**Міністерство Освіти І НАУКИ України**

**Національний університет "Львівська політехніка"**

Інститут **КНІТ**

Кафедра **ПЗ**

### ЗВІТ

До лабораторної роботи № 2

**З дисципліни:** *“Безпека програм та даних”*

**На тему:** *“Створення програмного засобу для забезпечення*

*цілісності інформації”*

**Лектор:**

доц. каф. ПЗ

Сенів М. М.

**Виконав:**

ст. гр. ПЗ-44

Вонс Ю. А.

**Прийняв:**

асист. каф. ПЗ

Самбір А. А.

« \_\_\_\_ » \_\_\_\_\_\_\_\_ 2019 р.

∑= \_\_\_\_\_ .

Львів – 2019

**Тема роботи:** Створення програмного засобу для забезпечення цілісності інформації.

**Мета роботи:** Ознайомитись з методами криптографічного забезпечення цілісності інформації, навчитись створювати програмні засоби для забезпечення цілісності інформації з використанням алгоритмів хешування.

**TЕОРЕТИЧНІ ВІДОМОСТІ**

Хеш-функція – це функція, що перетворює вхідні дані будь-якого розміру в дані фіксованого розміру. Таким чином, функція хешування отримує на вхід повідомлення M довільної довжини, а на вихід видає хеш-код H(M) фіксованого розміру, який іноді називають профілем повідомлення. Хеш-код є функцією усіх бітів повідомлення і забезпечує можливість контролю помилок: зміна будь-якої кількості бітів повідомлення призводить до зміни хеш-коду.

Одним із популярних захищених алгоритмів хешування є алгоритм MD5, який був розроблений Роном Райвестом з Масачусетського технологічного інституту на початку 90-х років. Цей алгоритм широко використовується в операційних системах таких як: Windows, Linux, BSD, PGP (система захисту інформації).

Основним використанням даного алгоритму є криптографічні контрольні суми, наприклад для перевірки цілісності отриманих з мережі Інтернет файлів. Криптографічна хеш-функція дозволяє легко перевірити, що деякі вхідні дані зіставляються із заданим значенням хешу, але, якщо вхідні дані невідомі, то навмисно важко відновити вхідне значення знаючи збережене значення хеш-функції. Це використовується для забезпечення цілісності переданих даних.

**ЗАВДАННЯ**

Створити програмну реалізацію алгоритму хешування MD5. Тестування створеної програми провести з використання наступних тестових значень хешу:

* H() = D41D8CD98F00B204E9800998ECF8427E
* H(a) = 0CC175B9C0F1B6A831C399E269772661
* H(abc) = 900150983CD24FB0D6963F7D28E17F72
* H(message digest) = F96B697D7CB7938D525A2F31AAF161D0
* H(abcdefghijklmnopqrstuvwxyz) = C3FCD3D76192E4007DFB496CCA67E13B
* H(ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789) = D174AB98D277D9F5A5611C2C9F419D9F
* H(12345678901234567890123456789012345678901234567890123456789012345678901234567890) = 57EDF4A22BE3C955AC49DA2E2107B67A

Програмна реалізація повинна виводити значення хешу як для рядка, заданого в полі вводу, так і для файлу. Результат роботи програми повинен відображатись на екрані з можливістю наступного запису в файл. Крім того програма повинна мати можливість перевірити цілісність будь-якого файлу за наявним файлом з MD5 хешем, записаним у шістнадцятковому форматі. У звіті навести протокол тестування і роботи програми та зробити висновки.

