ХЭШ (часть II)

www.shumkoff.ru

Шумков Е.А. КубГТУ

АЛГОРИТМ MD5 (1991 г.)

Целостность информации — термин в информатике (криптографии, теории телекоммуникаций, теории информационной безопасности), означающий, что данные не были изменены при выполнении какой-либо операции над ними, будь то передача, хранение или отображение.
В телекоммуникации целостность данных часто проверяют, используя хеш-сумму сообщения

Предшественник MD5 – алгоритм MD4, а до него MD2

Автор – Рональд Линн Ривест. Профессор МИТ. Родился в 1947 году. В 2002 году получил премию Тьюринга.

Общая постановка проблемы:
Пусть у нас есть набор данных, для простоты будем рассматривать натуральные числа от 1 до 10^6 . И пусть есть некоторая функция, в которой один параметр — натуральное число от 1 до 10^6, а возвращаемое значение — натуральное число от 1 до 1000. Собственно, нам не важно, что конкретно делает эта функция, а важно то, что она каждому натуральному числу от 1 до 10^6 ставит в соответствие другое натуральное число от 1 до 1000.Тогда:

```
int hash(long int x)
{
    if (x%1000==0) return 1000;
    return (x % 1000);
}
```

Собственно – это и есть простая хеш-функция. Если мы знаем параметр функции, то однозначно можем сказать, какой будет результат. А если нам известен результат, то можем ли мы узнать однозначно параметр? Конечно, нет. Для числа 234 параметр может быть 234,1234, 2234,3234… Поэтому однозначно восстановить параметр не получится.

Для функции из примера, если известен результат, можно легко найти параметр, для которого будет такой же результат. А вот для функции MD5 это сделать не так-то просто. Т.е. если у нас есть только результат функции MD5, то мы не сможем найти параметр, для которого функция выдаст этот же результат (речь даже не идет про однозначное восстановление параметра).

Алгоритм удовлетворяет следующим условиям:

1. **При одинаковых начальных значениях результат работы должен быть также одинаковый**
2. **При незначительных изменениях входных данных, результат функции будет сильно отличаться**
3. **Трудно найти две пары значений с одинаковыми MD5**
4. **Результат работы функции MD5 - строка фиксированной длины**

Пусть требуется шифровка паролей. Можно поступить следующем образом:

1. **В начале пароля и в конце добавляем произвольные переменные.**

Допустим, что у нас пароль – «password». В начало пароля добавим восклицательный знак, а в конец точку. Получим «!password.»

2. **Полученную строку обработаем функцией md5 и результат сохраняем**

Теперь чтобы проверить правильность введенного пароля делаем следующее:

1. **Сохраняем введенный пользователем пароль в переменную**
2. **В начало и в конец переменной добавляем все те же «!» в начало и «.» в конец**
3. **Применяем функцию MD5**
4. **Сравниваем результат с тем, что хранится в базе. Результаты должны совпасть**

http://fegorsk.ru

Итак, алгоритм MD5

1. Пусть q будет длина получившейся последовательности (ровно 64 бита, возможно, с незначащими нулями). К получившейся последовательности приписывается 1. в результате длина последовательности увеличивается на 1.

2. Затем к последовательности приписываются нули, пока длина не станет по модулю 512 равна 448 (length mod 512=448).

3. Далее к последовательности дописываются младшие 32 бита числа q, а затем — старшие. Длина последовательности становится кратной 512. Полученную последовательность назовем S.

4. Для подсчета результата используются четыре двойных слова (32 бита). Эти двойные слова инициализируются следующими шестнадцатеричными значениями, где первым следует самый младший байт:

A: 01 23 45 67

B: 89 ab cd ef

C: fe dc ba 98

D: 76 54 32 10

Также для подсчета результата используются следующие функции:

F(X,Y,Z) = XY v not(X) Z

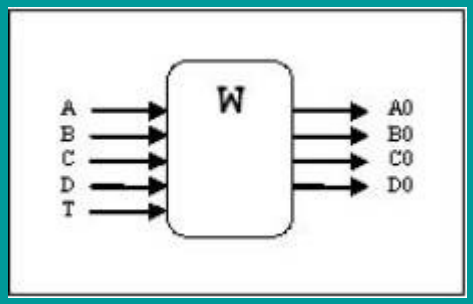
G(X,Y,Z) = XZ v Y not(Z)

H(X,Y,Z) = X xor Y xor Z

I(X,Y,Z) = Y xor (X v not(Z))

X,Y,Z — это двойные слова. Результаты функций, также двойные слова. Для подсчета используется еще одна функция (назовем ее W). Она хитро обрабатывает данные и возвращает результат

(функцию W описывать не будем, см., например, Википедию)
Обработка данных происходит с использованием функций F, G, H, I.



