МИНОБРНАУКИ РОССИИ

Федеральное общеобразовательное государственное учреждение

высшего образования

“Костромской государственный университет”

(КГУ)

Институт физико-математических и естественных наук

Защита информации

Направление подготовки/специальность 10.03.01

Информационная безопасность профиль Организация и технология защиты информации

Дисциплина “Криптографические методы защиты информации”

**КУРСОВАЯ РАБОТА**

Применение криптографических хеш-функций

Выполнил студент

Шулетов Михаил Александрович

Группа 17-ИББо-6

Проверил к.т.н., доцент

Щекочихин Олег Владимирович

Оценка\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Подпись преподавателя\_\_\_\_\_\_\_\_\_

Кострома

2019

**Аннотация**

Курсовая работа состоит из пояснительной записки в объёме 23 страниц, в том числе 8 иллюстраций. Пояснительная записка включает введение, 3 раздела, заключение, библиографический список. Список литературы содержит список наименований.

**Ключевые слова**: ХЕШИРОВАНИЕ, КРИПТОГРАФИЧЕСКАЯ ХЕШ-ФУНКЦИЯ, ПАРОЛЬ.

**Цель курсовой работы:** реализация криптографического алгоритма MD4.

**Задачи курсовой работы:**

1. узнать и познакомиться с криптографическими хеш-функциями;
2. познакомиться с принципом работы криптографических хеш-функций;
3. реализовать одну из криптографических хеш-функций в среде программирования.

**Оглавление**

[**Введение**](#_p3q7mysohqcy)4

[**1.Хеш-функция**](#_sg6x788o6gom)6

[1.1. Понятие хеш-функции](#_an2v74jnumg1) 6

[1.2. Свойства хеш-функций](#_mca4f4ik492p) 7

[1.3. Криптографические хеш-функции](#_30pquw25pb6) 8

[1.4. Виды криптографических хеш-функций](#_meoqe3befu4k) 9

[1.5. Требования хеш-функций](#_n7p09h5f1b4) 10

[**2. Алгоритм MD4**](#_aavv12k3cvr2)12

[2.1. История появления](#_5rmrhrdpk7o4) 12

[2.2. Применение MD4](#_ysxoa7gqjbg) 12

[2.2.1. Электронная подпись](#_ijuoukq0feac) 12

[2.2.2. Проверка парольной фазы](#_15saaa8at8pv) 13

[2.2.3. Хеширование биткоина](#_dpiu5fsh5o2r) 13

[2.4. Надёжность MD4](#_906dpez1jjyg) 16

[2.5. Сравнение с MD5](#_b2t8p4j3iri2) 16

[**3. Практическая часть**](#_px7ox347nl3s)17

[3.1. Реализация алгоритма MD4](#_wc1wz9izog6) 17

[3.2. Взлом MD4](#_cz0cjg9wfkrg) 20

[**4. Заключение**](#_o5bhw8mxebtt)22

[**5. Список литературы**](#_svovofd9jct0)23

# **Введение**

Слово хеш происходит от английского «hash», одно из значений которого трактуется как путаница или мешанина. Собственно, это довольно полно описывает реальное значение этого термина. Часто еще про такой процесс говорят «хеширование», что опять же является производным от английского hashing (рубить, крошить, спутывать и т.п.).

Появился этот термин в середине прошлого века среди людей занимающихся обработках массивов данных. Хеш-функция позволяла привести любой массив данных к числу заданной длины. Например, если любое число (любой длинны) начать делить много раз подряд на одно и то же простое число, то полученный в результате остаток от деления можно будет называть хешем. Для разных исходных чисел остаток от деления (цифры после запятой) будет отличаться.

В настоящее время практически ни одно приложение криптографии не обходится без использования хэширования.

При решении класса практических задач выбирается такая хеш-функция, которая является наиболее оптимальной именно для данного класса. В общем случае следует использовать «хорошую» функцию. Когда хеш-функцию называют «хорошей», то подразумевают под этим, что она:

* вычисляется достаточно быстро;
* сводит к минимуму число коллизий.

Предотвратить коллизии могут далеко не все хеш-функции, но «хорошие» способны минимизировать вероятность их появления. При определенных обстоятельствах (известна некоторая информация о ключах), можно найти идеальную хеш-функцию, т. е. такую, которая полностью исключает возможность появления коллизий.