**ТЕКСТ ПРОГРАМИ**

1. **MD5.h**

class MD5

{

public:

ui\* T = new ui[64]

{

0xd76aa478, 0xe8c7b756, 0x242070db, 0xc1bdceee,

0xf57c0faf, 0x4787c62a, 0xa8304613, 0xfd469501,

0x698098d8, 0x8b44f7af, 0xffff5bb1, 0x895cd7be,

0x6b901122, 0xfd987193, 0xa679438e, 0x49b40821,

0xf61e2562, 0xc040b340, 0x265e5a51, 0xe9b6c7aa,

0xd62f105d, 0x02441453, 0xd8a1e681, 0xe7d3fbc8,

0x21e1cde6, 0xc33707d6, 0xf4d50d87, 0x455a14ed,

0xa9e3e905, 0xfcefa3f8, 0x676f02d9, 0x8d2a4c8a,

0xfffa3942, 0x8771f681, 0x6d9d6122, 0xfde5380c,

0xa4beea44, 0x4bdecfa9, 0xf6bb4b60, 0xbebfbc70,

0x289b7ec6, 0xeaa127fa, 0xd4ef3085, 0x04881d05,

0xd9d4d039, 0xe6db99e5, 0x1fa27cf8, 0xc4ac5665,

0xf4292244, 0x432aff97, 0xab9423a7, 0xfc93a039,

0x655b59c3, 0x8f0ccc92, 0xffeff47d, 0x85845dd1,

0x6fa87e4f, 0xfe2ce6e0, 0xa3014314, 0x4e0811a1,

0xf7537e82, 0xbd3af235, 0x2ad7d2bb, 0xeb86d391

};

MD5() = delete;

MD5(string text);

void update(const unsigned char\* buf, ui length);

MD5& convert\_128();

string result() const;

private:

void compression(const uc block[64]);

uc tmp\_buf[64];

ui counter[2] = { 0 };

ui registers[4] = { 0x67452301, 0xefcdab89 , 0x98badcfe, 0x10325476 };

uc results[16];

ui F(ui B, ui C, ui D);

ui G(ui B, ui C, ui D);

ui H(ui B, ui C, ui D);

ui I(ui B, ui C, ui D);

ui rotate\_left(ui value, ui n);

void FF(ui& A, ui B, ui C, ui D, ui x, ui count, ui t);

void GG(ui& A, ui B, ui C, ui D, ui x, ui count, ui t);

void HH(ui& A, ui B, ui C, ui D, ui x, ui count, ui t);

void II(ui& A, ui B, ui C, ui D, ui x, ui count, ui t);

};

1. **MD5.cpp**

#include "md5.h"

ui MD5::F(ui B, ui C, ui D) {

return (B & C) | (~B & D);

}

ui MD5::G(ui B, ui C, ui D) {

return (B & D) | (C & ~D);

}

ui MD5::H(ui B, ui C, ui D) {

return B ^ C ^ D;

}

ui MD5::I(ui B, ui C, ui D) {

return C ^ (B | ~D);

}

ui MD5::rotate\_left(ui value, ui count) {

return (value << count) | (value >> (32 - count));

}

void MD5::FF(ui& A, ui B, ui C, ui D, ui x, ui count, ui t) {

A = rotate\_left(A + F(B, C, D) + x + t, count) + B;

}

void MD5::GG(ui& A, ui B, ui C, ui D, ui x, ui count, ui t) {

A = rotate\_left(A + G(B, C, D) + x + t, count) + B;

}

void MD5::HH(ui& A, ui B, ui C, ui D, ui x, ui count, ui t) {

A = rotate\_left(A + H(B, C, D) + x + t, count) + B;

}

void MD5::II(ui& A, ui B, ui C, ui D, ui x, ui count, ui t) {

A = rotate\_left(A + I(B, C, D) + x + t, count) + B;

}

MD5::MD5(string text)

{

update((const unsigned char\*)text.c\_str(), text.length());

convert\_128();

}

void MD5::compression(const uc block[64])

{

ui A = registers[0], B = registers[1], C = registers[2], D = registers[3], x[16];

for (unsigned int i = 0, j = 0; j < 64; i++, j += 4)

x[i] = ((ui)block[j]) | (((ui)block[j + 1]) << 8) | (((ui)block[j + 2]) << 16) | (((ui)block[j + 3]) << 24);

int i = 0;

FF(A, B, C, D, x[0], 7, T[i++]); FF(D, A, B, C, x[1], 12, T[i++]);

FF(C, D, A, B, x[2], 17, T[i++]); FF(B, C, D, A, x[3], 22, T[i++]);

FF(A, B, C, D, x[4], 7, T[i++]); FF(D, A, B, C, x[5], 12, T[i++]);

