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

Суровцева А.С.

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 MD5.

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

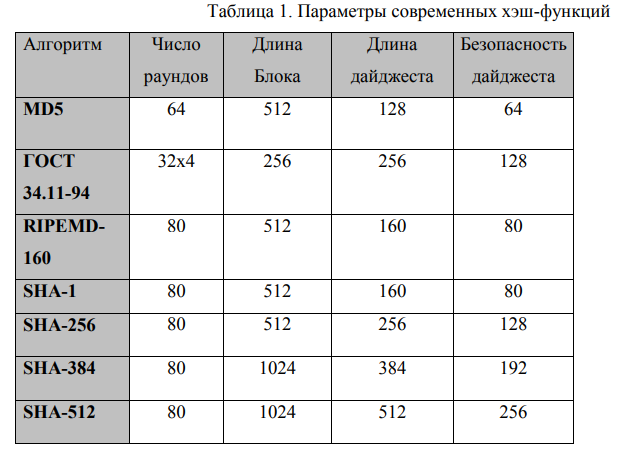
M такого, что Н(М)=h, должна быть вычислительно трудной;

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

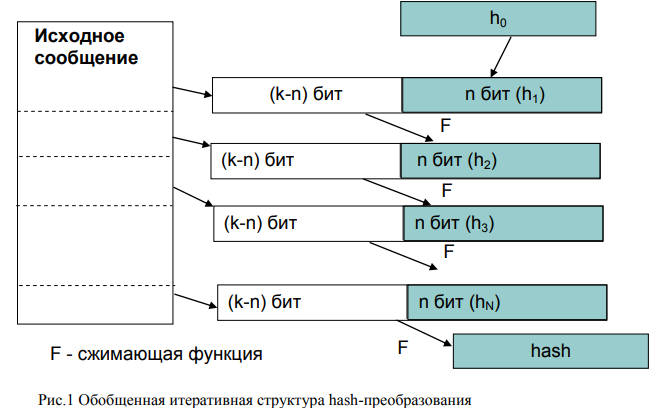
передающей стороне.

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).

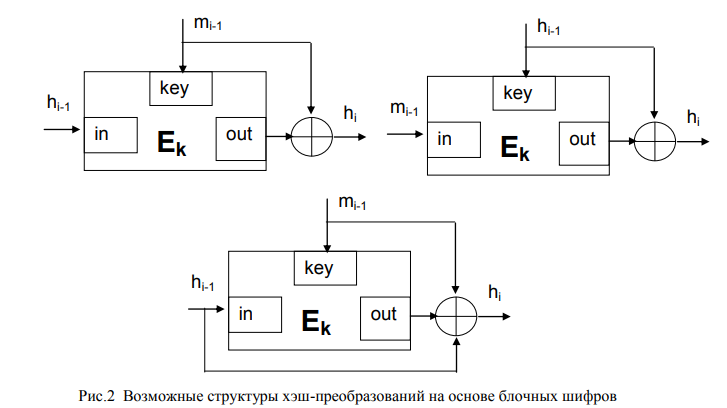


Из схемы очевидно, что криптографические свойства хэш-функции во многом

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 MD5 (Message Digest), разработанный Роном Ривестом в 1991 г.,

формирует 128-битный хэш. Данный алгоритм используется в таких современных

протоколах защиты информации, как SSL и IP Sec. Современные исследования продемонстрировали нестойкость данного алгоритма к обнаружению коллизий, поэтому его использование для целей практической защиты данных может быть небезопасным.

**Листинг программы**

#define CRYPTOPP\_ENABLE\_NAMESPACE\_WEAK 1

#include <iostream>

#include <iomanip>

#include <fstream>

#include <string>

#include "C:\Libraries\cryptopp820\md5.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::Weak::MD5::DIGESTSIZE];

Weak::MD5 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 k, name\_file;

while (true) {

do {

system("cls");

cout << "\n Доступные операции: ";

cout << "\n1. Получить хэш";

cout << "\n0. Выйти";

cout << "\n Ваш выбор: ";

cin >> k;

} while (k.length() > 1 || k != "1" && k != "0");

if (k == "1") {

system("cls");

cout << "\n Введите название файла: ";

cin >> name\_file;

string message = readFile(name\_file);

string crypt;

crypt = Encrypt(message);

ofstream fout;

fout.open("hash.txt", ios::trunc | ios::binary);

fout << crypt;

fout.close();

system("cls");

cout << "\nХэш записан в файл. \n";

system("pause");

