МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования  
 «Ижевский государственный технический университет имени М.Т.Калашникова»

Факультет «Информатика и вычислительная техника»

Кафедра «Вычислительная техника»

Отчет

по лабораторной работе №4

на тему «Хэширование информации»

по дисциплине «Методы и способы защиты компьютерной информации»

Выполнил:

Проверил:

студент группы Б08-781-1

Мушегов В.А.

к.т.н., доцент

Марков Е.М.

Ижевск 2020

**Цель работы**

Изучение основных принципов хэширования информации, приобретение навыков программной реализации алгоритмов ЭЦП и защиты данных с ее помощью.

**Задание**

Составить систему ЭЦП с использованием асимметричного шифра и хэшфункции SHA-1.

Размер ключей при формировании и проверке подписи – не менее 128 бит. При проверке подписи сохраненная ЭЦП должна сравниваться с актуальной для текущего состояния файла данных.

**Основные сведения**

Многие современные технологии безопасности (например, аутентификации,

ЭЦП) применяют односторонние функции шифрования, называемые также хэшфункциями. Основное назначение подобных функций – получение из сообщения

произвольного размера его дайджеста – значения фиксированного размера. Дайджест может быть использован в качестве контрольной суммы исходного сообщения, обеспечивая таким образом (при использовании соответствующего протокола)

контроль целостности информации . Основные свойства хэш-функции:

1) на вход хэш-функции подается сообщение произвольной длины;

2) на выходе хэш-функции формируется блок данных фиксированной длины;

3) значения на выходе хэш-функции распределены по равномерному закону;

4) при изменении одного бита на входе хэш-функции существенно изменяется

выход.

Кроме того, для обеспечения устойчивости хэш-функции к атакам она должна

удовлетворять следующим требованиям:

1) если мы знаем значение хэш-функции h, то задача нахождения сообщения

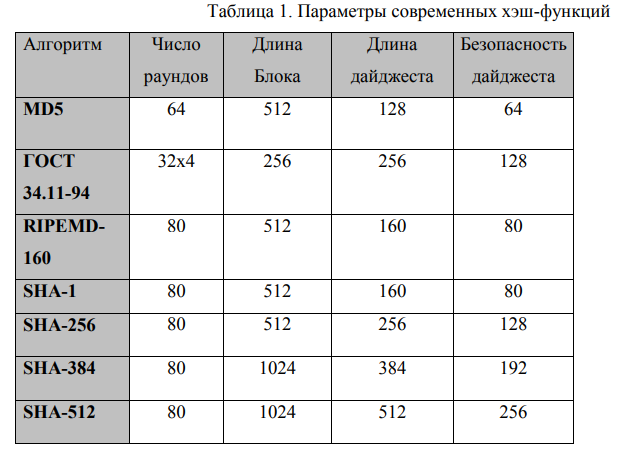
M такого, что Н(М)=h, должна быть вычислительно трудной;

2) при заданном сообщении M задача нахождения другого сообщения M, такого, что Н(М)=H(M’), должна быть вычислительно трудной

Если хэш-функция будет удовлетворять перечисленным свойствам, то формируемое ею значение будет уникально идентифицировать сообщения, и всякая попытка изменения сообщения при передаче будет обнаружена путем выполнения хэширования на принимающей стороне и сравнением с дайджестом, полученным на

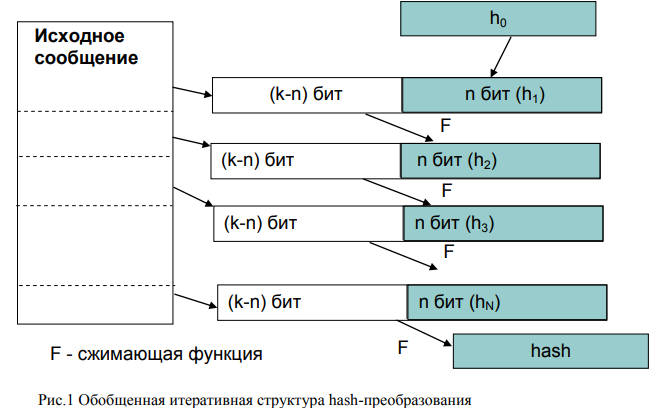
передающей стороне.

