|  |  |
| --- | --- |
| Министерство образования Республики Беларусь  Учреждение образования «Полоцкий государственный университет» | |
|  | Факультет информационных технологий  Кафедра технологий программирования |
| Лабораторная работа №2 по курсу «Теория информации»  «Электронная цифровая подпись» | |
| Выполнил | Студент гр. 21-ИТ-1  Катушёнок И.В. |
| Проверила | Васильева Д.М. |
| Полоцк, 2023г. | |

**Ход работы**

Алгоритм безопасного хеширования SHA-1 был опубликован в 1995 году в качестве замены использовавшегося до этого алгоритма хеширования SHA-0, в котором была обнаружена уязвимость.

**Задание:**

1. Реализовать алгоритм вычисления хеш-функции SHA-1 для файла с произвольным размером и содержимым.

2. Реализовать программное средство, выполняющее генерацию и проверку ЭЦП файла с произвольным содержимым на базе алгоритма RSA с использованием для вычисления хеш-функции ранее реализованного алгоритма SHA-1.

Листинг 1 - реализация программы:

public partial class PSA

{

char[] characters = new char[] { '#', 'А', 'Б', 'В', 'Г', 'Д', 'Е', 'Ё', 'Ж', 'З', 'И',

'Й', 'К', 'Л', 'М', 'Н', 'О', 'П', 'Р', 'С',

'Т', 'У', 'Ф', 'Х', 'Ц', 'Ч', 'Ш', 'Щ', 'Ь', 'Ы', 'Ъ',

'Э', 'Ю', 'Я', ' ', '1', '2', '3', '4', '5', '6', '7',

'8', '9', '0' };

//зашифровать

public void Start(long p, long q, ref long d, ref long n)

{

string s = "";

StreamReader sr = new StreamReader("in.txt");

while (!sr.EndOfStream)

{

s += sr.ReadLine();

}

sr.Close();

s = s.ToUpper();

long n\_ = p \* q;

long m = (p - 1) \* (q - 1);

long d\_ = Calculate\_d(m);

long e\_ = 10199;//Calculate\_e(d, m);

List<string> result = RSA\_Endoce(s, e\_, n\_);

StreamWriter sw = new StreamWriter("out1.txt");

foreach (string item in result)

sw.WriteLine(item);

sw.Close();

d = d\_;

n = n\_;

}

//расшифровать

public void startD(long d, long n)

{

List<string> input = new List<string>();

StreamReader sr = new StreamReader("out1.txt");

while (!sr.EndOfStream)

{

input.Add(sr.ReadLine());

}

sr.Close();

string result = RSA\_Dedoce(input, d, n);

StreamWriter sw = new StreamWriter("out2.txt");

sw.WriteLine(result);

sw.Close();

}

//зашифровать

private List<string> RSA\_Endoce(string s, long e, long n)

{

List<string> result = new List<string>();

BigInteger bi;

for (int i = 0; i < s.Length; i++)

{

int index = Array.IndexOf(characters, s[i]);

bi = new BigInteger(index);

bi = BigInteger.Pow(bi, (int)e);

BigInteger n\_ = new BigInteger((int)n);

bi = bi % n\_;

result.Add(bi.ToString());

}

return result;

}

//расшифровать

private string RSA\_Dedoce(List<string> input, long d, long n)

{

string result = "";

BigInteger bi;

foreach (string item in input)

{

bi = new BigInteger(Convert.ToDouble(item));

bi = BigInteger.Pow(bi, (int)d);

BigInteger n\_ = new BigInteger((int)n);

bi = bi % n\_;

int index = Convert.ToInt32(bi.ToString());

result += characters[index].ToString();

}

return result;

}

//вычисление параметра d. d должно быть взаимно простым с m

private long Calculate\_d(long m)

{

long d = m - 1;

for (long i = 2; i <= m; i++)

if ((m % i == 0) && (d % i == 0)) //если имеют общие делители

{

d--;

i = 1;

}

return d;

}

//вычисление параметра e

private long Calculate\_e(long d, long m)

{

long e = 10;

while (true)

{

if ((e \* d) % m == 1)

break;

else

e++;

}

return e;

}

}

class Program

{

static string ByteArrayToString(byte[] \_by)

{

string result = "";

for (int c = 0; c < \_by.Length; c++)

{

result += \_by[c].ToString("X2");

}

return result;

}

static void Main(string[] args)

{

byte[] b = { 5, 6 };

HashAlgorithm sha = SHA1.Create();

byte[] result = sha.ComputeHash(File.ReadAllBytes(@"X:\vs\git\lab\sem4\TI\2\t.txt"));

string results;

Console.WriteLine(results = ByteArrayToString(result));

//Console.WriteLine("task 2");

Console.ReadKey();

PSA psa = new PSA();

long p = 101, q = 103, d = 0, n = 0;

psa.Start(p, q, ref d, ref n);

psa.startD(d, n);

}

}

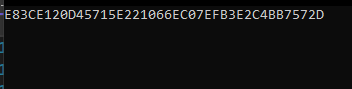


Рисунок 1. – Хэш-код

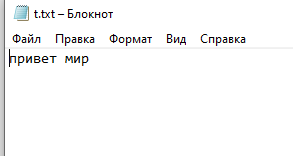


Рисунок 2. – исходный файл хэш-кода

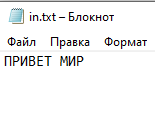


Рисунок 3. – исходный текст

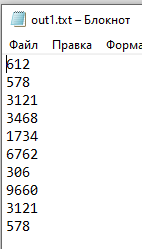


Рисунок 4. – зашифрованный текст