g***Министерство образования Республики Беларусь***

***Учреждение образования***

***«Брестский государственный технический университет»***

***Кафедра ИИТ***

**Лабораторная работа №1**

**По дисциплине КМЗИ за V семестр**

**Тема: «Криптографические хэш-функции»**

**Выполнил:**

Студент группы ИИ-15 (1)

3-го курса

Волк И. А.

**Проверил:**

Хацкевич М. В.

Брест 2019

**Цель:** Изучить существующие алгоритмы вычисления дайджестов сообщений и написать программу, реализующую заданный алгоритм хэширования.

**Ход работы**

Для начала я прочел материал, сопутствующий данной лабораторной работе. Вооруженный полученными знаниями, я приступил к реализации одного из алгоритмов, а именно RIPEMD-128.

**Результаты работы:**

|  |  |
| --- | --- |
| Сообщение | Хэш |
|  | 47dbf429 818ba3fe eb58b976 a0376301 |
| abc | 3a91f429 1c8ba3fe 6558b976 b83d9301 |
| Ilya Vouk | 6e7984a9 fa8bf97e 5895f976 2da6f301 |
| Ilya vouk | c27b84a9 fb0c017e a895f976 4da6f301 |

**Вывод:** изучили существующие алгоритмы вычисления дайджестов сообщений и написали программу, реализующую заданный алгоритм хэширования.

**Листинги программы:**

**RIPEMD128.h**

#include <vector>

#include <stdexcept>

#include <string>

class RIPEMD128

{

// Initial values

std::vector<unsigned int> h;

// Initial lenght

size\_t m\_initial\_length;

private:

// step 1

std::string add\_additional\_bits(const std::string &input);

// step 2

std::string add\_initial\_length(const std::string &input);

// step 4

void main\_cycle(const std::string &input);

// constants and functions

int f(int j, int x, int y, int z);

int K1(int j);

int K2(int j);

std::vector<int> R1(int j);

std::vector<int> R2(int j);

std::vector<int> S1(int j);

std::vector<int> S2(int j);

// initialize the output vector

void init\_h();

// logging

void log\_everything(const std::vector<int> &others) const;

public:

RIPEMD128();

// step 5. Result

std::string operator()(const std::string &input);

};

**RIPEMD128.cpp**

#include "RIPEMD128.h"

#include <sstream>

#include <iostream>

RIPEMD128::RIPEMD128()

{

init\_h();

}

std::string RIPEMD128::add\_additional\_bits(const std::string &input)

{

std::string result = input;

result += char(1 << 7);

for (int i = result.size(); i % 64 != 56; i++)

result += char(0);

return result;

}

std::string RIPEMD128::add\_initial\_length(const std::string &input)

{

std::string result = input;

for (int i = 0; i < 8; i++)

result += ((7 - i) \* 8 >> m\_initial\_length) & 0xFF;

return result;

}

void RIPEMD128::log\_everything(const std::vector<int> &others) const

{

std::clog << "H: ";

for (int i = 0; i < 4; i++)

{

if (i != 0)

std::clog << ' ';

std::clog << std::hex << h[i];

}

std::clog << "\nInitial length: " << std::dec << m\_initial\_length << std::endl;

std::clog << "\nOthers: " << std::endl;

for (int i = 0; i < others.size(); i++)

{

std::clog << i << ": " << std::hex << others[i] << std::endl;

}

}

void RIPEMD128::main\_cycle(const std::string &M)

{

int T,

A1, B1, C1, D1,

A2, B2, C2, D2;

for (int i = 0; i < M.size() / 512 + 1; i++)

{

A1 = h[0];

B1 = h[1];

C1 = h[2];

D1 = h[3];

A2 = h[0];

B2 = h[1];

C2 = h[2];

D2 = h[3];

for (int j = 0; j < 64; j++)

{

T = (A1 + f(j, B1, C1, D1) + M[i \* 512 + R1(j)[j % 16]] + K1(j)) << S1(j)[j % 16];

A1 = D1;

D1 = C1;

C1 = B1;

B1 = T;

T = (A2 + f(63 - j, B2, C2, D2) + M[i \* 512 + R2(j)[j % 16]] + K2(j)) << S2(j)[j % 16];

A2 = D2;

D2 = C2;

C2 = B2;

B2 = T;

}

T = h[1] + C1 + D2;

h[1] = h[2] + D1 + A2;

h[2] = h[3] + A1 + B2;

h[3] = h[0] + B1 + C2;

h[0] = T;

}

}

std::string RIPEMD128::operator()(const std::string &input)

{

init\_h();

m\_initial\_length = input.size() \* 8;

std::string result = add\_additional\_bits(input);

result = add\_initial\_length(result);

main\_cycle(result);

std::stringstream result\_stream;

for (int i = 0; i < 4; i++)

{

if (i != 0)

result\_stream << ' ';

result\_stream << std::hex << h[i];

}

return result\_stream.str();

}

/////////////////////////////

// Constants and functions //

/////////////////////////////

void RIPEMD128::init\_h()

{

h = {

0x67452301, 0xEFCDAB89, 0x98BADCFE, 0x10325476,

0xC3D2E1F0, 0x76543210, 0xFEDCBA98, 0x89ABCDEF,

0x01234567, 0x3C2D1E0F };