Еще одной особенностью хэш-функций является то, что они не допускают обратного преобразования – получить исходное сообщения по его дайджесту невозможно. Поэтому их называют еще односторонними функциями шифрования. В таблице 1 дан cравнительный анализ наиболее популярных хэш-функций.



Под безопасностью здесь понимается стойкость к атакам типа "парадокс дня

рождения".Хэш-функции строятся по итеративной схеме, когда исходное сообщение разбивается на блоки определенного размера, и над ними выполняются ряд преобразований с использованием как обратимых, так и необратимых операций. Как правило, в состав хэширующего преобразования включается сжимающая функция, поскольку частую по размеру меньше блока, подаваемого на вход. На вход каждого цикла хэширования подается выход предыдущего цикла, а также очередной блок сообщения. Таким образом, на каждом цикле выход хэш-функции hi представляет собой хэш первых i блоков (рис.1).

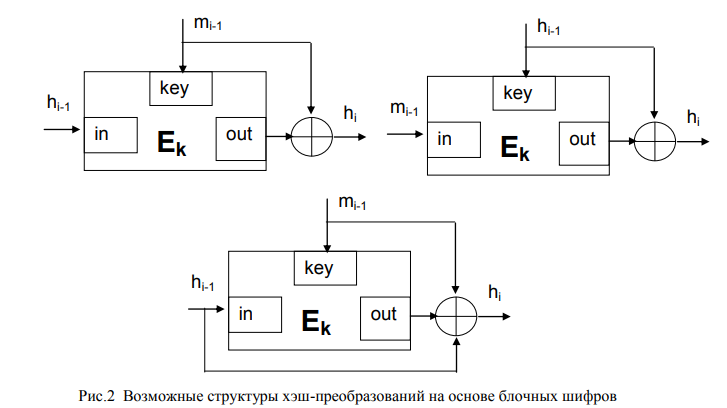


Из схемы очевидно, что криптографические свойства хэш-функции во многом

зависят от выбора сжимающей функции. В качестве одного из вариантов подобного

выбора может стать использование блочного шифра, так как он хорошо рандомизирует входящее сообщение и благодаря лавинному эффекту выход блочного шифра

зависит от каждого бита входного сообщения. Варианты формирования сжимающей

функции из блочного шифра приведены на рис.2. 

Алгоритм хэширования SHA (Secure Hash Algorithm) описан в стандарте SHS и обеспечивает безопасность электронной цифровой подписи DSA.

**Листинг программы**

#define CRYPTOPP\_ENABLE\_NAMESPACE\_WEAK 1

#include <iostream>

#include <iomanip>

#include <fstream>

#include <string>

#include "C:\Libraries\cryptopp820\sha.h"

#include "C:\Libraries\cryptopp820\hex.h"

using namespace std;

using namespace CryptoPP;

string readFile(const std::string& fileName) {

ifstream f(fileName, ios::binary);

stringstream ss;

ss << f.rdbuf();

return ss.str();

}

std::string Encrypt(std::string message) {

setlocale(LC\_ALL, "RUS");

byte digest[CryptoPP::SHA1::DIGESTSIZE];

SHA1 hash;

hash.CalculateDigest(digest, (const byte\*)message.c\_str(), message.length());

HexEncoder encoder;

string output;

encoder.Attach(new StringSink(output));

encoder.Put(digest, sizeof(digest));

encoder.MessageEnd();

return output;

}

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

setlocale(LC\_ALL, "RUS");

string file = "text.txt";

string message = readFile(file);

string crypt;

crypt = Encrypt(message);

ofstream fout;

fout.open("hash\_" + file, ios::trunc | ios::binary);

fout << crypt;

fout.close();

return 0;

}

**Результаты работы**

