Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Інститут комп’ютерних наук та інформаційних технологій

Кафедра програмного забезпечення



**ЗВІТ**

**Про виконання лабораторної роботи № 2**

**з дисципліни «Безпека програм та даних»**

**«Створення програмного засобу для забезпечення цілісності інформації»**

**Лектор:**

доц. кафедри ПЗ

Сенів М. М.

**Виконав:**

студ. групи ПІ-41

Павленчик М. М.

**Прийняв:**

доц. кафедри ПЗ

Сенів М. М.

«\_\_\_» \_\_\_\_\_\_ 2018 р.

∑ = \_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_

Львів – 2018

**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.

### Теоретичні відомості

Хеш функція – це функція, що відображає вхідне слово скінченної довжини у скінченному алфавіті в слово заданої, зазвичай фіксованої довжини. 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.

Основні області використання хеш функцій – аутентифікація інформації та цифровий підпис. Практичне використання функцій хешування накладає на них ряд вимог, наведених нижче:

1. Хеш-функція H повинна застосовуватися до блоку даних будь-якої довжини.
2. Хеш-функція H створює вихід фіксованої довжини.
3. H(M) відносно легко (за поліноміальний час) обчислюється для будь-якого значення M, а алгоритм обчислення повинен бути практичним з погляду як апаратної, так і програмної реалізації.
4. Для будь-якого даного значення хеш-коду h обчислювально неможливо знайти M таке, що H(M)=h. Таку властивість іноді називають односторонністю.
5. Для будь-якого даного блоку x обчислювально неможливо знайти y≠x, для якого H(x)=H(y). Таку властивість іноді називають слабкою опірністю колізіям.
6. Обчислювально неможливо знайти довільну пару різних значень x та y, для яких H(x)=H(y). Таку властивість іноді називають сильною опірністю колізіям.

**Завдання для лабораторної роботи**

Створити програмну реалізацію алгоритму хешування MD5. 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. У звіті навести протокол тестування і роботи програми та зробити висновки.

### Результати

Протокол роботи програми

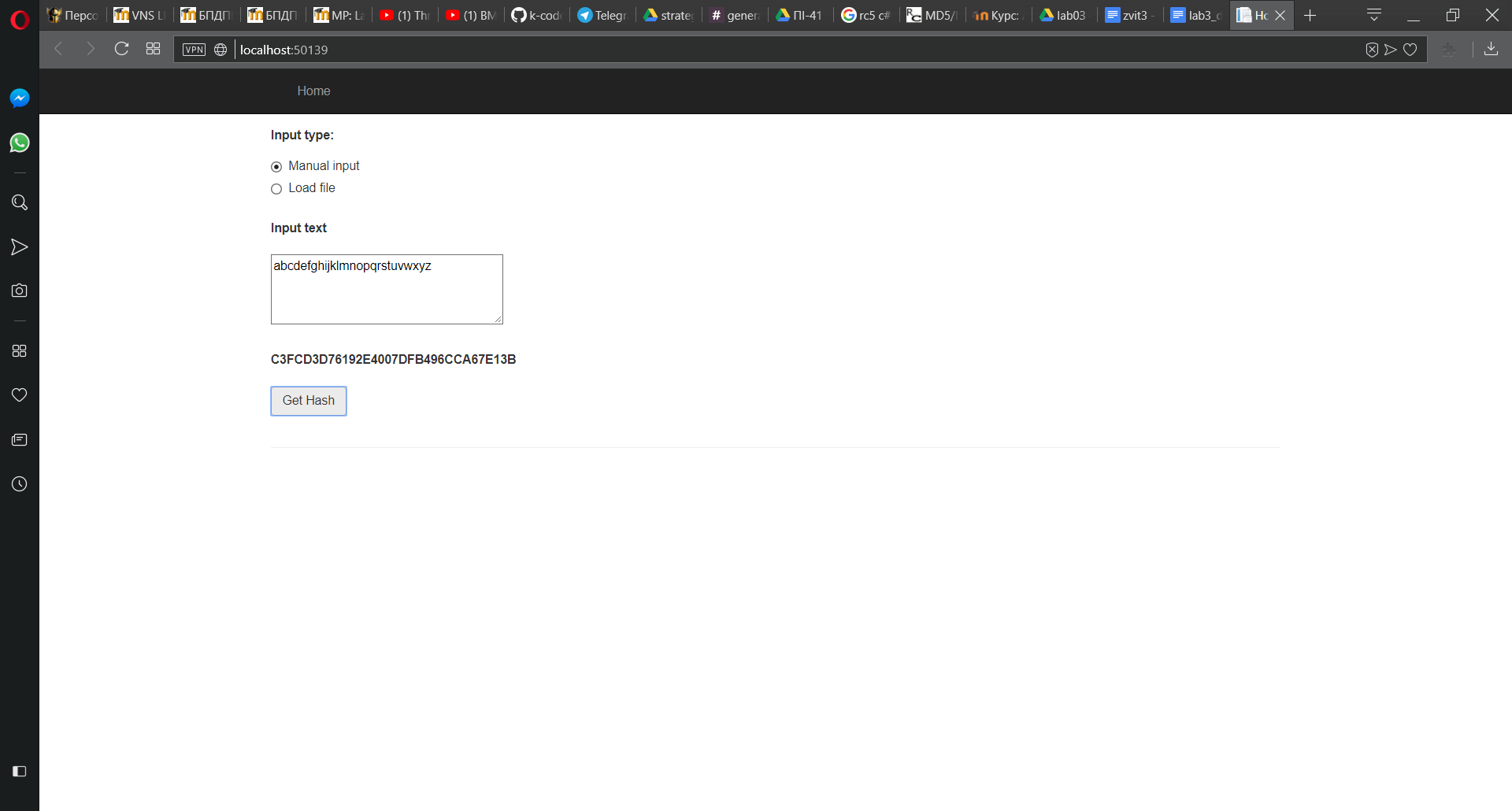


Рис. 1. Протокол роботи застосунку

Вихідний код програми

public class HashService

{

private const int CycleCount = 4;

private const int RoundStepsCount = 16;

public HashService()

