МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования



НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

ОТЧЁТ

по лабораторной работе №2

по дисциплине

Защита информации

Вариант 5

РУКОВОДИТЕЛЬ:

Капранов С.Н.

СТУДЕНТ:

Докукин Д.В.

ГРУППА:

20-ПО

Нижний Новгород 2023

Задание

Лабораторная работа 2 – Шифрование методом гаммирования

Задание №2.1

Реализовать алгоритм шифрования данных «Шифрование методом гаммирования»  
По модулю 2

Введение

Для данного алгоритма, ограничений по символам нет. Его концепция позволяет рассматривать символы, как набор байт, легко интерпретируемых в числовые значения. Это позволяет шифровать любые данные любым ключом. При реализации алгоритма, я ограничился текстовыми данными.

Описание алгоритма

Гаммирование. В этом способе шифрование выполняется путем сложения символов исходного текста и ключа по модулю, равному числу букв в алфавите. Если в исходном алфавите, например, 33 символа, то сложение производится по модулю 33. Такой процесс сложения исходного текста и ключа называется в криптографии наложением гаммы.

Пусть символам исходного алфавита соответствуют числа от 0 (А) до 32 (Я). Если обозначить число, соответствующее исходному символу, x, а символу ключа – k, то можно записать правило гаммирования следующим образом:

z = x + k (mod N),

где z – закодированный символ, N - количество символов в алфавите, а сложение по модулю N - операция, аналогичная обычному сложению, с тем отличием, что если обычное суммирование дает результат, больший или равный N, то значением суммы считается остаток от деления его на N. Например, пусть сложим по модулю 33 символы Г (3) и Ю (31):

3 + 31 (mod 33) = 1,

то есть в результате получаем символ Б, соответствующий числу 1.

Наиболее часто на практике встречается двоичное гаммирование. При этом используется двоичный алфавит, а сложение производится по модулю два. Операция сложения по модулю 2 часто обозначается , то есть можно записать:



Операция сложения по модулю два в алгебре логики называется также "исключающее ИЛИ" или по-английски XOR.

Демонстрация работы программы

Для демонстрации, согласно требованиям, будут зашифрованы следующие сообщения:

1. Нижегородский Государственный Технический Университет
2. Докукин Дмитрий Владимирович

В качестве ключа использована последовательность:

abcdef123456

Зашифрованное сообщение получается в формате бинарного файла.

Синтакс программы:

./gamming encrypt|decrypt <source\_file\_path> [<destination\_file\_path>] [-k=\"<key\_file\_path>\"]

Первым аргументом идет имя программы.

Далее, тип операции.

Третьим имя(путь) шифруемого/расшифровываемого файла.

Четвёртый служит именем файла, куда будет записан шифртекст.

Пятый аргумент указывает на файл с ключом.

Пример 1

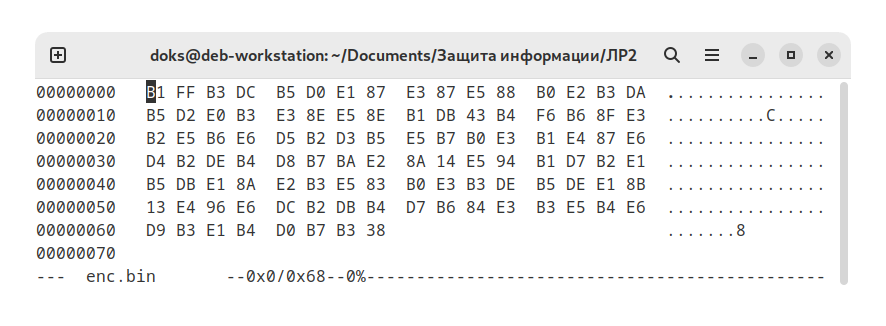
Шифрование сообщения: Нижегородский Государственный Технический Университет