}

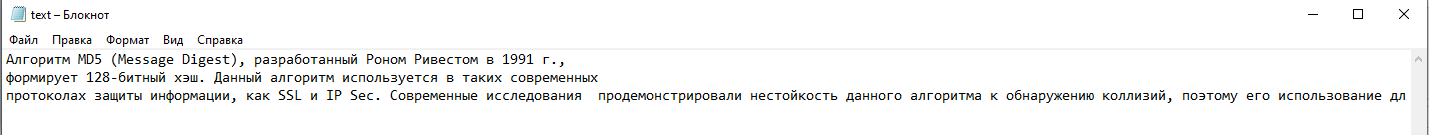
if (k == "0") return 0;

}

return 0;

}

**Результаты работы**



**Ответы на вопросы**

1)Для чего был разработан, где применяется.

MD5 (англ. Message Digest 5) — 128-битный алгоритм хеширования, разработанный профессором Рональдом Л. Ривестом из Массачусетского технологического института (Massachusetts Institute of Technology, MIT) в 1991 году. Предназначен для создания «отпечатков» или дайджестов сообщения произвольной длины и последующей проверки их подлинности. Широко применялся для проверки целостности информации и хранения хешей паролей.

MD5 — один из серии алгоритмов по построению дайджеста сообщения, разработанный профессором Рональдом Л. Ривестом из Массачусетского технологического института. Был разработан в 1991 году как более надёжный вариант предыдущего алгоритма MD4. Описан в RFC 1321. Позже Гансом Доббертином были найдены недостатки алгоритма MD4.

В 1993 году Берт ден Бур и Антон Босселарс показали, что в алгоритме возможны псевдоколлизии, когда разным инициализирующим векторам соответствуют одинаковые дайджесты для входного сообщения.

В 1996 году Ганс Доббертин объявил о коллизии в алгоритме, и уже в то время было предложено использовать другие алгоритмы хеширования, такие как Whirlpool, SHA-1 или RIPEMD-160.

Из-за небольшого размера хеша в 128 бит можно рассматривать birthday-атаки. В марте 2004 года был запущен проект MD5CRK с целью обнаружения уязвимостей алгоритма, используя birthday-атаки. Проект MD5CRK закончился 17 августа 2004 года, когда Ван Сяоюнь, Фэн Дэнго, Лай Сюэцзя и Юй Хунбо обнаружили уязвимости в алгоритме.