Использовать хеш-функцию или нет, зависит от того насколько целесообразно применение ее свойств, а также свойств алгоритма, по которому она может быть реализована на различных языках программирования. В одних ситуациях наиболее важна высокая скорость работы, в других равномерное распределение хеш-кодов и т. п.. Далее будет подробно расписано о криптографических хеш-функциях, а также о двух известных алгоритмах хеширования.

# **1.Хеш-функция**

## **1.1. Понятие хеш-функции**

Хеш-функции – это функции, предназначенные для «сжатия» произвольного сообщения или набора данных, записанных, как правило, в двоичном алфавите, в некоторую битовую комбинацию фиксированной длины, называемую сверткой. Хеш-функции имеют разнообразные применения при проведении статистических экспериментов, при тестировании логических устройств, при построении алгоритмов быстрого поиска и проверки целостности записей в базах данных. Основным требованием к хеш-функциям является равномерность распределения их значений при случайном выборе значений аргумента.

Простые хеш-функции (ненадежные, но быстро рассчитываемые) применяются при проверке целостности передачи пакетов по протоколу TCP/IP (и ряду других протоколов и алгоритмов, для выявления аппаратных ошибок и сбоев — так называемое избыточное кодирование). Если рассчитанное значение хеша совпадает с отправленным вместе с пакетом (так называемой контрольной суммой), то значит потерь по пути не было (можно переходить к следующему пакету). Да, есть вероятность, что произойдет накладка — их называют коллизиями. Ведь для разных изначальных данных может получиться один и тот же хеш. Чем проще используется функция, тем выше такая вероятность. Но тут нужно просто выбирать между тем, что важнее в данный момент — надежность идентификации или скорость работы. В случае TCP/IP важна именно скорость. Но есть и другие области, где важнее именно надежность. Похожая схема используется и в [технологии блокчейн](https://ktonanovenkogo.ru/zarabotok_na_saite/dengi/blokchejn-chto-eto-takoe-prostymi-slovami.html), где хеш выступает гарантией целостности цепочки транзакций (платежей) и защищает ее от несанкционированных изменений. Благодаря ему и распределенным вычислениям взломать блокчейн очень сложно и на его основе благополучно [существует множество криптовалют](https://ktonanovenkogo.ru/zarabotok_na_saite/dengi/kriptovalyuta-chto-eto.html), включая самую популярную из них — [это биткоин](https://ktonanovenkogo.ru/zarabotok_na_saite/dengi/bitkoin-koshelek-kak-sozdat-oficialnom-sajte.html). Последний существует уже с 2009 год и до сих пор не был взломан.

Болеесложные хеш-функции используются в криптографии. Главное условие для них — невозможность по конечному результату (хешу) вычислить начальный (массив данных, который обработали данной хеш-функцией). Второе главное условие — стойкость к коллизиями, т.е. низкая вероятность получения двух одинаковых хеш-сумм из двух разных массивов данных при обработке их этой функцией. Расчеты по таким алгоритмам более сложные, но тут уже главное не скорость, а надежность. О них будет рассказано более подробно чуть позже.

## **1.2. Свойства хеш-функций**

Из всего сказанного в пункте 1.1, можно сформулировать свойства, которыми обладают хеш-функции:

1. Хеш-функция должна уметь приводить любой объем данных (а все они цифровые) к числу заданной длины (по сути это сжатие до битовой последовательности заданной длины хитрым способом);
2. Малейшее изменение (хоть на один бит) входных данных должно приводить к полному изменению хеша;
3. Функция должна быть стойкой в обратной операции, т.е. вероятность восстановления исходных данных по хешу должна быть весьма низкой (хотя последнее сильно зависит от задействованных мощностей);
4. Должна иметь как можно более низкую вероятность возникновения коллизий.
5. Хорошая хеш-функция не должна сильно нагружать компьютер при своем исполнении. От этого сильно зависит скорость работы системы на ней построенной. Всегда должен соблюдаться компромисс между скорость работы и качеством получаемого результата.
6. Алгоритм работы функции должен быть открытым, чтобы любой желающий мог бы оценить ее криптостойкость, т.е. вероятность восстановления начальных данных по выдаваемому хешу.