В файле source.txt записано сообщение. В файле key.txt записан ключ. Шифртекст будет записан в файл en.bin

Следующая команда вызывает шифрование указанного сообщения:

./gamming encrypt source.txt enc.bin -k=key.txt

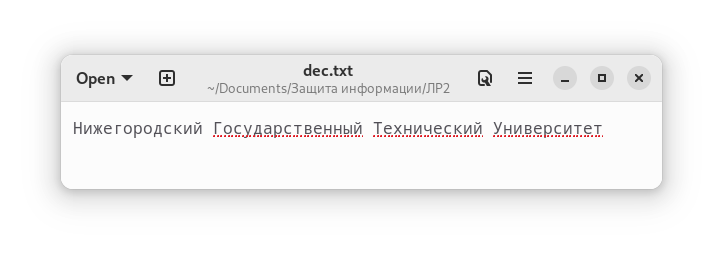
С помощью консольной утилиты hexedit можно посмотреть содержимое зашифрованного файла:



При расшифровке, указываю операцию расшифровки, файл с шифртекстом (enc.bin),имя файла для расшифрованного сообщения (dec.txt) и файл с ключом (key.txt):

./gamming decrypt enc.bin dec.txt -k=key.txt

Содержимое файла dec.txt:

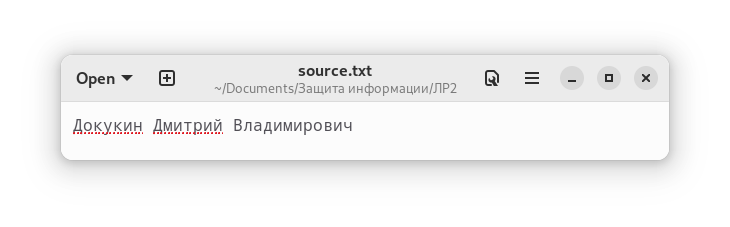


Можно наблюдать, что расшифровка прошла успешно.

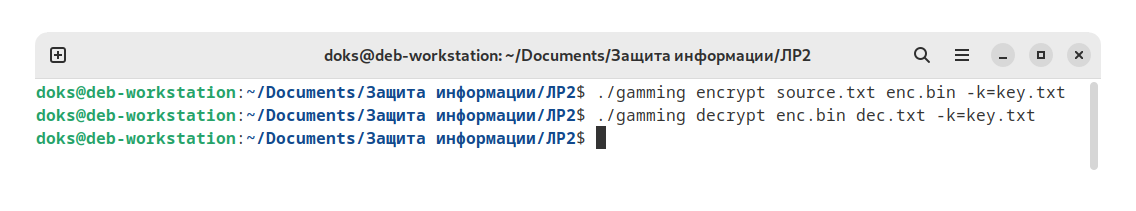
Пример 2

Шифрование и расшифровка сообщения: Докукин Дмитрий Владимирович

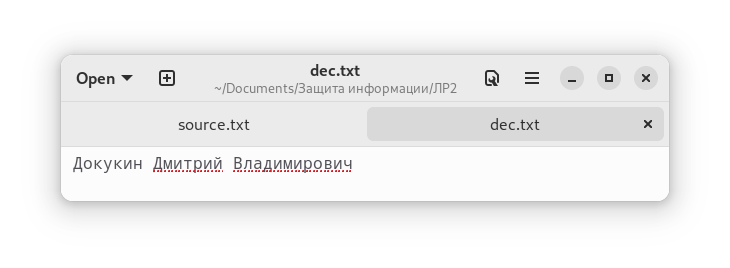
Содержимое файла source.txt:



Обе операции показаны на одном скриншоте:



Содержимое файла dec.txt:



Процессы прошли успешно.

Программа

В программе представлены два класса:

InputHandler – отвечает за обработку ввода. Проверяет входные данные на корректность, хранит в себе имена файлов, а также имеет методы извлечения ключа из файла.

