**Начало.**

Что-нибудь от себя.

Сегодня мы расскажем об алгоритме хеширования MD5. Этот 128-битный алгоритм хеширования был разработан профессором Рональдом Л. Ривестом из Массачусетского технологического института в 1991 году. Предназначен для создания «отпечатков» или хеш-сумм сообщения произвольной длины и последующей проверки их подлинности. Широко применялся для проверки целостности информации и хранения хешей паролей. (можно забить и в конце добавить, либо вообще не говорить Является прямым потомком алгоритма MD4, но с рядом отличий:

* В новый алгоритм добавили ещё один раунд, теперь их стало 4 вместо 3 в MD4.
* Добавили новую константу для того, чтобы свести к минимуму влияние входного сообщения.
* Изменились функция G
* Изменился порядок работы с входными словами в раундах 2 и 3.)

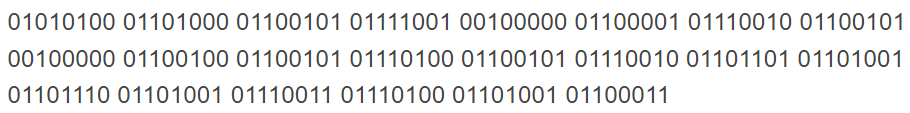
**Алгоритм.**

Теперь перейдем к самому интересному, а именно к тому, как работает сам алгоритм хеширования. Главная функция алгоритма MD5 – преобразование входных данных любой длины в фиксированные 128-битные строки (32 символа). Весь алгоритм мы будем объяснять сразу на конкретном примере, чтобы было как можно более понятно.

И так, на вход наш алгоритм будет получать строку: **«They are deterministic».**

Когда мы помещаем **«They are deterministic»** в хеш-функцию MD5, первое, что происходит, это то, что наша фраза преобразуется в двоичную. Это делается в соответствии с Американским стандартным кодом для обмена информацией (ASCII).

(Можно вставить табличку АСКИ в хорошем качестве, чтобы показать, что да, действительно, буквы так и преобразовались.)



Следующим шагом в MD5 является добавление нулевых бит или выравнивание. Входные данные в MD5 разбиваются на 512-битные блоки с добавлением заполнения для заполнения остального пространства в блоке. Наш ввод имеет длину 22 символа, включая пробелы, и каждый символ имеет длину 8 бит. Это означает, что входные данные составляют 176 бит. При вводе всего 176 бит в 512-битном блоке, который необходимо заполнить, нам нужно 336 бит для завершения блока.

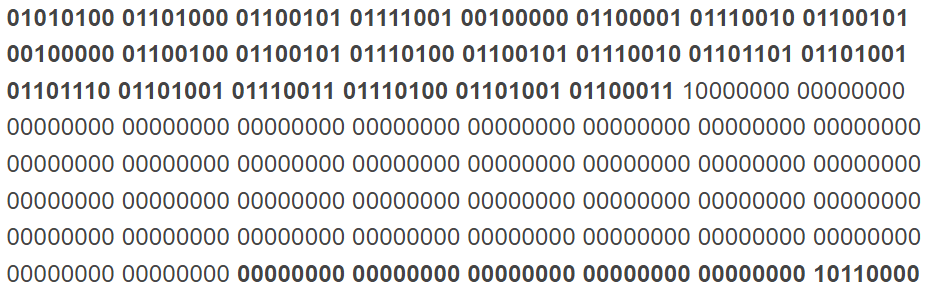
Схема заполнения MD5 кажется довольно странной. После размещения начальных 176 битов двоичного кода, которые представляют наш ввод, остальная часть блока дополняется одной единицей, а затем достаточным количеством нулей, чтобы довести его длину до 448 бит. Это происходит подобным образом:

448 – 1 – 176 = 271

Поэтому заполнение для этого блока будет включать единицу, а затем дополнительные 271 ноль. Причина, по которой нам нужно дополнить его только до 448 бит (вместо 512), заключается в том, что последние 64 бита (512 – 64 = 448) зарезервированы для отображения длины сообщения в двоичном виде. В нашем случае число 176 равно 10110000 в двоичном формате. Это формирует самый конец схемы заполнения, в то время как предыдущие 56 бит (64 минус восемь бит, составляющих 10110000) заполнены нулями.