## **1.3. Криптографические хеш-функции**

Криптографической хеш-функцией называется всякая хеш-функция, являющаяся криптостойкой, то есть удовлетворяющая ряду требований специфичных для криптографических приложений. В криптографии хэш-функции применяются для решения следующих задач:

— построения систем контроля целостности данных при их передаче или хранении,

— аутентификация источника данных.

Хэш-функцией называется всякая функция h:X -> Y, легко вычислимая и такая, что для любого сообщения M значение h(M) = H (свертка) имеет фиксированную битовую длину. X — множество всех сообщений, Y — множество двоичных векторов фиксированной длины.

Как правило хэш-функции строят на основе так называемых одношаговых сжимающих функций y = f(x1, x2) двух переменных, где x1, x2 и y — двоичные векторы длины m, n и n соответственно, причем n — длина свертки, а m — длина блока сообщения.

Для получения значения h(M) сообщение сначала разбивается на блоки длины m (при этом, если длина сообщения не кратна m то последний блок неким специальным образом дополняется до полного), а затем к полученным блокам M1, M2,.., MN применяют следующую последовательную процедуру вычисления свертки:

Ho = v,

Hi = f(Mi,Hi-1), i = 1,.., N,

h(M) = HN

Здесь v — некоторая константа, часто ее называют инициализирующим вектором. Она выбирается из различных соображений и может представлять собой секретную константу или набор случайных данных (выборку даты и времени, например).

При таком подходе свойства хэш-функции полностью определяются свойствами одношаговой сжимающей функции.

## 

## **1.4. Виды криптографических хеш-функций**

Выделяют два важных вида криптографических хэш-функций — ключевые и бесключевые.

Ключевые хэш-функции называют кодами аутентификации сообщений. Они дают возможность без дополнительных средств гарантировать как правильность источника данных, так и целостность данных в системах с доверяющими друг другу пользователями.

Бесключевые хэш-функции называются кодами обнаружения ошибок. Они дают возможность с помощью дополнительных средств (шифрования, например) гарантировать целостность данных. Эти хэш-функции могут применяться в системах как с доверяющими, так и не доверяющими друг другу пользователями.

## **1.5. Требования хеш-функций**

Для криптографических хеш-функций важно, чтобы при малейшем изменении аргумента значение функции также сильно изменилось. Это называется лавинным эффектом.

К ключевым функциям хэширования предъявляются следующие требования:

— невозможность фабрикации,

— невозможность модификации.

Первое требование означает высокую сложность подбора сообщения с правильным значением свертки. Второе — высокую сложность подбора для заданного сообщения с известным значением свертки другого сообщения с правильным значением свертки.

К бесключевым функциям предъявляют требования:

— однонаправленность,

— устойчивость к коллизиям,

— устойчивость к нахождению второго прообраза.

Под однонаправленностью понимают высокую сложность нахождения сообщения по заданному значению свертки. Следует заметить что на данный момент нет используемых хэш-функций с доказанной однонаправленностью.

Под устойчивостью к коллизиям понимают сложность нахождения пары сообщений с одинаковыми значениями свертки. Обычно именно нахождение способа построения коллизий криптоаналитиками служит первым сигналом устаревания алгоритма и необходимости его скорой замены.

Под устойчивостью к нахождению второго прообраза понимают сложность нахождения второго сообщения с тем же значением свертки для заданного сообщения с известным значением свертки.

# **2. Алгоритм MD4**

## **2.1. История появления**

Самая быстрая, оптимизирована для 32-битных машин среди семейства MD-функций.

Хеш-функция, разработанная профессором Массачусетского университета Рональдом Ривестом в 1990 году и впервые описанная в [RFC](https://www.wikiwand.com/ru/RFC) 1186. Содержит три цикла по 16 шагов каждый. В 1993 году был описан алгоритм взлома MD4, поэтому на сегодняшний день данная функция не рекомендована для использования с реальными приложениями.

Для произвольного входного сообщения [функция](https://ru.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) генерирует 128-разрядное хеш-значение, называемое [дайджестом](https://ru.wikipedia.org/wiki/%D0%A5%D0%B5%D1%88-%D1%81%D1%83%D0%BC%D0%BC%D0%B0) сообщения. Этот алгоритм используется в протоколе [аутентификации](https://ru.wikipedia.org/wiki/%D0%90%D1%83%D1%82%D0%B5%D0%BD%D1%82%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D1%8F) [MS-CHAP](https://ru.wikipedia.org/wiki/MS-CHAP), разработанном корпорацией [Майкрософт](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D0%B9%D0%BA%D1%80%D0%BE%D1%81%D0%BE%D1%84%D1%82) для выполнения процедур проверки подлинности удаленных рабочих станций [Windows](https://ru.wikipedia.org/wiki/Windows). Является предшественником [MD5](https://ru.wikipedia.org/wiki/MD5).

