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

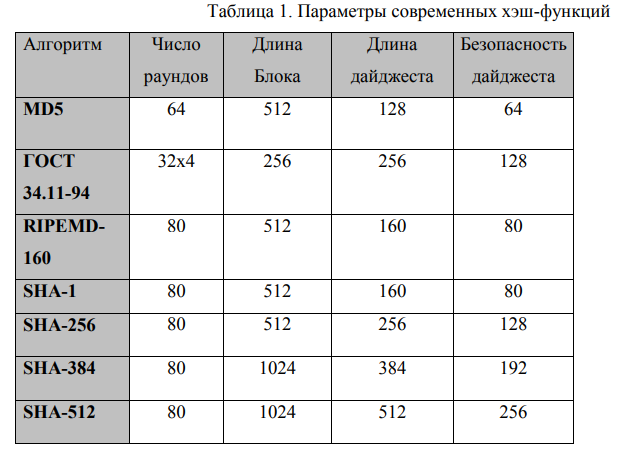
M такого, что Н(М)=h, должна быть вычислительно трудной;

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

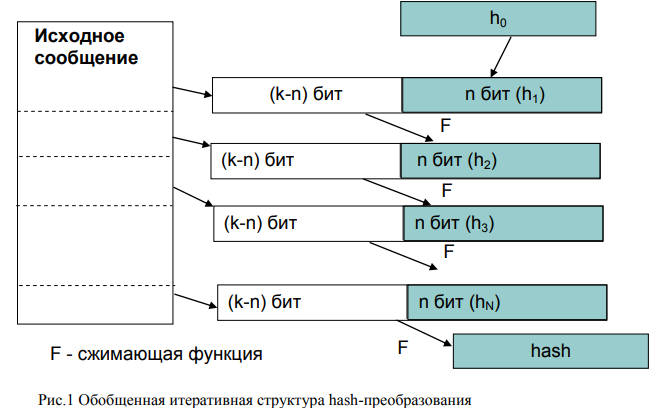
передающей стороне.

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).

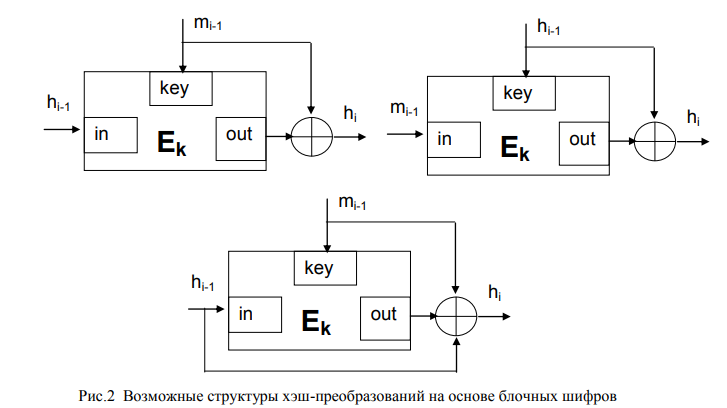


Из схемы очевидно, что криптографические свойства хэш-функции во многом

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 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");

}

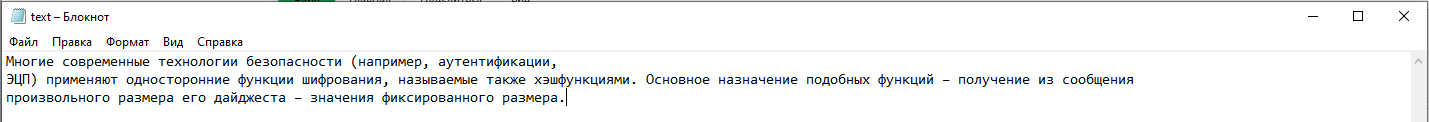
if (k == "0") return 0;

}

return 0;

}

**Результаты работы**



**Ответы на вопросы**

1)Для чего был разработан, где применяется.

Secure Hash Algorithm 1 — алгоритм криптографического хеширования. Описан в RFC 3174. Для входного сообщения произвольной длины (максимум примерно 2 эксабайта) алгоритм генерирует 160-битное (20 байт) хеш-значение, называемое также дайджестом сообщения, которое обычно отображается как шестнадцатиричное число, длиной в 40 цифр. Используется во многих криптографических приложениях и протоколах. Также рекомендован в качестве основного для государственных учреждений в США. Принципы, положенные в основу SHA-1, аналогичны тем, которые использовались Рональдом Ривестом при проектировании MD4.

