Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Российский химико-технологический университет имени Д.И. Менделеева»

Кафедра информационных компьютерных технологий

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 9

Выполнил студент группы КС-36 Алёшин Михаил Алексеевич

Ссылка на репозиторий: https://github.com/MUCTR-IKT-CPP/MAAleshin\_36\_algo

Приняли: Пысин Максим Дмитриевич

Краснов Дмитрий Олегович

Дата сдачи: 14.02.2022

Оглавление

[Описание задачи. 2](#_Toc63548272)

[Описание метода/модели. 2](#_Toc63548273)

[Выполнение задачи. 2](#_Toc63548274)

[Заключение. 2](#_Toc63548275)

# Описание задачи.

В рамках лабораторной работы необходимо реализовать MD5 хеширование. Для реализованной хеш функции провести следующие тесты:

* Провести сгенерировать 1000 пар строк длинной 128 символов отличающихся друг от друга 1,2,4,8,16 символов и сравнить хеши для пар между собой, проведя поиск одинаковых последовательностей символов в хешах и подсчитав максимальную длину такой последовательности. Результаты для каждого количества отличий нанести на график, где по оси х кол-во отличий, а по оси y максимальная длинна одинаковой последовательности.
* Провести N = 10^i(i от 2 до 6) генерацию хешей для случайно сгенерированных строк длинно 256 символов, и выполнить поиск одинаковых хешей в итоговом наборе данных, результаты привести в таблице где первая колонка это N генераций, а вторая таблица наличие и кол-во одинаковых хешей, если такие были.
* Провести по 1000 генераций хеша для строк длинной n (64, 128, 256, 512, 1024, 2048, 4096, 8192)(строки генерировать случайно для каждой серии), подсчитать среднее время и построить зависимость скорости расчета хеша от размера входных данных

# Описание метода/модели.

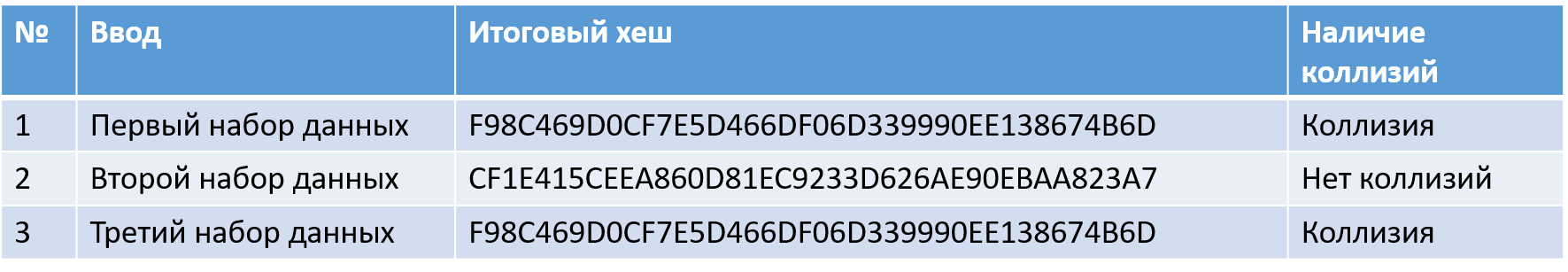
Хеширование, это применение особого алгоритма к некоторым входным данным произвольного типа, преобразующего их в битовую строку установленной длинны, строка часто выводиться в форме шестнадцатеричного чиста. Алгоритмом, который применяется для хеширования называют хеш-функцией или функцией свертки. Результат работы алгоритма хеширования называют хешем, хеш-кодом, хеш-суммой.