## **2.2. Применение MD4**

### **2.2.1. Электронная подпись**

Чтобы убедиться, что сообщение отправил конкретный отправитель, вместе с сообщением передаётся так называемая электронная подпись. Получатель проверяет, действительно ли электронная подпись относится к данному сообщению.

В связи с тем, что использование криптографии с открытыми ключами (подписывание, проверка подписей и т. д.), процесс очень медленный, более того, если подписывать всё сообщение целиком, то размеры этой подписи будут сопоставимы с размером сообщения; подписывают не сообщение, а хеш-функцию от сообщения. И далее получатель, когда расшифровывает подпись, получает хеш-функцию. Далее он сравнивает хеш-функцию от того сообщения, которое он получил, и хеш-функцию, которая была получена в результате расшифровки. За счет того, что хеш-функция имеет фиксированную длину, она меньше, чем само сообщение. Это позволяет быстро вычислить электронную цифровую подпись. Размер этой подписи будет мал по сравнению с размером сообщения.

### **2.2.2. Проверка парольной фазы**

В большинстве случаев парольные фразы не хранятся на целевых объектах, хранятся лишь их хеш-значения. Хранить парольные фразы нецелесообразно, так как в случае несанкционированного доступа к файлу с фразами злоумышленник узнает все парольные фразы и сразу сможет ими воспользоваться, а при хранении хеш-значений он узнает лишь хеш-значения, которые не обратимы в исходные данные, в данном случае в парольную фразу. В ходе процедуры аутентификации вычисляется хеш-значение введённой парольной фразы, и сравнивается с сохранённым.

### **2.2.3. Хеширование биткоина**

Система биткойна является децентрализованной системой без выделенных центров генерации блоков. Каждый участник может взять набор транзакций, ожидающих включения в журнал, и сформировать новый блок. Более того, в системах типа BitCoin такой участник (майнер) ещё и получит премию в виде определённой суммы или комиссионных от принятых в блок транзакций

**2.3. Принцип работы MD4**

Алгоритм состоит из пяти шагов:

1. Добавление недостающих битов:

Сообщение расширяется так, чтобы его длина в битах по модулю 512 равнялась 448. Таким образом, в результате расширения, сообщению недостает 64 бита до длины, кратной 512 битам. Расширение производится всегда, даже если сообщение изначально имеет нужную длину.

Расширение производится следующим образом: один бит, равный 1, добавляется к сообщению, а затем добавляются биты, равные 0, до тех пор, пока длина сообщения не станет равной 448 по модулю 512. В итоге, к сообщению добавляется как минимум 1 бит, и как максимум 512.

1. Добавление длины сообщения:

64-битное представление b (длины сообщения перед добавлением набивочных битов) добавляется к результату предыдущего шага. В маловероятном случае, когда b больше, чем 2 в степени 64, используются только 64 младших бита. Эти биты добавляются в виде двух 32-битных слов, и первым добавляется слово, содержащее младшие разряды.

На этом этапе (после добавления битов и длины сообщения) мы получаем сообщение длиной кратной 512 битам. Это эквивалентно тому, что это сообщение имеет длину, кратную 16-ти 32-битным словам. Пусть M[0...N-1] означает массив слов получившегося сообщения (здесь N кратно 16).

1. Инициализация MD-буфера:

Для вычисления хеша сообщения используется буфер, состоящий из 4 слов (A,B,C,D)(32-битных регистров): . Эти регистры инициализируются следующими шестнадцатеричными числами (младшие байты сначала):

word A: 01 23 45 67

word B: 89 ab cd ef

word C: fe dc ba 98

word D: 76 54 32 10

1. Обработка сообщения блоками по 16 слов:

Для начала определим три вспомогательные функции, каждая из которых получает на вход три 32-битных слова, и по ним вычисляет одно 32-битное слово.