{

MDBuffer = new MDBufferRepository();

RoundService = new RoundService();

}

private MDBufferRepository MDBuffer { get; set; }

private RoundService RoundService { get; set; }

public string GetHash(byte[] input)

{

input = GetFullInput(input);

return ProcessBlock(input, true);

}

public string ProcessBlock(byte[] block, bool isFirstBlock = false)

{

var roundModels = RoundService.GetRoundModels(block);

if (isFirstBlock)

MDBuffer.ResetBuffer();

foreach (var roundModel in roundModels)

{

var startupBuffer = MDBuffer.Clone();

for (int cycleNumber = 0; cycleNumber < CycleCount; cycleNumber++)

{

for (int i = 0; i < RoundStepsCount; i++)

{

var a = MDBuffer[0, i % 4];

var b = MDBuffer[1, i % 4];

var c = MDBuffer[2, i % 4];

var d = MDBuffer[3, i % 4];

var idx = RoundService.GetRoundIdx(i, cycleNumber);

RoundService.ApplyRoundFunction(

ref a, b, c, d,

roundModel[idx],

(i + 1) + 16 \* cycleNumber,

RoundService.S[cycleNumber, i % 4],

cycleNumber);

MDBuffer[i % 4] = a;

}

}

MDBuffer = MDBuffer + startupBuffer;

}

return MDBuffer.ToString();

}

public AppendModel GetAppendModel(long startupSize)

{

var inputSize = startupSize \* 8;

var inputSizeAppend = BitConverter.GetBytes(inputSize);

var appendSize = inputSize % 512 == 448 ? 64 :

inputSize % 512 > 448 ? (960 - inputSize % 512) / 8

: (448 - inputSize % 512) / 8;

var append = new byte[appendSize];

append[0] = 0x80;

return new AppendModel { InputSize = inputSizeAppend, Append = append };

}

private byte[] GetFullInput(byte[] startupInput)

{

var appends = GetAppendModel(startupInput.LongLength);

var result = new byte[startupInput.Length + appends.InputSize.Length + appends.Append.Length];

Array.Copy(startupInput, result, startupInput.Length);

Array.Copy(appends.Append, 0, result, startupInput.Length, appends.Append.Length);

Array.Copy(appends.InputSize, 0, result, startupInput.Length + appends.Append.Length, appends.InputSize.Length);

return result;

}

}

public class RoundService

{

public const int BlockSize = 64;

public List<RoundInputModel> GetRoundModels(byte[] input)

{

var roundsCount = input.Length / BlockSize;

var roundModels = new List<RoundInputModel>();

for (int i = 0; i < roundsCount; i++)

{

var roundInput = new byte[BlockSize];

Array.Copy(input, i \* BlockSize, roundInput, 0, BlockSize);

roundModels.Add(new RoundInputModel(roundInput));

}

return roundModels;

}

public void ApplyRoundFunction(ref uint A, uint B, uint C, uint D, uint X, int i, byte s, int cycleNumber)

{

var F = GetRoundFunction(cycleNumber);

A = B + GetLeftRotate(A + F(B, C, D) + X + GetT(i), s);

}

public byte[,] S { get; private set; } = new byte[,]

{

{ 7, 12, 17, 22 },

{ 5, 9, 14, 20 },

{ 4, 11, 16, 23 },

{ 6, 10, 15, 21 }

};

public int GetRoundIdx(int i, int cycleNumber)

{

switch (cycleNumber)

{

case 0: return i;

case 1: return (1 + 5 \* i) % 16;

case 2: return (5 + 3 \* i) % 16;

case 3: return (7 \* i) % 16;

default: throw new Exception("Unknown cycle number.");

}

}

private RoundFunction GetRoundFunction(int i)

{

switch (i)

{

case 0: return GetF;

case 1: return GetG;

case 2: return GetH;

case 3: return GetI;

default: throw new Exception("Unknown round function id.");

}

}

private uint GetT(int i)

{

return (uint)(Math.Pow(2, 32) \* Math.Abs(Math.Sin(i)));

}

private uint GetF(uint B, uint C, uint D)

{

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

}

private uint GetG(uint B, uint C, uint D)

{

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

}

private uint GetH(uint B, uint C, uint D)

{

return B ^ C ^ D;

}

private uint GetI(uint B, uint C, uint D)

{

return C ^ (B | ~D);

}

private uint GetLeftRotate(uint value, byte s)

{

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

}

public delegate uint RoundFunction(uint B, uint C, uint D);

}

**Висновки**

На даній лабораторній роботі я ознайомився з методами криптографічного забезпечення цілісності інформації та навчився створювати програмні засоби для забезпечення цілісності інформації з використанням алгоритмів хешування.