FF(C, D, A, B, x[6], 17, T[i++]); FF(B, C, D, A, x[7], 22, T[i++]);

FF(A, B, C, D, x[8], 7, T[i++]); FF(D, A, B, C, x[9], 12, T[i++]);

FF(C, D, A, B, x[10], 17, T[i++]);FF(B, C, D, A, x[11], 22, T[i++]);

FF(A, B, C, D, x[12], 7, T[i++]); FF(D, A, B, C, x[13], 12, T[i++]);

FF(C, D, A, B, x[14], 17, T[i++]);FF(B, C, D, A, x[15], 22, T[i++]);

GG(A, B, C, D, x[1], 5, T[i++]); GG(D, A, B, C, x[6], 9, T[i++]);

GG(C, D, A, B, x[11], 14, T[i++]);GG(B, C, D, A, x[0], 20, T[i++]);

GG(A, B, C, D, x[5], 5, T[i++]); GG(D, A, B, C, x[10], 9, T[i++]);

GG(C, D, A, B, x[15], 14, T[i++]);GG(B, C, D, A, x[4], 20, T[i++]);

GG(A, B, C, D, x[9], 5, T[i++]); GG(D, A, B, C, x[14], 9, T[i++]);

GG(C, D, A, B, x[3], 14, T[i++]); GG(B, C, D, A, x[8], 20, T[i++]);

GG(A, B, C, D, x[13], 5, T[i++]); GG(D, A, B, C, x[2], 9, T[i++]);

GG(C, D, A, B, x[7], 14, T[i++]); GG(B, C, D, A, x[12], 20, T[i++]);

HH(A, B, C, D, x[5], 4, T[i++]); HH(D, A, B, C, x[8], 11, T[i++]);

HH(C, D, A, B, x[11], 16, T[i++]);HH(B, C, D, A, x[14], 23, T[i++]);

HH(A, B, C, D, x[1], 4, T[i++]); HH(D, A, B, C, x[4], 11, T[i++]);

HH(C, D, A, B, x[7], 16, T[i++]); HH(B, C, D, A, x[10], 23, T[i++]);

HH(A, B, C, D, x[13], 4, T[i++]); HH(D, A, B, C, x[0], 11, T[i++]);

HH(C, D, A, B, x[3], 16, T[i++]); HH(B, C, D, A, x[6], 23, T[i++]);

HH(A, B, C, D, x[9], 4, T[i++]); HH(D, A, B, C, x[12], 11, T[i++]);

HH(C, D, A, B, x[15], 16, T[i++]);HH(B, C, D, A, x[2], 23, T[i++]);

II(A, B, C, D, x[0], 6, T[i++]); II(D, A, B, C, x[7], 10, T[i++]);

II(C, D, A, B, x[14], 15, T[i++]);II(B, C, D, A, x[5], 21, T[i++]);

II(A, B, C, D, x[12], 6, T[i++]); II(D, A, B, C, x[3], 10, T[i++]);

II(C, D, A, B, x[10], 15, T[i++]);II(B, C, D, A, x[1], 21, T[i++]);

II(A, B, C, D, x[8], 6, T[i++]); II(D, A, B, C, x[15], 10, T[i++]);

II(C, D, A, B, x[6], 15, T[i++]); II(B, C, D, A, x[13], 21, T[i++]);

II(A, B, C, D, x[4], 6, T[i++]); II(D, A, B, C, x[11], 10, T[i++]);

II(C, D, A, B, x[2], 15, T[i++]); II(B, C, D, A, x[9], 21, T[i++]);

registers[0] += A;

registers[1] += B;

registers[2] += C;

registers[3] += D;

memset(x, 0, sizeof x);

}

void MD5::update(const unsigned char input[], ui length)

{

ui index = counter[0] / 8 % 64;

if ((counter[0] += (length << 3)) < (length << 3))

counter[1]++;

counter[1] += (length >> 29);

ui firstpart = 64 – index, i;

if (length >= firstpart) {

memcpy(&tmp\_buf[index], input, firstpart);

compression(tmp\_buf);

for (i = firstpart; i + 64 <= length; i += 64)

compression(&input[i]);

index = 0;