На каждую битовую позицию F действует как условное выражение: если X, то Y; иначе Z. Функция F могла бы быть определена с использованием вместо V, поскольку XY и \_XZ не могут равняться 1 одновременно. G действует на каждую битовую позицию как функция максимального значения: если по крайней мере в двух словах из X,Y,Z соответствующие биты равны 1, то G выдаст 1 в этом бите, а иначе G выдаст бит, равный 0. Интересно отметить, что если биты X, Y и Z статистически независимы, то биты F(X,Y,Z) и G(X,Y,Z) будут также статистически независимы. Функция H реализует побитовый xor, она обладает таким же свойством, как F и G.

1. Формирование хеша:

Результат (хеш-функция) получается как ABCD. То есть, мы выписываем 128 бит, начиная с младшего бита A, и заканчивая старшим битом D.

## 

## **2.4. Надёжность MD4**

Уровень безопасности, закладывавшийся в MD4, был рассчитан на создание достаточно устойчивых гибридных систем электронной цифровой подписи, основанных на MD4 и криптосистеме с открытым ключом. [Рональд Ривест](https://ru.wikipedia.org/wiki/%D0%A0%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%B4_%D0%A0%D0%B8%D0%B2%D0%B5%D1%81%D1%82) считал, что алгоритм хеширования MD4 можно использовать и для систем, нуждающихся в сильной криптостойкости. Но в то же время он отмечал, что MD4 создавался прежде всего как очень быстрый алгоритм хеширования, поэтому он может быть плох в плане криптостойкости. Как показали последовавшие исследования, он был прав, и для приложений, где важна прежде всего криптостойкость, стал использоваться алгоритм [MD5](https://ru.wikipedia.org/wiki/MD5).

## **2.5. Сравнение с MD5**

* + MD4 использует три цикла из 16 шагов каждый, в то время как MD5 использует четыре цикла из 16 шагов каждый.
  + В MD4 дополнительная константа в первом цикле не применяется. Аналогичная дополнительная константа используется для каждого из шагов во втором цикле. Другая дополнительная константа используется для каждого из шагов в третьем цикле. В MD5 различные дополнительные константы, Т[i], применяются для каждого из 64 шагов.
  + MD5 использует четыре элементарные логические функции, по одной на каждом цикле, по сравнению с тремя в MD4, по одной на каждом цикле.
  + В MD5 на каждом шаге текущий результат складывается с результатом предыдущего шага. Например, результатом первого шага является измененное слово А. Результат второго шага хранится в D и образуется добавлением А к циклически сдвинутому влево на определенное число бит результату элементарной функции. Аналогично, результат третьего шага хранится в С и образуется добавлением D к циклически сдвинутому влево результату элементарной функции. MD4 это последнее сложение не включает.

# **3. Практическая часть**

## **3.1. Реализация алгоритма MD4**

Как было сказано в аннотации, целью данной курсовой работы была реализация одного из алгоритмов хеширования. Я решил выбрать алгоритм MD4. Реализован данный алгоритм был на языке программирования C#.

**Структура программы**

Структура программы состоит из нескольких библиотек (рис. 1). Все библиотеки необходимы для повышения безопасности и производительности данной программы.