}

int RIPEMD128::f(int j, int x, int y, int z)

{

if (0 <= j && j <= 15)

return x ^ y ^ z;

else if (16 <= j && j <= 31)

return (x & y) | (~x & z);

else if (32 <= j && j <= 47)

return (x | ~y) ^ z;

else if (48 <= j && j <= 63)

return (x & z) | (y & ~z);

else if (64 <= j && j <= 79)

return x ^ (y | ~z);

else

throw std::out\_of\_range("j is out of range");

}

int RIPEMD128::K1(int j)

{

if (0 <= j && j <= 15)

return 0x00000000;

else if (16 <= j && j <= 31)

return 0x5A827999;

else if (32 <= j && j <= 47)

return 0x6ED9EBA1;

else if (48 <= j && j <= 63)

return 0x8F1BBCDC;

else if (64 <= j && j <= 79)

return 0xA953FD4E;

else

throw std::out\_of\_range("j is out of range");

}

int RIPEMD128::K2(int j)

{

if (0 <= j && j <= 15)

return 0x50A28BE6;

else if (16 <= j && j <= 31)

return 0x5C4DD124;

else if (32 <= j && j <= 47)

return 0x6D703EF3;

else if (48 <= j && j <= 63)

return 0x7A6D76E9;

else if (64 <= j && j <= 79)

return 0x00000000;

else

throw std::out\_of\_range("j is out of range");

}

std::vector<int> RIPEMD128::R1(int j)

{

if (0 <= j && j <= 15)

return {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15};

else if (16 <= j && j <= 31)

return {7, 4, 13, 1, 10, 6, 15, 3, 12, 0, 9, 5, 2, 14, 11, 8};

else if (32 <= j && j <= 47)

return {3, 10, 14, 4, 9, 15, 8, 1, 2, 7, 0, 6, 13, 11, 5, 12};

else if (48 <= j && j <= 63)

return {1, 9, 11, 10, 0, 8, 12, 4, 13, 3, 7, 15, 14, 5, 6, 2};

else if (64 <= j && j <= 79)

return {4, 0, 5, 9, 7, 12, 2, 10, 14, 1, 3, 8, 11, 6, 15, 13};

else

throw std::out\_of\_range("j is out of range");

}

std::vector<int> RIPEMD128::R2(int j)

{

if (0 <= j && j <= 15)

return {5, 14, 7, 0, 9, 2, 11, 4, 13, 6, 15, 8, 1, 10, 3, 12};

else if (16 <= j && j <= 31)

return {6, 11, 3, 7, 0, 13, 5, 10, 14, 15, 8, 12, 4, 9, 1, 2};

else if (32 <= j && j <= 47)

return {15, 5, 1, 3, 7, 14, 6, 9, 11, 8, 12, 2, 10, 0, 4, 13};

else if (48 <= j && j <= 63)

return {8, 6, 4, 1, 3, 11, 15, 0, 5, 12, 2, 13, 9, 7, 10, 14};

else if (64 <= j && j <= 79)

return {12, 15, 10, 4, 1, 5, 8, 7, 6, 2, 13, 14, 0, 3, 9, 11};

else

throw std::out\_of\_range("j is out of range");

}

std::vector<int> RIPEMD128::S1(int j)

{

if (0 <= j && j <= 15)

return {11, 14, 15, 12, 5, 8, 7, 9, 11, 13, 14, 15, 6, 7, 9, 8};

else if (16 <= j && j <= 31)

return {7, 6, 8, 13, 11, 9, 7, 15, 7, 12, 15, 9, 11, 7, 13, 12};

else if (32 <= j && j <= 47)

return {11, 13, 6, 7, 14, 9, 13, 15, 14, 8, 13, 6, 5, 12, 7, 5};

else if (48 <= j && j <= 63)

return {11, 12, 14, 15, 14, 15, 9, 8, 9, 14, 5, 6, 8, 6, 5, 12};

else if (64 <= j && j <= 79)

return {9, 15, 5, 11, 6, 8, 13, 12, 5, 12, 13, 14, 11, 8, 5, 6};

else

throw std::out\_of\_range("j is out of range");

}

std::vector<int> RIPEMD128::S2(int j)

{

if (0 <= j && j <= 15)

return {8, 9, 9, 11, 13, 15, 15, 5, 7, 7, 8, 11, 14, 14, 12, 6};

else if (16 <= j && j <= 31)

return {9, 13, 15, 7, 12, 8, 9, 11, 7, 7, 12, 7, 6, 15, 13, 11};

else if (32 <= j && j <= 47)

return {9, 7, 15, 11, 8, 6, 6, 14, 12, 13, 5, 14, 13, 13, 7, 5};

else if (48 <= j && j <= 63)

return {15, 5, 8, 11, 14, 14, 6, 14, 6, 9, 12, 9, 12, 5, 15, 8};

else if (64 <= j && j <= 79)

return {8, 5, 12, 9, 12, 5, 14, 6, 8, 13, 6, 5, 15, 13, 11, 11};

else

throw std::out\_of\_range("j is out of range");

}