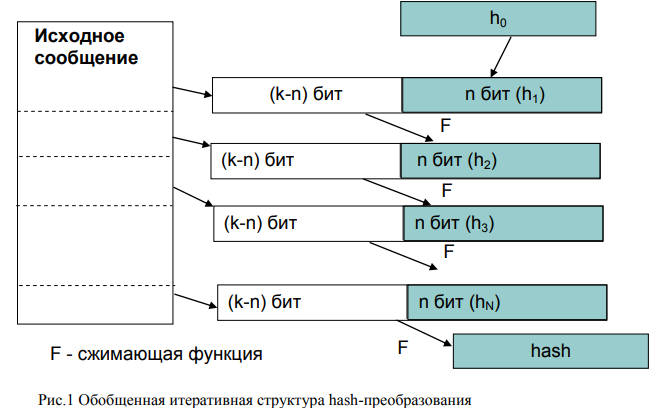
2)Принцип работы и характеристики

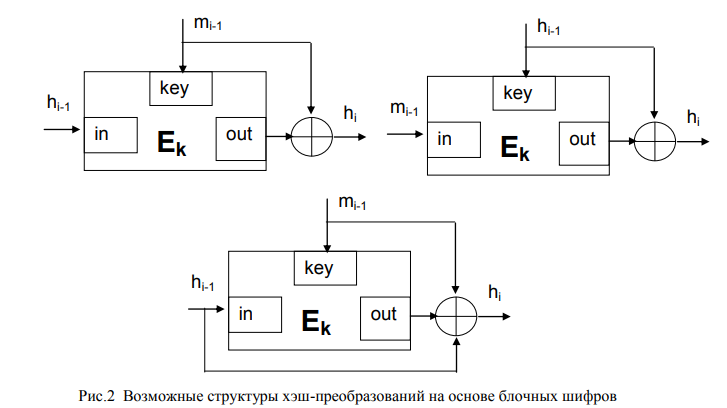
Алгоритм состоит из пяти шагов:  
1)Append Padding Bits  
В исходную строку дописывают единичный байт 0х80, а затем дописывают нулевые биты, до тех пор, пока длина сообщения не будет сравнима с 448 по модулю 512. То есть дописываем нули до тех пор, пока длина нового сообщения не будет равна [длина] = (512\*N+448),  
где N — любое натуральное число, такое, что это выражение будет наиболее близко к длине блока.  
2)Append Length  
Далее в сообщение дописывается 64-битное представление длины исходного сообщения.  
3)Initialize MD Buffer  
На этом шаге инициализируется буффер  
word A: 01 23 45 67  
word B: 89 ab cd ef  
word C: fe dc ba 98  
word D: 76 54 32 10  
Как можно заметить буффер состоит из четырех констант, предназначенный для сбора хэша.  
4)Process Message in 16-Word Blocks  
На четвертом шаге в первую очередь определяется 4 вспомогательные логические функции, которые преобразуют входные 32-битные слова, в 32-битные выходные.  
F(X,Y,Z) = XY v not(X) Z  
G(X,Y,Z) = XZ v Y not(Z)  
H(X,Y,Z) = X xor Y xor Z  
I(X,Y,Z) = Y xor (X v not(Z))  
Также на этом шаге реализуется так называемый «белый шум» — усиление алгоритма, состоящее 64 элементного массива, содержащего псевдослучайные числа, зависимые от синуса числа i:  
T[i]=4,294,967,296\*abs(sin(i))  
Далее копируем каждый 16-битный блок в массив X[16] и производим манипуляции:  
AA = A  
BB = B  
CC = C  
DD = D  
Затем происходят преобразования-раунды, которых всего будет 4. Каждый раунд состоит из 16 элементарных преобразований, которые в общем виде можно представить в виде [abcd k s i], которое, в свою очередь, можно представить как A = B + ((A + F(B,C,D) + X[k] + T[i]) <<< s), где  
A, B, C, D — регистры  
F(B,C,D) — одна из логических функций  
X[k] — k-тый элемент 16-битного блока.  
T[i] — i-тый элемент таблицы «белого шума»  
<<< s — операция циклического сдвига на s позиций влево.  
В конце суммируем результаты вычислений:  
A = A + AA  
B = B + BB  
C = C + CC  
D = D + DD  
5) Output  
Выводя побайтово буффер ABCD начиная с A и заканчивая D получим хэш.

3)Криптостойкость для алгоритма

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 MD5 (Message Digest), разработанный Роном Ривестом в 1991 г.,

формирует 128-битный хэш. Данный алгоритм используется в таких современных протоколах защиты информации, как SSL и IP Sec. Современные исследования продемонстрировали нестойкость данного алгоритма к обнаружению коллизий, поэтому его использование для целей практической защиты данных может быть небезопасным.

4)Криптоанализ

В 1993 году Берт ден Бур и Антон Босселарс показали, что в алгоритме возможны псевдоколлизии, когда разным инициализирующим векторам соответствуют одинаковые дайджесты для входного сообщения.

В 1996 году Ганс Доббертин объявил о коллизии в алгоритме, и уже в то время было предложено использовать другие алгоритмы хеширования, такие как Whirlpool, SHA-1 или RIPEMD-160.

Из-за небольшого размера хеша в 128 бит можно рассматривать birthday-атаки. В марте 2004 года был запущен проект MD5CRK с целью обнаружения уязвимостей алгоритма, используя birthday-атаки. Проект MD5CRK закончился 17 августа 2004 года, когда Ван Сяоюнь, Фэн Дэнго, Лай Сюэцзя и Юй Хунбо обнаружили уязвимости в алгоритме.

На данный момент существуют несколько видов «взлома» хешей MD5 — подбора сообщения с заданным хешем:

- Перебор по словарю

- Brute-force

- RainbowCrack

- Коллизия хеш-функции

При этом методы перебора по словарю и brute-force могут использоваться для взлома хеша других хеш-функций (с небольшими изменениями алгоритма). В отличие от них, RainbowCrack требует предварительной подготовки радужных таблиц, которые создаются для заранее определённой хеш-функции. Поиск коллизий специфичен для каждого алгоритма.