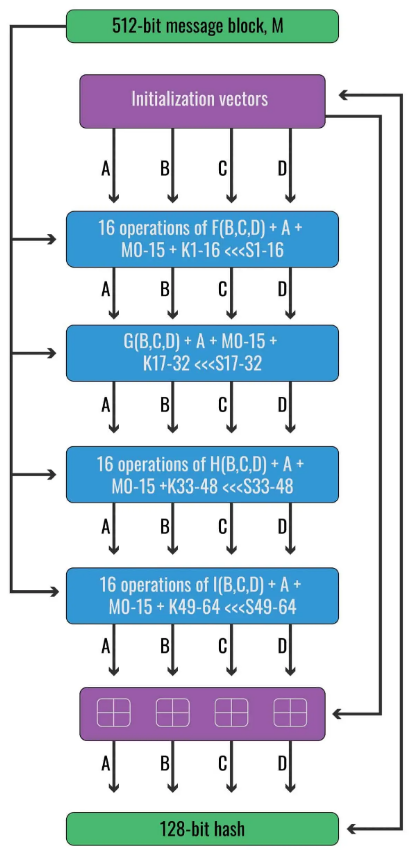
В тех случаях, когда длина сообщения занимает большее количество бит, нулей будет меньше. Если исходная длина входных данных превышает 64 бита (если она больше 264, что равно 18 446 744 073 709 551 616 в десятичном виде), то используются только младшие 64 бита. Т.е. мы всегда берем остаток от деления на 64.

Когда схема заполнения завершена, мы получаем следующую 512-битную строку:



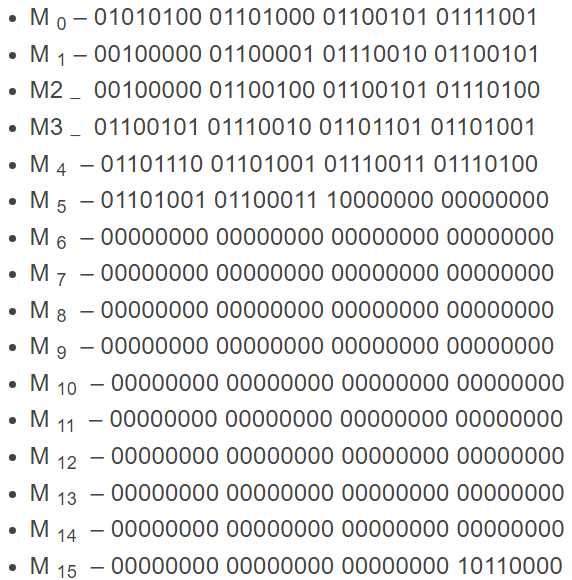
(это больше для нас Первые 176 бит (длина варьируется в зависимости от начального ввода) представляют наш начальный ввод **«They are deterministic»** в двоичном виде. Следующие 272 бита — это единица, за которой следует 271 ноль. Последние 64 бита — это длина нашего начального ввода (176 бит), записанного в двоичном виде.)

**Основной алгоритм.**

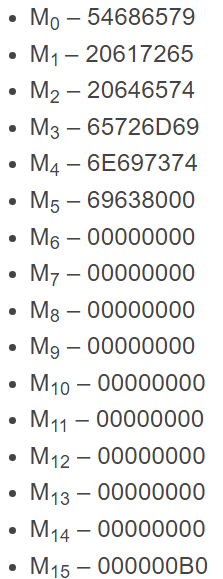


**Вход М –** Вверху у нас есть входные данные, которые говорят о 512-битном блоке сообщений, M. На этом этапе диаграммы она уже включает все отступы, которые мы добавили на последнем шаге. Если вы проследите за стрелкой вниз, вы увидите, что она входит в каждый из четырех прямоугольников «16 операций…». Каждый из этих четырех прямоугольников называется раундом, и каждый из них состоит из серии из шестнадцати операций.