Gamming – отвечает за шифрование и расшифровку. Хранит в себе значение ключа. Имеет методы шифрования и расшифрования.

Экземпляры данных классов объявляются в файле main.cpp, где происходит последовательный вызов нужных методов для выполнения требуемой операции.

Листинг

Файл Gamming.h

#ifndef GRONSFELD\_H

#define GRONSFELD\_H

#include <vector>

#include <string>

#include <fstream>

class Gamming

{

private:

std::string m\_key;

public:

Gamming(std::string key) : m\_key(key) {}

void encrypt(char\* msg, size\_t msgSize);

void decrypt(char\* msg, size\_t msgSize);

};

#endif // GRONSFELD\_H

Файл InputHandler.h

#ifndef INPUTHANDLER\_H

#define INPUTHANDLER\_H

#include <string>

#include <fstream>

#include <vector>

class InputHandler

{

public:

enum Status

{

NONE, // None

S, // Only the Source was entered

SD, // Source and Destination were entered

SK, // Source and Key were entered

SDK, // Source, Destination and key were entered

INVALID // Invalid input

};

enum Command

{

ENCRYPT,

DECRYPT

};

private:

std::string m\_sourceName;

std::string m\_destinationName;

std::string m\_key;

Status m\_status;

Command m\_command;

bool isKey(char\* str);

Status checkArgs(int argc, char\* argv[]);

bool isKeyCorrect(const std::string key);

std::string getKeyPath(char\* arg);

std::string extractKey(const std::string keyPath);

public:

InputHandler(): m\_status(Status::NONE) {}

bool process(int argc, char\* argv[]);

void printMessage();

std::string getSource() const;

std::string getDestination() const;

std::string getKey() const;

Command getCommand() const;

~InputHandler() {}

};

#endif // INPUTHANDLER\_H

Файл main.cpp

#include <iostream>

#include "InputHandler.h"

#include "Gamming.h"

int main (int argc, char\* argv[])

{

InputHandler ih;

if (!ih.process(argc, argv))

{

ih.printMessage();

return 0;

}

std::ifstream source;

std::ofstream destination;

if (ih.getCommand() == InputHandler::Command::ENCRYPT)

{

source.open(ih.getSource(), std::ios\_base::in);

destination.open(ih.getDestination(), std::ios\_base::out | std::ios\_base::binary);

}

else if (ih.getCommand() == InputHandler::Command::DECRYPT)

{

source.open(ih.getSource(), std::ios\_base::in | std::ios\_base::binary);

destination.open(ih.getDestination(), std::ios\_base::out);

}

if (!source.is\_open())

{

std::cout << "Error opening source\n";

return 0;

}

if (!destination.is\_open())

{

std::cout << "Error creating dest\n";

return 0;

}

source.seekg(0, source.end);

size\_t sSize = source.tellg();

source.seekg(0, source.beg);

char\* buff = new char[sSize + 1];

source.read(buff, sSize);

source.close();

buff[sSize] = '\0';

Gamming gm(ih.getKey());

if (ih.getCommand() == InputHandler::Command::ENCRYPT)

gm.encrypt(buff, sSize);

else if (ih.getCommand() == InputHandler::Command::DECRYPT)

gm.decrypt(buff, sSize);

destination.write(buff, sSize);

destination.close();

delete[] buff;

return 0;

}

Файл Gamming.cpp

#include "Gamming.h"

void Gamming::encrypt(char\* msg, size\_t msgSize)

{

for (int i = 0, j = 0; i < msgSize; ++i)

{

msg[i] = (msg[i] ^ m\_key[j]);

if (++j == m\_key.size())

j = 0;

}

}

void Gamming::decrypt(char\* msg, size\_t msgSize)

{

for (int i = 0, j = 0; i < msgSize; ++i)

{

msg[i] = (msg[i] ^ m\_key[j]);

if (++j == m\_key.size())

j = 0;

}

}

Файл InputHandler.cpp

#include "InputHandler.h"