Подготовили данные, далее расчет результата:
а) Запоминаем первые 512 бит последовательности S
Б) Удаляем первые 512 бит последовательности S (можно обойтись и без удаления, но тогда на первом шаге надо брать не первые 512, а следующие 512 бит)
В) Вызываем функцию W. Параметры A,B,C,D — это текущие значения соответствующих двойных слов. Параметр T — это запомненные 512 бит.
Г) Прибавляем к A A0
Д) B=B+B0 , C=C+C0 , D=D+D0
Е) Если длина последовательности 0, выходим
Ж) Переходим к шагу 1
После выполнения данного алгоритма А)-Ж) - результат работы MD5.
A,B,C,D — это результат (его длина будет 128 бит).

Часто можно видеть результат MD5 как последовательность из 32 символов 0..f. Это то же самое, только результат записан не в двоичной системе счисления, а в шестнадцатеричной.

КАК МОЖНО ПРИМЕНЯТЬ MD5 (несколько вариантов, но не все):
-пароли обычно хранятся в зашифрованном виде (при помощи MD5). После того, как пользователь введет свой пароль, от пароля вычисляется хэш-функция MD5. Результат сравнивается со значением, хранящимся в базе. Если значения равны, то пароль верен.

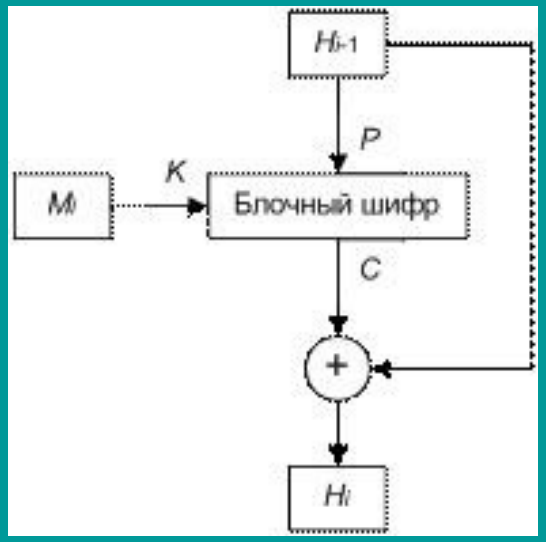
-Еще MD5 можно использовать в качестве контрольной суммы. Предположим, -необходимо куда-то скопировать файл. Причем нет никаких гарантий, что файл будет -доставлен без повреждений. Перед отправкой можно посчитать MD5 от содержимого - файла и передать результат вместе с файлом. Затем посчитать MD5 от принятого файла - и сравнить два результата. Если результаты различные, то это означает, что файл -или результат был испорчен при передаче.

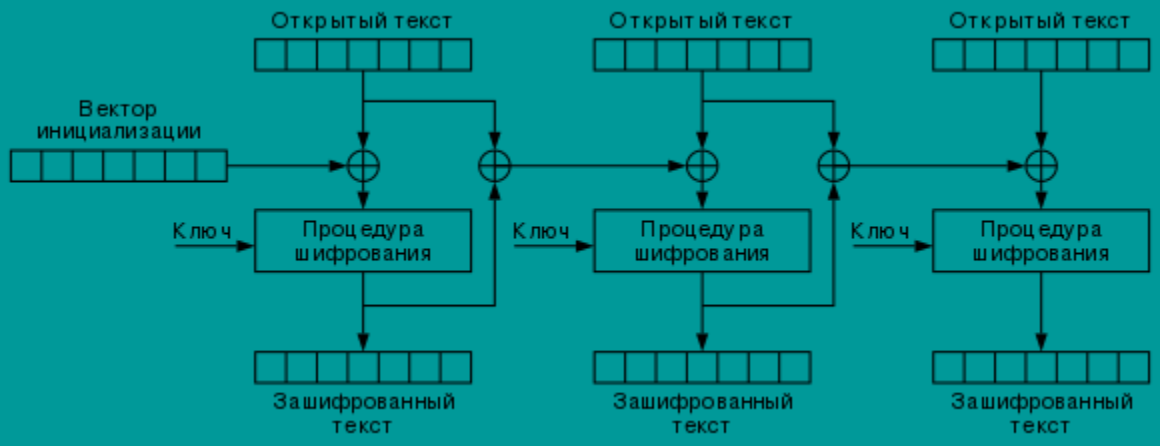
-- Последнее время MD5 стали использовать интернет-казино. Перед тем, как сделать -ставку, игрок получает хэш от результата игры. Когда ставка сделана, игрок получает -результат игры (например, выпало число 26). Посчитав от результата хэш-функцию, -можно убедиться, что казино сгенерировало это число до того, как игрок сделал ставку. -Но не стоит думать, что выиграть в этом казино очень просто. Весь секрет в том что, - вероятность выигрыша подобрана таким образом, что игрок почти всегда -будет в проигрыше. // http://www.nestor.minsk.by