В 1993 году NSA совместно с NIST разработали алгоритм безопасного хеширования (сейчас известный как SHA-0) (опубликован в документе FIPS PUB 180) для стандарта безопасного хеширования. Однако вскоре NSA отозвало данную версию, сославшись на обнаруженную ими ошибку, которая так и не была раскрыта. И заменило его исправленной версией, опубликованной в 1995 году в документе FIPS PUB 180-1. Эта версия и считается тем, что называют SHA-1. Позже, на конференции CRYPTO в 1998 году два французских исследователя представили атаку на алгоритм SHA-0, которая не работала на алгоритме SHA-1 Возможно, это и была ошибка, открытая NSA.

2)Принцип работы и характеристики

SHA-1 реализует хеш-функцию, построенную на идее функции сжатия. Входами функции сжатия являются блок сообщения длиной 512 бит и выход предыдущего блока сообщения. Выход представляет собой значение всех хеш-блоков до этого момента. Иными словами хеш-блок Mi равен hi = f(Mi, hi-1). Хеш-значением всего сообщения является выход последнего блока.

Исходное сообщение разбивается на блоки по 512 бит в каждом. Последний блок дополняется до длины, кратной 512 бит. Сначала добавляется 1 (бит), а потом нули, чтобы длина блока стала равной (512 — 64 = 448) бит. В оставшиеся 64 бита записывается длина исходного сообщения в битах (в big-endian формате). Если последний блок имеет длину более 448, но менее 512 бит, то дополнение выполняется следующим образом: сначала добавляется 1 (бит), затем нули вплоть до конца 512-битного блока; после этого создается ещё один 512-битный блок, который заполняется вплоть до 448 бит нулями, после чего в оставшиеся 64 бита записывается длина исходного сообщения в битах (в big-endian формате). Дополнение последнего блока осуществляется всегда, даже если сообщение уже имеет нужную длину.

Инициализируются пять 32-битовых переменных.

A = a = 0x67452301

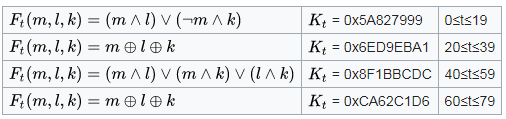
B = b = 0xEFCDAB89

C = c = 0x98BADCFE

D = d = 0x10325476

E = e = 0xC3D2E1F0

Определяются четыре нелинейные операции и четыре константы.



Главный цикл

Главный цикл итеративно обрабатывает каждый 512-битный блок. Итерация состоит из четырёх этапов по двадцать операций в каждом. Блок сообщения преобразуется из 16 32-битовых слов Mi в 80 32-битовых слов Wj.

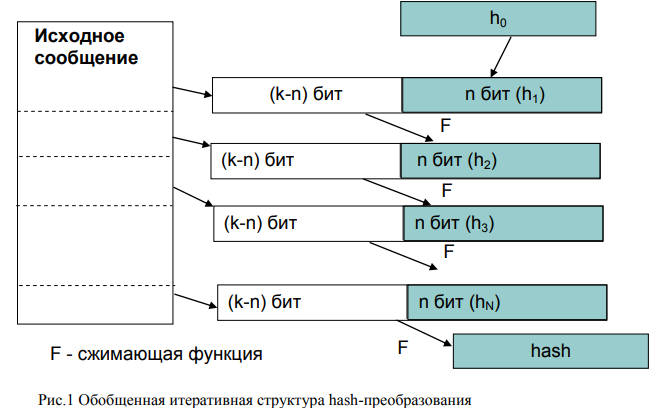
После этого a, b, c, d, e прибавляются к A, B, C, D, E, соответственно. Начинается следующая итерация.

Итоговым значением будет объединение пяти 32-битовых слов в одно 160-битное хеш-значение.

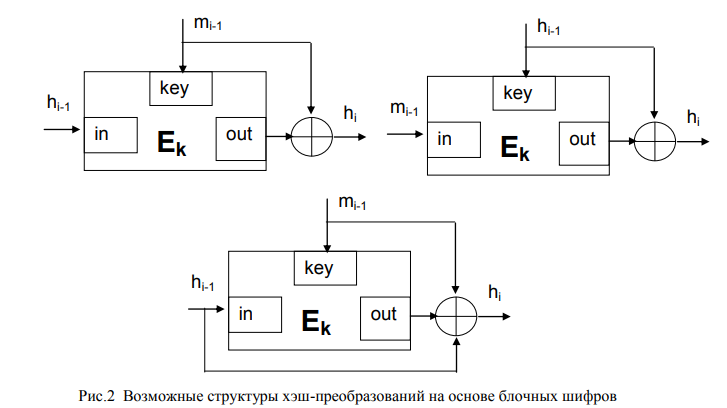
3)Криптостойкость для алгоритма

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

4)Криптоанализ

Криптоанализ хеш-функций направлен на исследование уязвимости для различного вида атак. Основные из них:

* нахождение коллизий — ситуация, когда двум различным исходным сообщениям соответствует одно и то же хеш-значение.
* нахождение прообраза — исходного сообщения — по его хешу.

