параметры. Такая последовательность состоит из целых чисел от 0 до $m - 1$. Если элементы γ_i и γ_j совпадут, то совпадут и последующие участки: $\gamma_{i+1}=\gamma_{j+1}$, $\gamma_{i+2}=\gamma_{j+2}$. Таким образом, ПСП является периодической. Знание периода гаммы существенно облегчает криптоанализ. Максимальная длина периода равна m. Для ее достижения необходимо удовлетворить следующим условиям:

1. b и m — взаимно простые числа;
2. $a - 1$ делится на любой простой делитель числа m;
3. $a - 1$ кратно 4, если m кратно 4.

Стойкость шифров, основанных на процедуре гаммирования, зависит от характеристик гаммы — длины и равномерности распределения вероятностей появления знаков гаммы.

При использовании генератора ПСП получаем бесконечную гамму. Однако, возможен режим шифрования конечной гаммы. В роли конечной гаммы может выступать фраза. Как и ранее, используется алфавитный порядок букв, т.е. буква "а" имеет порядковый номер 1, "б" - 2 и т.д.

Ход работы

Для решения поставленной задачи реализуем алгоритм шифрования гаммированием конечной гаммой на языке программирования C++ (Листинг-1), а также проведём тест данного алгоритма, чтобы проверить корректность его работы (Листинг-2):

```
#include <iostream>
#include <cstdint>
#include <map>

std::wostream& operator<<(std::wostream& out, const std::map<wchar_t,
std::uint32_t>& alphabet)
{
    out << "Alphabet:\n";
    int i = 0;
    for (const auto& pair : alphabet)
    {
        out << '[' << pair.first << ']' << '{' << pair.second << '}' <<
'\t';
        if (++i % 10 == 0)
            out << '\n';
    }

    return out;
}

void printTextCodes(const std::wstring& enteredText, const
std::map<wchar_t, std::uint32_t>& alphabet)
{
    for (const auto& symbol : enteredText)
    {
        std::wcout << alphabet.at(symbol) << '\t';
    }
}

int main()
{
    setlocale(LC_ALL, "");

    //Define an alphabet with use of map
    const wchar_t alphabetBeginSymbol = L'A';
    const wchar_t alphabetEndSymbol = L'Я';
    const int alphabethLength = alphabetEndSymbol - alphabetBeginSymbol +
1;
    std::map<wchar_t, std::uint32_t> alphabet{};
    for (std::uint32_t symbol_index = 1; symbol_index <= alphabethLength;
++symbol_index)
    {
        alphabet.insert({ alphabetBeginSymbol + (symbol_index - 1),
symbol_index });
    }
    std::wcout << alphabet << '\n';
    std::wcout << "Alphabet length: " << alphabethLength << '\n';

    //Define input open message
    std::wstring enteredMessage = L"ПРИКАЗ";
    size_t enteredMessageLength = enteredMessage.size();
    //std::wcin >> enteredMessage;
    std::wcout << "Entered message: " << enteredMessage << '\n';
```

```
    std::wcout << "Entered message(codes):\n";
    printTextCodes(enteredMessage, alphabet);
    std::wcout << '\n';

    //Define input gamma
    std::wstring enteredGamma = L"ГАММА";
    size_t enteredGammaLength = enteredGamma.size();
    //!std::wcin >> enteredGamma;
    std::wcout << "Entered gamma: " << enteredGamma << '\n';
    std::wcout << "Entered gamma(codes):\n";
    printTextCodes(enteredGamma, alphabet);
    std::wcout << '\n';

    //Perform encrypting (by applying gamma)
    std::wcout << "-----------------------------------ENCRYPTING----------
------------------------\n";
    std::wstring encryptedMessage;

    for (std::uint32_t symbol_index = 0; symbol_index <
enteredMessageLength; ++symbol_index)
    {
        //ENCRYPTED_SYMBOL_CODE = ENTERED_MESSAGE_CODE + ENTERED_GAMMA %
ALPHABET_LENGTH
        std::uint32_t encryptedSymbolCode =
(alphabet.at(enteredGamma[symbol_index % enteredGammaLength]) +
            (alphabet.at(enteredMessage[symbol_index]) % alphabethLength))
% alphabethLength;

        for (auto& element : alphabet)
            if (element.second == encryptedSymbolCode)
            {
                encryptedMessage += element.first;
                break;
            }
    }

    std::wcout << "Encrypted message: " << encryptedMessage << '\n';
    std::wcout << "Encrypted message(codes):\n";
    printTextCodes(encryptedMessage, alphabet);
    std::wcout << '\n';
}
```

Листинг-2(алгоритм, реализующий метод шифрования гаммирования конечной гаммой)

```
[А]{1}  [Б]{2}  [В]{3}  [Г]{4}  [Д]{5}  [Е]{6}  [Ж]{7}  [З]{8}  [И]{9}
[Й]{10}
[К]{11} [Л]{12} [М]{13} [Н]{14} [О]{15} [П]{16} [Р]{17} [С]{18} [Т]{19}
[У]{20}
[Ф]{21} [Х]{22} [Ц]{23} [Ч]{24} [Ш]{25} [Щ]{26} [Ъ]{27} [Ы]{28} [Ь]{29}
[Э]{30}
```

```
[Ю]{31} [Я]{32}
Alphabet length: 32
Entered message: ПРИКАЗ
Entered message(codes):
16      17      9       11      1       8
Entered gamma: ГАММА
Entered gamma(codes):
4       1       13      13      1
----------------------------------ENCRYPTING----------------------------
------
Encrypted message: УСХЧБЛ
Encrypted message(codes):
20      18      22      24      2       12
```

Листинг-2(результаты работы алгоритма шифрования гаммирования конечной гаммой)

Исходя из полученных результатов (Листинг-2), можно судить о том, что данный алгоритм производит успешные шифрование вводимого текста, последовательно накладывая элементы гаммы шифра на каждый символ исходного текста.

Заключение

В ходе проделанной лабораторной работы мной были усвоены знания по принципам работы с шифрами гаммирования, а также получены навыки по их реализации.