Это означает, что наш ввод M является вводом на каждом из этих четырех этапов. Однако, прежде чем его можно будет использовать в качестве входных данных, наш 512-битный блок M необходимо разделить на шестнадцать 32-битных «слов». Каждому из этих слов присваивается собственный номер от M0 до M15. В нашем примере это 16 слов:



Для удобства представим эти числа в 16СС.



Каждое из этих шестнадцати значений выступает в качестве входных данных для сложного набора операций, которые представлены каждым прямоугольником «16 операций…». Хотя каждый из этих M входов используется в каждом отдельном раунде, они добавляются в разном порядке.

* В первом раунде в алгоритм последовательно добавляются M входов, например, M0, M1, M2… M15.
* Во втором раунде входы M добавляются в следующем порядке:

М1, М6, М11, М0, М5, М10, М15, М4, М9, М14, М3, М8, М13, М2, М7, М12

* В третьем раунде входы M добавляются в следующей последовательности:

М5, М8, М11, М14, М1, М4, М7, М10, М13, М0, М3, М6, М9, М12, М15, М2

* В четвертом раунде входы M добавляются в следующем порядке:

М0, М7, М14, М5, М12, М3, М10, М1, М8, М15, М6, М13, М4, М11, М2, М9

Для простоты наш пример будет придерживаться одного 512-битного блока данных, содержащего 16 слов.

**Векторы инициализации алгоритма MD5.** Теперь, когда мы немного объяснили наши входные данные M, пришло время обратить наше внимание на векторы инициализации , которые показаны чуть ниже 512-битного блока сообщений, M на диаграмме.

В начале векторы инициализации представляют собой четыре отдельных числа, указанных в RFC, стандарт описывающий MD5. Это:

* A – 01234567
* B – 89abcdef
* C – fedcba98
* D – 76543210

По мере продвижения по алгоритму эти числа будут заменены различными выходными данными, которые мы получаем в результате вычислений. Однако эти четыре вектора инициализации важны для начала работы. Все четыре из них являются входными данными первого прямоугольника «16 операций…».

**Функции MD5 F, G, H и I.** Первым шагом в прямоугольнике «16 операций…» является функция:

F(B, C, D) = (B∧C)∨(¬B∧D)

отметить, что векторы инициализации B, C и D используются в этой функции в качестве входных данных.

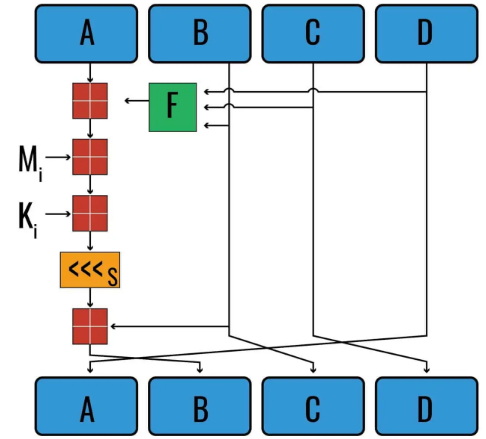
Функция F(B, C, D) используется для 16 операций первого раунда. В последующих раундах его место занимают следующие функции:

Раунд 2: G(B, C, D) = (B∧D)∨(C∧¬D)

Раунд 3: H(B, C, D) =B⊕C⊕D

Раунд 4: I(B, C, D) = C⊕(B∨¬D)

**Операции.** Эта диаграмма дает приблизительный обзор того, что происходит внутри каждого прямоугольника «16 операций…»:



Когда мы увеличим масштаб каждого прямоугольника «16 операций…», вы увидите, что стрелки от B, C и D указывают на прямоугольник с надписью F. Это представляет функцию F(B, C, D) — обратите внимание, что в другом три раунда функция F заменяется функциями G, H и I соответственно.