При решении методом «грубой силы»:

* первая задача требует в среднем 2160/2 = 280 операций, если использовать атаку Дней рождения.
* вторая требует 2160 операций.

От устойчивости хеш-функции к нахождению коллизий зависит безопасность электронной цифровой подписи с использованием данного хеш-алгоритма. От устойчивости к нахождению прообраза зависит безопасность хранения хешей паролей для целей аутентификации.

В январе [2005 года](https://ru.wikipedia.org/wiki/2005_%D0%B3%D0%BE%D0%B4) Винсент Рэймен и Elisabeth Oswald опубликовали сообщение об атаке на усечённую версию SHA-1 (53 раунда вместо 80), которая позволяет находить коллизии меньше, чем за 280 операций.

В феврале 2005 года [Сяоюнь Ван](https://ru.wikipedia.org/w/index.php?title=%D0%A1%D1%8F%D0%BE%D1%8E%D0%BD%D1%8C_%D0%92%D0%B0%D0%BD&action=edit&redlink=1" \o "Сяоюнь Ван (страница отсутствует)), [Ицюнь Лиза Инь](https://ru.wikipedia.org/w/index.php?title=%D0%98%D1%86%D1%8E%D0%BD%D1%8C_%D0%9B%D0%B8%D0%B7%D0%B0_%D0%98%D0%BD%D1%8C&action=edit&redlink=1" \o "Ицюнь Лиза Инь (страница отсутствует)) и [Хунбо Юй](https://ru.wikipedia.org/w/index.php?title=%D0%A5%D1%83%D0%BD%D0%B1%D0%BE_%D0%AE%D0%B9&action=edit&redlink=1" \o "Хунбо Юй (страница отсутствует)) (Xiaoyun Wang, Yiqun Lisa Yin, Hongbo Yu) представили атаку на полноценный SHA-1, которая требует менее 269 операций.

В августе 2005 года на CRYPTO 2005 специалисты представили улучшенную версию атаки на полноценный SHA-1, с вычислительной сложностью в 263 операций. В декабре 2007 года детали этого улучшения были проверены Мартином Кохраном.

Кристоф де Каньер и Кристиан Рехберг позже представили усовершенствованную версию атаки на SHA-1, за что были удостоены награды за лучшую статью на конференции ASIACRYPT 2006. Ими была представлена двух-блоковая коллизия на 64-раундовый алгоритм с вычислительной сложностью около 235 операций.

Существует масштабный исследовательский проект, стартовавший в технологическом университете австрийского города Грац, который : «… использует компьютеры, соединенные через Интернет, для проведения исследований в области криптоанализа. Вы можете поучаствовать в проекте загрузив и запустив бесплатную программу на своем компьютере.»

Хотя теоретически SHA-1 считается взломанным (количество вычислительных операций сокращено в 280-63 = 131 072 раза), на практике подобный взлом неосуществим, так как займёт пять миллиардов лет.

Из-за блочной и итеративной структуры алгоритмов, а также отсутствия специальной обработки в конце хеширования, все хеш-функции семейства SHA уязвимы для атак удлинением сообщения и коллизиям при частичном хешировании сообщения. Эти атаки позволяют подделывать сообщения, подписанные только хешем — SHA(message||key) или SHA(key || message ) — путём удлинения сообщения и пересчёту хеша без знания значения ключа. Простейшим исправлением, позволяющим защититься от этих атак, является двойное хеширование: SHAd(message)=SHA(SHA (0b|| message )) (0b — блок нулей той же длины, что и блок хеш-функции).

SHAppening

8 октября 2015 Marc Stevens, Pierre Karpman, и Thomas Peyrin опубликовали атаку на функцию сжатия алгоритма SHA 1, которая требует всего 257 вычислений.

Реальный взлом: SHAttered (нахождение коллизий)

23 февраля 2017 года специалисты из Google и CWI объявили о практическом взломе алгоритма, опубликовав 2 PDF-файла с одинаковой контрольной суммой SHA-1. Это потребовало перебора 9x1018 вариантов, что заняло бы 110 лет на 1 GPU.