}

else

i = 0;

memcpy(&tmp\_buf[index], &input[i], length - i);

}

MD5& MD5::convert\_128()

{

static unsigned char padding[64] = {

0x80, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0

};

unsigned char bits[8];

for (ui i = 0, j = 0; j < 8; i++, j += 4) {

bits[j] = counter[i] & 0xff;

bits[j + 1] = (counter[i] >> 8) & 0xff;

bits[j + 2] = (counter[i] >> 16) & 0xff;

bits[j + 3] = (counter[i] >> 24) & 0xff;

}

ui index = counter[0] / 8 % 64, padLen = (index < 56) ? (56 - index) : (120 - index);

update(padding, padLen);

update(bits, 8);

for (ui i = 0, j = 0; j < 16; i++, j += 4) {

results[j] = registers[i] & 0xff;

results[j + 1] = (registers[i] >> 8) & 0xff;

results[j + 2] = (registers[i] >> 16) & 0xff;

results[j + 3] = (registers[i] >> 24) & 0xff;

}

return \*this;

}

1. **MainWindow.cpp**

void MainWindow::on\_pushButton\_clicked()

{

string sInput = ui->lineEdit->text().toStdString();

MD5 \*obj = new MD5(sInput);

ui->lineEdit\_2->setText(QString::fromStdString(obj->result()));

delete obj;

}

void MainWindow::on\_action\_triggered()

{

QFile file("C://Users//hp//Documents//QtProjects//SD2//input.txt");

if (!file.open(QIODevice::ReadOnly | QIODevice::Text))

return ;

QString line;

QTextStream in(&file);

while (!in.atEnd())

line = in.readLine();

ui->lineEdit->setText(line);

}

void MainWindow::on\_action\_2\_triggered()

{

QFile file("C://Users//hp//Documents//QtProjects//SD2//output.txt");

if (!file.open(QIODevice::WriteOnly | QIODevice::Append | QIODevice::Text)) return;

QTextStream out(&file);

string sInput = ui->lineEdit->text().toStdString();

MD5 \*obj = new MD5(sInput);

out << QString::fromStdString(obj->result());

out << endl;

file.close();

}

void MainWindow::on\_action\_3\_triggered()

{

QString filename=QFileDialog::getOpenFileName(this, tr("Open File"), QDir::currentPath(), tr("TXT Files (\*.txt)"));

ui->lineEdit->setText(filename);

MD5 \*obj = new MD5(filename.toStdString());

QString hashfile="d09099f68243132e30a42ad4d190c568";

if(hashfile == QString::fromStdString(obj->result())){

ui->lineEdit\_2->setText("Хеш сума файлу співпадає із заданою.");

} else {

ui->lineEdit\_2->setText("Хеш сума файлу не співпадає із заданою. Спробуйте ще раз.");