Алгоритм MD5 происходит от MD4. 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, в каждом раунде на каждом шаге и каждый раз константа разная, она суммируется с результатом F и блоком данных.

Коллизия хеш-функции — это получение одинакового значения функции для разных сообщений и идентичного начального буфера. В отличие от коллизий, *псевдоколлизии* определяются как равные значения хеша для разных значений начального буфера, причём сами сообщения могут совпадать или отличаться. В MD5 вопрос коллизий не решается. В 1996 году Ганс Доббертин нашёл псевдоколлизии в MD5, используя определённые *инициализирующие* векторы, отличные от стандартных. Оказалось, что можно для известного сообщения построить второе, такое, что оно будет иметь такой же хеш, как и исходное. С точки зрения математики это означает:
MD5(IV,L1) = MD5(IV,L2), где IV — начальное значение буфера,
а L1 и L2 — различные сообщения. *allrefrs.ru*

В 2005 году Ван Сяоюнь и Юй Хунбо из университета Шаньдуна в Китае опубликовали алгоритм, который может найти две различные последовательности в 128 байт, которые дают одинаковый MD5-хеш.





Блочное шифрование

Схемы - Википедия

Домашнее задание – схемы Мейера – Матиаса и Девиса - Мейера

АЛГОРИТМЫ ПОИСКА ПОДСТРОКИ В СТРОКЕ

Поиск подстроки в строке — одна из простейших задач поиска информации. Применяется в виде встроенной функции в текстовых редакторах, СУБД, поисковых машинах, языках программирования и т. п.

В задачах поиска традиционно принято обозначать шаблон поиска как needle (с англ. ' — ' «иголка»), а строку, в которой ведётся поиск — как haystack (с англ. ' — ' «стог сена»).

На сегодняшний день существует огромное разнообразие алгоритмов поиска подстроки. Программисту приходится выбирать подходящий в зависимости от различных факторов

Основные алгоритмы поиска подстроки:
-Алгоритм Бойера – Мура
-Алгоритм Бойера – Мура Хорспула
-Алгоритм Санди
-Алгоритм Рабинера-Карпа
-Алгоритм Ахо-Корасик
-и мн. другие

АЛГОРИТМ БОЙЕРА-МУРА

Если стоп-символ «к» оказался за другой буквой «к», эвристика стоп-символа не работает.

```
Строка:          * * * * к к о л * * * * *
Шаблон:            к о л о к о л
Следующий шаг:   к о л о к о л       ?????
```

В таких ситуациях выручает третья идея АБМ — эвристика совпавшего суффикса.

3. Эвристика совпавшего суффикса. Если при сравнении строки и шаблона совпало один или больше символов, шаблон сдвигается в зависимости от того, какой суффикс совпал.

```
Строка:          * * т о к о л * * * * *
Шаблон:          к о л о к о л
Следующий шаг:             к о л о к о л
```

В данном случае совпал суффикс «окол», и шаблон сдвигается вправо до ближайшего «окол». Если подстроки «окол» в шаблоне больше нет, но он начинается на «кол», сдвигается до «кол», и т. д.

Алгоритм Бойера — Мура

```
void boyer_moore_search(char* text, char* pattern,
                        pattern_found_callback callback) {
    int i, j, char_pos[UCHAR_MAX + 1];
    memset(char_pos, -1, sizeof(char_pos));
    for (i = 0; pattern[i]; ++i) {
        char_pos[(unsigned char) pattern[i]] = i;
    }
    int n = strlen(text), m = strlen(pattern);
    for (i = 0; i + m <= n; ) {
        for (j = m - 1; 0 <= j; --j) {
            if (text[i + j] != pattern[j]) {
                i += max(1, j - char_pos[text[i + j]]);
                break;
            }
        }
        if (j < 0) {
            if (!(*callback)(text, pattern, i)) { return; }
            ++i;
        }
    }
}
```