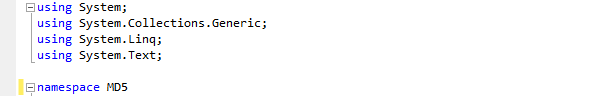
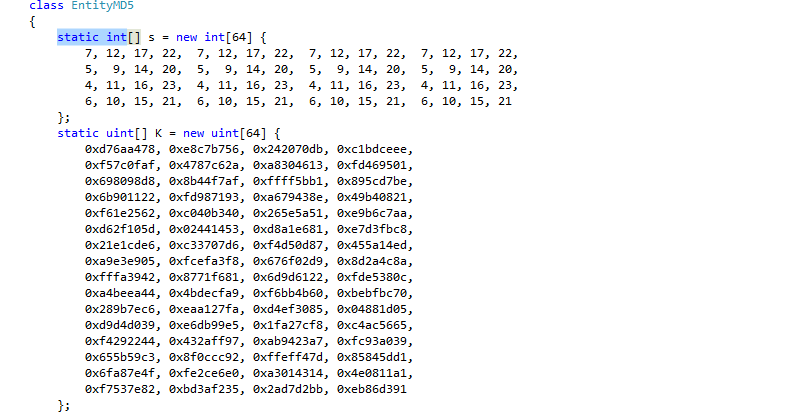
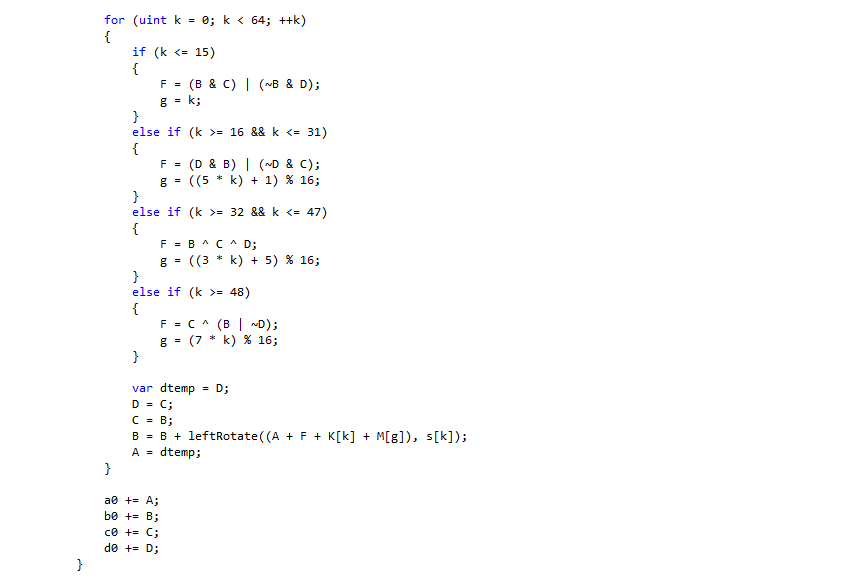


Рис. 1. Структура алгоритма

Далее происходит инициализация буфера (рис. 2). Для его инициализации требуется 4 переменных, причём его начальные значения задаются шестнадцатеричными числами.

Рис. 2. Инициализация буфера

На рис. 3 представлена часть программы, которая отвечает за вычисления хеша в цикле.

Рис. 3. Вычисление в цикле

**Интерфейс**

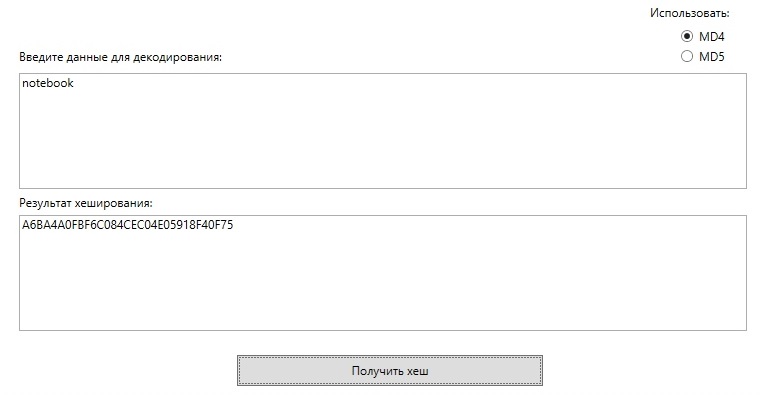
Теперь рассмотрим интерфейс данной программы (рис. 4). Он достаточно прост. 

Рис. 4. Интерфейс

## **3.2. Взлом MD4**

Но, как было сказано ранее в работе, данный алгоритм можно взломать. Для этого существует множество различных способов и программ, которые можно найти в интернете. Но я решил воспользоваться программой Hashcat, которая позволяет взламывать пароли с помощью подбора всех возможных вариантов. Да. если нам нужно будет хешировать очень длинный текст и потом впоследствии его взломать, на это уйдёт очень много времени. Однако для взлома паролей и небольших текстовых документов данная программа подходит неплохо.

В качестве исходного хеша возьмём рис. 4.

Итак, на рис. 5 представлен bat-файл, предназначенный для hashcat.

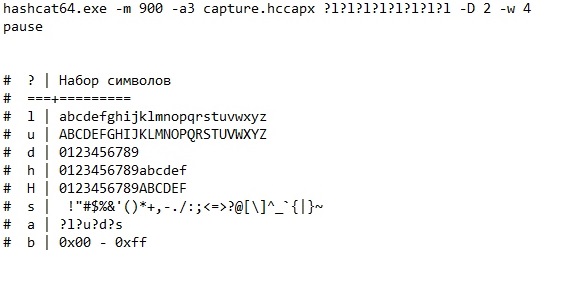


Рис. 5. Bat-файл для hashcat

Далее, полученный хеш мы записываем в отдельный файл (рис. 6).