#include <cstring>

#include <iostream>

bool InputHandler::isKey(char\* str)

{

if ( (\*(str) == '-')

&& (\*(str + 1) == 'k')

&& (\*(str + 2) == '='))

{

return true;

} else

return false;

}

InputHandler::Status InputHandler::checkArgs(int argc, char\* argv[])

{

if (argc < 3 || argc > 5)

return Status::INVALID;

if (strcmp(argv[1], "encrypt") == 0)

m\_command = Command::ENCRYPT;

else if (strcmp(argv[1], "decrypt") == 0)

m\_command = Command::DECRYPT;

else

return Status::INVALID;

Status result {Status::NONE};

for (int i = 2; i < argc - 1; ++i)

{

if (isKey(argv[i]))

{

result = Status::INVALID;

break;

}

}

if (result != INVALID && isKey(argv[argc - 1]))

{

if (argc == 4)

result = Status::SK;

if (argc == 5)

result = Status::SDK;

}

return result;

}

std::string InputHandler::getKeyPath(char\* arg)

{

std::string path;

for (int i = 3; i < strlen(arg); ++i)

{

path += \*(arg + i);

}

return path;

}

std::vector<unsigned int> InputHandler::extractKey(const std::string keyPath)

{

std::ifstream keyFile;

std::vector<unsigned int> key;

keyFile.open(keyPath, std::ios\_base::in);

if (!keyFile.is\_open())

{

std::cout << "Invalid key path\n";

return key;

}

char ch = keyFile.get();

while (ch != EOF)

{

if (ch >= '0' && ch <= '9')

{

key.push\_back(ch - '0');

ch = keyFile.get();

} else if (ch == '\n' || ch == ' ')

{

ch = keyFile.get();

} else

{

key.clear();

break;

}

}

keyFile.close();

return key;

}

std::vector<unsigned int> InputHandler::convertKey(const std::string keyString)

{

std::vector<unsigned int> key;

for (const auto ch : keyString)

{

if (!(ch >= '0') && !(ch <= '9'))

return std::vector<unsigned int>();

key.push\_back(ch - '0');

}

return key;

}

bool InputHandler::process(int argc, char\* argv[])

{

using std::string, std::cout, std::cin;

Status result = checkArgs(argc, argv);

if (result == Status::INVALID)

{

return false;

}

if ((result == Status::SK) || (result == Status::SDK))

{

string keyFile = getKeyPath(argv[argc - 1]);

m\_key = extractKey(keyFile);

}

else

{

cout << "Enter key: ";

std::string keyString;

cin >> keyString;

m\_key = convertKey(keyString);

}

if (m\_key.size() == 0)

{

return false;

}

m\_sourceName = argv[2];

if (result == Status::SD || result == Status::SDK)

m\_destinationName = argv[3];

else

{

switch(m\_command)

{

case Command::ENCRYPT:

m\_destinationName = "encrypted\_" + m\_sourceName;

break;

case Command::DECRYPT:

m\_destinationName = "decrypted\_" + m\_sourceName;

break;

}

}

cout << "argc = " << argc << '\n';

cout << "Source = " << m\_sourceName << '\n';

cout << "Destination = " << m\_destinationName << '\n';

cout << "Key: ";

for (auto el : m\_key)

cout << el;

cout << '\n';

return true;

}

void InputHandler::printMessage()

{

char syntax[] = "Syntax:\n" \

"./gamming encrypt|decrypt <source\_file\_path> [<destination\_file\_path>] [-k=\"<key\_file\_path>\"]\n";

std::cout << syntax;

}

std::string InputHandler::getSource() const

{

return m\_sourceName;

}

std::string InputHandler::getDestination() const

{

return m\_destinationName;

}

std::vector<unsigned int> InputHandler::getKey() const

{

return m\_key;

}

InputHandler::Command InputHandler::getCommand() const

{

return m\_command;

}