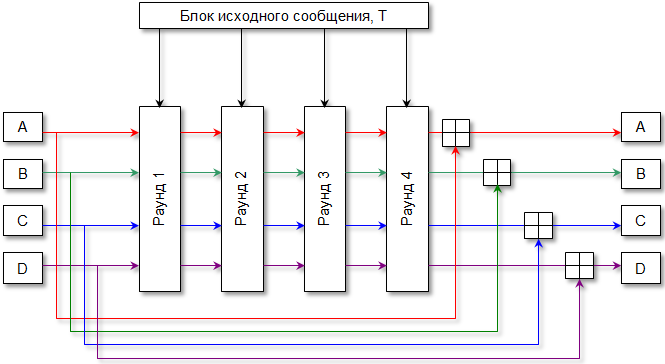
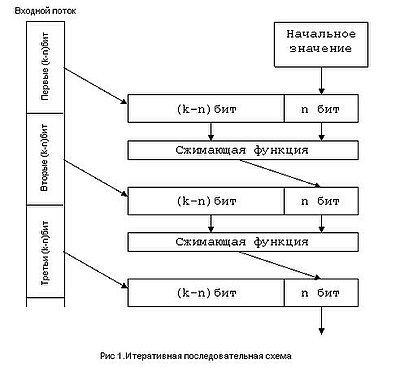
**Модель**

По умолчанию, базовая реализация хэш-функции должна удовлетворять следующим свойствам:

* Детерминированность – т.е. функция должна, в обязательном порядке, выдавать одинаковый вывод на одинаковы ввод.
* Скорость вычисления – функция должна быстро вычисляться, что бы ее можно было эффективно использовать для итеративных и постоянно возникающих процессов.
* Минимальное количество коллизий. – все хеширующие функции не гарантируют полное отсутствие колизий на бесконечном наборе входных данных.

Коллизии это пересечение значений выдаваемых хеш-функцией как результат своей работы для двух абсолютно разных вводов.





Идеальной хеш-функцией называется и считается та хеш функция которая никогда не дает одинаковых результатов на различные входные данные, т.е. никогда не возникает коллизий.

Для обеспечения такого результата хеширование производиться в 2 уровня.

Разумеется есть ограничение: идеальная хешь-функция существует только для конечного набора входных данных, т.е. предполагается что после создания контейнера состав ключей уже не меняется, что будет гарантировать быстроту и однозначность доступа к данным, но вызывать трудоемкую перегенерацию при каждом добавлении.

Примеры часто используемых хеш функций:

MD5 - Message Digest #5

SHA-1 - Secure Hash Algorithm 1 (1995) использует 160 бит для хранения результата

SHA-256 - Secure Hash Algorithm 2 использует 256 бит для хранения результата

SHA-512 - Secure Hash Algorithm 2 использует 512 бит для хранения результата

SHA-3 - Secure Hash Algorithm 3 новый вид алгоритма из семейства SHA

**Решение проблем коллизий**

Используя хеш-функцию можно составить таблицу содержащуюю пары ключ значение, где ключом будет является хеш, а значением будет является некоторое хранимое по этому ключу значение, таким образом будет получена хеш-таблица.

В таких хеш-таблицах используются простые хеш-функции, которые гораздо чаще чем криптографические хеш функции будут давать эффект коллизий.

Выделяют 2 метода для решения проблемы коллизий:

* Метод цепочек – составляется таблица итоговых хешей хранимых ключей, далее, в случае если возникает коллизия ключей, в эту таблицу записывается не один ключ, а список ключей, и в дальнейшем, при поиске данных в такой таблицы, мы сначала быстро перемещаемся по ней используя поиск по хешу, а затем ищем уже внутри списка ключей, размер которого предполагается несоизмеримом меньшим чем список хешей.
* Метод открытой адресации – составляется таблица итоговых хешей, которая хранит пары хеш – ключ, и при поиске по такой таблице, мы сначала находим нужный хеш после чего сверяемся с ключом, если он не совпадает ищем следующий подходящий хеш, и так до тех пор, пока не будет найден нужный ключ, или не кончаться варианты.

В обоих случая предполагается, что если искомое значение не было найдено, то создается новая пара и записывается в таблицу.