}

}

**РЕЗУЛЬТАТИ**

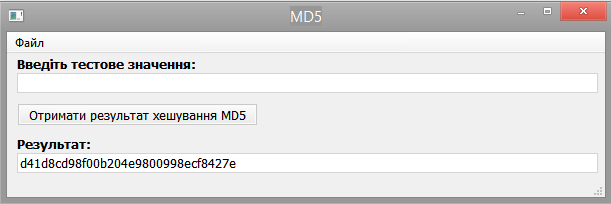


Рис. 1. Результат виконання тестового випадку 1

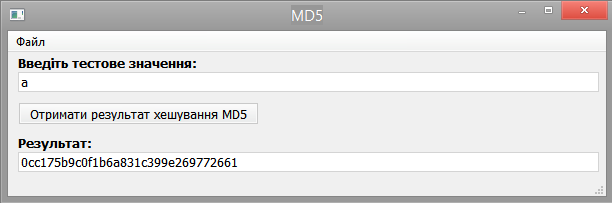


Рис. 2. Результат виконання тестового випадку 2

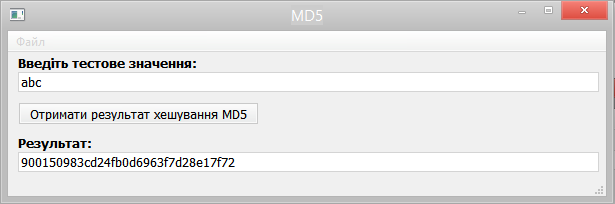


Рис. 3. Результат виконання тестового випадку 3

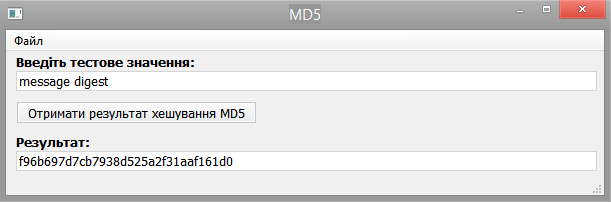


Рис. 4. Результат виконання тестового випадку 4

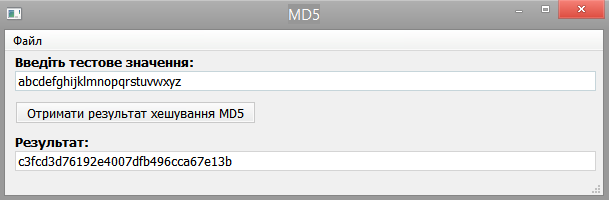


Рис. 5. Результат виконання тестового випадку 5

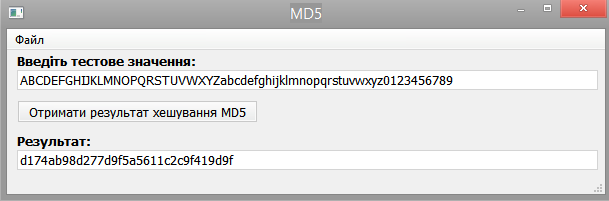


Рис. 6. Результат виконання тестового випадку 6

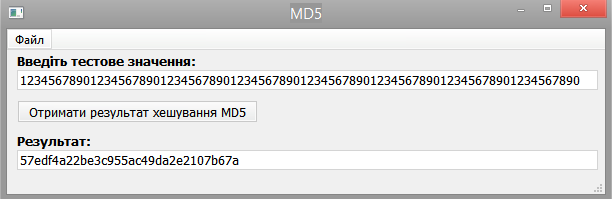


Рис. 7. Результат виконання тестового випадку 7

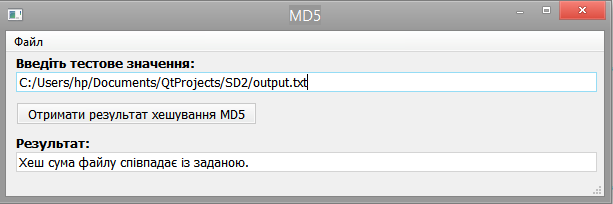


Рис. 8. Результат виконання цілісності файлу

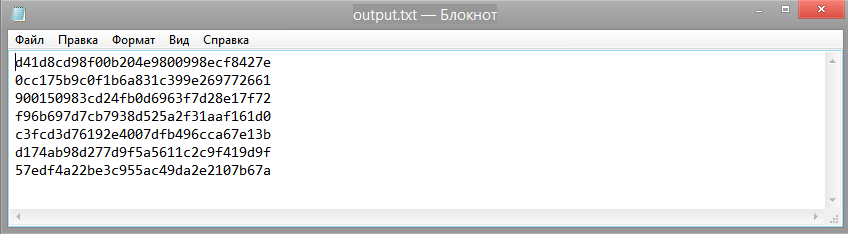


Рис. 9. Збереження результатів у файл

**ВИСНОВКИ**

Виконуючи дану лабораторну роботу, я ознайомився з методами криптографічного забезпечення цілісності інформації, навчився створювати програмні засоби для забезпечення цілісності інформації з використанням алгоритмів хешування.

Розроблена програма дозволяє забезпечити цілісність інформації, а саме введених даних за допомогою алгоритму хешування MD5. Програма може зчитувати дані із поля вводу або із файлу і виводити значення як на екран, так і у файл.