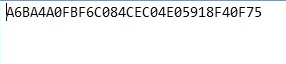


Рис. 6. Файл с искомым хешем

Затем данный с файл с нужным нам хешем отрабатывается в hashcat (рис. 7).

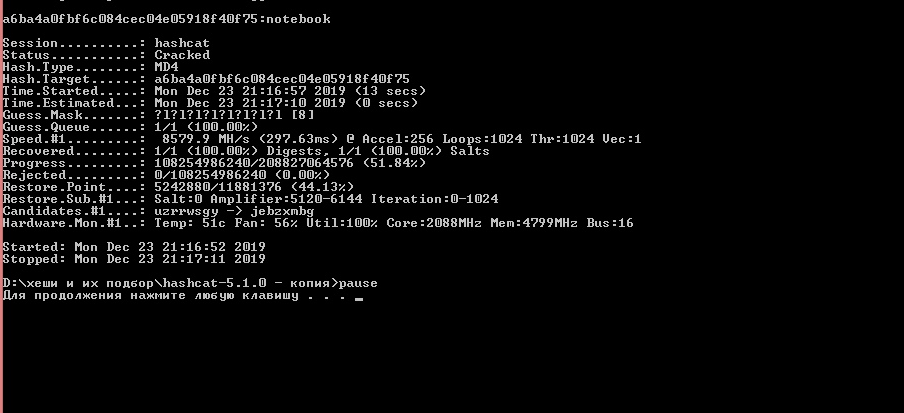


Рис. 7. Отработанный hashcat

Ну и на рис. 8. представлен результат работы программы. Из него видно, что данный хеш полностью совпадает с искомым хешем. 

Рис. 8. Результат работы hashcat

# **4. Заключение**

Хеширование в современном мире очень популярно. Оно используется в криптографии, электронных подписях, а также для проверки паролей. Хеш-функции необходимы для хеширования. Именно поэтому существует так много различных алгоритмов хеширования: SHA-256, множество видов алгоритма MD: MD2, MD4, MD5 и так далее. Они обладают всеми нужными свойствами, которые требуются при хешировании. И конечно же, все хеш-функции являются безопасными при их использовании.

Практически все алгоритмы хеширования предназначены для одной и той же цели. Однако и в прошлом, и в настоящем учёные смогли найти уязвимости для взлома криптографических хеш-функций. Именно по этой причине ни одна хеш-функция в настоящее время не может считаться по настоящему надёжной. В интернете можно найти множество различных программ для взлома многих алгоритмов хеширования. Однако криптография не стоит на месте и развивается: в ближайшее время будут разработаны новые, более надёжные криптографические хеш-функции, которые будут использоваться в хешировании.

# **5. Список литературы**

1. Б. Я. Рябко, А. Н. Фионов. Криптографические методы защиты информации. Учебное пособие /Б. Я. Рябко, А. Н. Фионов - Москва, 2005. — 229 с.
2. Брюс Шнайер. Прикладная криптография. Учебное пособие / 2002. — 815 с.
3. Хеш-алгоритмы [Электронный ресурс] // URL: <https://habr.com/ru/post/93226/>
4. Хеш-функции [Электронный ресурс] // MD4. URL: <http://kriptografea.narod.ru/MD4.html>
5. Хеш-функции [Электронный ресурс] // URL: <http://kvodo.ru/hesh-funktsii.html>
6. Что такое хеш [Электронный ресурс] // Свойства хеш-функций. URL: <https://ktonanovenkogo.ru/voprosy-i-otvety/xesh-chto-eto-takoe-xesh-funkciya.html>
7. Криптографическая хеш-функция [Электронный ресурс] // URL: <https://www.wikiwand.com/>
8. MD4 [Электронный ресурс] // URL: <https://ru.wikipedia.org/>
9. Что такое хеширование? [Электронный ресурс] // URL: <https://habr.com/ru/post/345740/>
10. Программы для взлома [Электронный ресурс] // URL: <https://xakep.ru/2012/10/29/x-tools-156/>