Если вы проследите за стрелкой из него в следующее поле, это означает, что выходные данные F(B, C, D) добавляются к вектору инициализации A со специальным типом добавления. В первой операции значение вектора инициализации A равно 01234567, но оно изменяется в последующих операциях.

Результат перемещается в следующее поле, где он добавляется к части ввода, представленной Mi. После этого к результату добавляется константа K с использованием того же специального типа сложения. Значения для K выводятся из формулы:

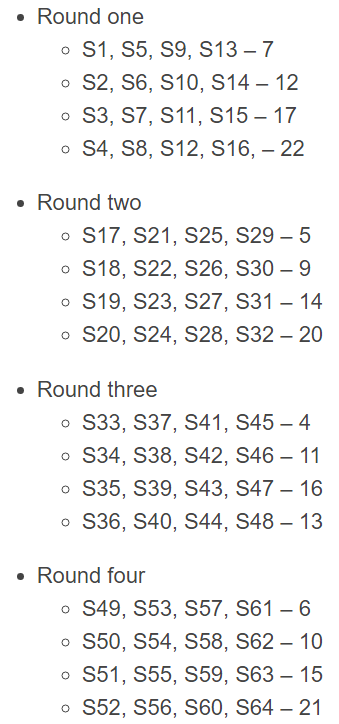
abs(sin(i+1))×2^32

Эта формула не слишком важна для понимания остальной части MD5. Однако нам нужны значения, к которым он приводит, а именно:

|  |  |  |
| --- | --- | --- |
|  |  |  |

Одно из этих значений K используется в каждой из 64 операций для 512-битного блока. K1-K16 используются в первом раунде, K17-K32 используются во втором раунде, K33-K48 используются в третьем раунде, а K49-K64 используются в четвертом раунде.

После добавления значения K следующим шагом является сдвиг количества битов влево на заданную величину Si. Величина, на которую сдвигается каждый бит, зависит от того, какую операцию выполняет алгоритм MD5. Каждая операция имеет предустановленное количество смен, и операции используют их в последовательном порядке (например, S1, S2, S3 и т. д.). Значения S:



После выполнения сдвига результат всех этих вычислений прибавляется к значению вектора инициализации B. Изначально оно равно 89abcdef, но в последующих операциях оно меняется.

Выход этого значения становится вектором инициализации для B в следующей операции. Векторы инициализации B, C и D перемешиваются на один пробел вправо, так что B становится C, C становится D, а D становится A в следующей операции.

**Четыре раунда по 16 операций.** Этот процесс зацикливается на 16 операций. Каждый раз предусмотренные выше входы используются для соответствующей операции. 17-я операция является началом второго раунда, и процесс продолжается аналогично, за исключением того, что вместо функции F используется функция G.

Все снова меняется к 33-й операции, когда функция H используется на протяжении третьего раунда. Четвертый раунд начинается с 49-й операции, вместо нее используется функция I.

По завершении четвертого раунда и его 64-й операции выходы добавляются к исходным векторам инициализации, которые мы перечислили выше. Результатом этого расчета является окончательный хеш MD5 нашего ввода.

**Итог первой операции.**

Теперь кратко разберем первую операцию, чтобы было более понятно.

Мы начали с наших четырех векторов инициализации:

A – 01234567

B – 89abcdef

C – fedcba98

D – 76543210

Мы помещаем эти последние три значения в функцию F:

F(B, C, D) = (B∧C)∨(¬B∧D)

Это дало нам результат:

F(B, C, D) = fedcba98

Мы взяли этот результат и поместили его в следующую формулу модульного сложения вместе с вектором инициализации A:

(X+Y) по модулю Z

Где:

Х = 01234567

Y = fedcba98

Z = 100000000

Это дало нам ответ:

ffffffff

Затем мы сделали еще несколько модульных сложений, на этот раз с первым словом нашего исходного ввода, M0, которое равно 54686579. Мы добавили его к результату последнего шага по той же формуле, которая дала нам:

54686578

Следующим шагом было еще одно модульное добавление, на этот раз с константой K, для которой мы перечислили значения в разделе «Операции» . K1 был d76aa478, который мы добавили к предыдущему результату, получив результат:

2bd309f0

На следующем шаге все смешалось, и мы преобразовали шестнадцатеричный результат в двоичный, чтобы мы могли легко увидеть семибитный сдвиг влево. Когда мы изменили его обратно на шестнадцатеричный, результат был таким:

e984f815

Затем мы вернулись к модульному сложению, добавив этот результат к вектору инициализации B, 89abcdef . В итоге мы получили следующее значение, которое становится вектором инициализации B в следующем раунде:

7330c604

Значения для B, C и D также были перетасованы вправо, что дало нам новые векторы инициализации для следующей операции:

A – 76543210

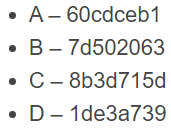
B – 7330c684

C – 89abcdef

D – fedcba98

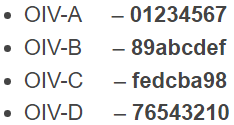
**Последний шаг алгоритма MD5 после 64 операций.**

64-я операция выполняется так же, как и все предыдущие, используя выходные данные 63-й операции в качестве векторов инициализации для функции I. Когда он прошел каждый из шагов операции, он дает нам новые значения для A, B, C и D. Предположим:



К этому моменту мы закончили нижний прямоугольник с надписью «16 операций…». Если вы проследите за стрелками вниз, то увидите, что они соединены с прямоугольником с четырьмя расчетами модульного сложения . Другие входные данные поступают из векторов инициализации, которые мы использовали в самом начале алгоритма MD5.

На этом этапе давайте будем называть их OIV-A, OIV-B и т. д. для исходного вектора инициализации A, B и т. д. Это должно помочь сохранить ясность. Этими исходными векторами инициализации были:



Еще раз, наше уравнение для модульного сложения:

(Х + Y) по модулю Z

В этом случае:

X – 60cdceb1 (выход для вектора инициализации A после 64-й операции).

Y – 01234567 (ОИВ-А).

Z – 100000000 (это 232).

Результатом является окончательное значение A = 61f11418

Нам нужно сделать то же самое уравнение для наших оставшихся чисел:

(А + В) по модулю С

Следовательно, нам нужно решить:

(B + OIV-B) mod С

(С + OIV-С) mod С

(D+OIV-D) mod С

Когда мы подставляем соответствующие числа, мы получаем:

(7d502063 + 89abcdef) мод 100000000

(8b3d715d + fedcba98) мод 100000000

(1de3a739 + 76543210) мод 100000000

В результате получает следующие значения:

B – 06fbee52

C – 8a1a2bf5

D – 9437d949

Все готово, чтобы получить окончательный хеш.

Хэш = ABCD

H(They are deterministic) = 61f1141806fbee528a1a2bf59437d949

(Если вообще на это будет время)

**Криптоанализ**

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

* Перебор по словарю
* Brute-force
* RainbowCrack
* Коллизия хеш-функции — это получение одинакового значения функции для разных сообщений и идентичного начального буфера.

**Выводы**

Основная задача любой функции хеширования сообщений -производить образы, которые можно считать относительно случайными. Чтобы считаться криптографически безопасной, хэш-функция должна отвечать двум основным требованиям. Во-первых, злоумышленник не может сгенерировать сообщение, соответствующее определенному хеш-значению. Во-вторых, невозможно создать два сообщения, которые производят одно и то же значение (коллизии в MD5).

К сожалению, выяснилось, что алгоритм MD5 не способен отвечать данным требованиям. IETF (Internet Engineering Task Force) рекомендовала новым проектам протоколов не использовать MD5, так как исследовательские атаки предоставили достаточные основания для исключения использования алгоритма в приложениях, которым требуется устойчивость к различного рода коллизиям.

Хеши MD5 больше не считаются безопасными, и их не рекомендовано использовать для криптографической аутентификации.