# Выполнение задачи.

Реализовано несколько функций для работы алгоритма MD5, такие как md5\_prepare(), md5\_padding(), md5\_group, md5(), hsh() и несколько логических функций. В функции hsh() выполняются все вызовы функций преобразований над строкой, такие как md5\_prepare(), которая подготавливает строку для разбития по блокам, md5\_padding() делает выравнивание, то есть, добавляет единичный бит и далее добавляет нули, пока длина сообщения не достигнет n % 512 = 448, далее в md5\_prepare() добавляется длина сообщения в формате little-endian. Далее в hsh() сообщение группируется по блокам функцией md5\_group() и отправляется в функцию md5(), в которой и происходит само хеширование и выдается конечный ответ в шестнадцатеричном формате. Далее проведем некоторые эксперименты. Сгенерируем 1000 пар строк длинной 128 символов отличающихся друг от друга 1,2,4,8,16 символов и сравним хеши для пар между собой, проведя поиск одинаковых последовательностей символов в хешах и подсчитав максимальную длину такой последовательности. Результаты для каждого количества отличий нанесем на график, где по оси х кол-во отличий, а по оси y максимальная длинна одинаковой последовательности.

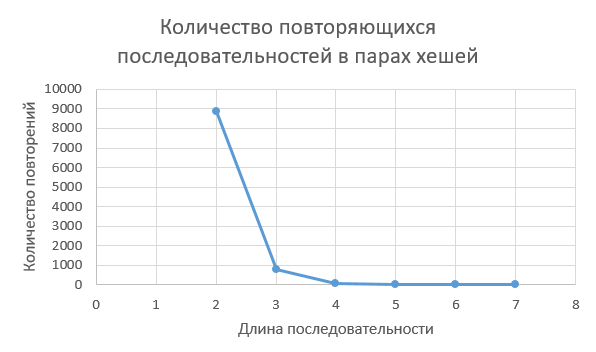


Рис. 1. Количество повторяющихся последовательностей в парах хешей.

На рис. 1 можно увидеть, что даже последовательность из 4 символов в парах строк из 128 символов на 5000 повторений встречаются очень мало раз. Далее проведем N = 10^i(i от 2 до 6) генераций хешей для случайно сгенерированных строк длинно 256 символов, и выполним поиск одинаковых хешей в итоговом наборе данных. В итоге на заданное количество генераций коллизий не было проявлено. Далее проведем по 1000 генераций хеша для строк длинной n (64, 128, 256, 512, 1024, 2048, 4096, 8192) (строки генерируются случайно для каждой серии), подсчитаем среднее время и построим зависимость скорости расчета хеша от размера входных данных

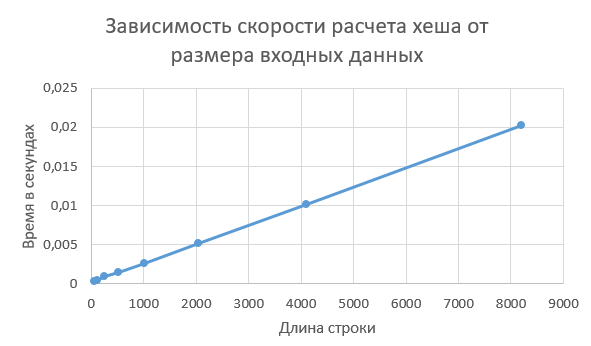


Рис. 2. Зависимость скорости расчета хеша от размера входных данных.

# Заключение.

Исходя из полученных нами данных можно увидеть, что хэши абсолютно похожих строк, различающихся хотя бы на 1 символ, будут абсолютно разными. Для самых разнообразных строк последовательности похожих срок/символов в хэщах будет, зачастую, не больше чем 2 в строках из 128 символов. Так же, для 10^6 элементов из строк длинной 256 символов не возникает коллизий, что еще раз показывает, что хеш является криптостойким. С увеличением длины строки не особо видно увеличение работы алгоритма, так как для строки 4096 элементов среднее время создания хеша 10^-3 секунд, а для 8192 элементов примерно в 2 раза больше времени. Из полученных результатов можно сделать вывод, что хеши хорошо подходят для сокрытия какой-либо информации (например, для хранения паролей в базе данных), так как на их расшифровку потребуется огромное количество времени и мощностей.