Тема: Криптографические хэш-функции

Однонаправленные хэш-функции

Семейство алгоритмов SHA

Семейство алгоритмов SHA (Secure Hash Algorithm) включает в себя 5 алгоритмов вычисления хэш-функции: SHA–1, SHA–224, SHA–256, SHA–384, SHA–512.

Четыре последние хэш-функции объединяются в подсемейство SHA–2. Алгоритм SHA–1 разработан Агентством национальной безопасности США (NSA) в 1995 году.
Алгоритмы подсемейства SHA–2 также разработаны Агентством национальной безопасности США и опубликованы Национальным институтом стандартов и технологий в федеральном стандарте обработки информации FIPS PUB 180–2 в августе 2002 года.

Эти алгоритмы используются в SSL, SSH, и т.д…, при передаче файлов по сети (BitTorrent).

Между собой алгоритмы отличаются криптостойкостью, которая обеспечивается для хэшируемых данных, а также размерами блоков и слов данных, используемых при хэшировании. Основные отличия алгоритмов можно представить в виде таблицы :

Алгоритм	Длина блока (бит)	Длина сообщения (бит)	Длина слова (бит)	Количество итераций в цикле
SHA–1	512	$< 2^{64}$	32	80
SHA–224	512	$< 2^{64}$	32	64
SHA–256	512	$< 2^{64}$	32	64
SHA–384	1024	$< 2^{128}$	64	80
SHA–512	1024	$< 2^{128}$	64	80

При описании алгоритма под термином ***слово понимается 32-битная последовательность (SHA-1, SHA-224, SHA-256), а под термином байт – 8-битная последовательность.***

Последовательность бит может быть интерпретирована естественным образом как последовательность байт, где каждая последовательная группа из 8 бит представляет собой 1 байт. Внутри байта биты располагаются следующим образом: сначала (слева) перечисляются более значимые биты (старшие биты, соответствующие более высокой степени двойки), а в конце (справа) оказываются наименее значимые биты (младшие). Такой порядок расположения бит (или байт) называется **big-endian (порядок от старшего к младшему).**

В алгоритмах используются побитные операции '$\wedge$', '$\vee$', '$\neg$', '$\oplus$', '>>' и '<<'. Под операцией '+' понимается сложение по модулю 2^{32} (или 2^{64} в зависимости от длины слова), т.е. $(X + Y) \pmod{2^{32}}$.

Про циклический сдвиг пример и операции пример

Также используются следующие обозначения для операций сдвига:

- сдвиг вправо $SHR^n(x) \equiv x >> n$;
- циклический сдвиг вправо $ROTR^n(x) \equiv (x >> n) \vee (x << w - n)$, где w – длина слова;
- циклический сдвиг влево $ROTL^n(x) \equiv (x << n) \vee (x >> w - n)$, где w – длина слова;

В алгоритмах используются следующие функции:

- $Ch(x, y, z) = (x \wedge y) \oplus (\neg x \wedge z)$;
- $Parity(x, y, z) = x \oplus y \oplus z$;
- $Maj(x, y, z) = (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z)$;
- $$f(x, y, z) = \begin{cases} Ch(x, y, z), & 0 \le t \le 19, \\ Parity(x, y, z), & 20 \le t \le 39, \\ Maj(x, y, z), & 40 \le t \le 59, \\ Parity(x, y, z), & 60 \le t \le 79. \end{cases}$$

Описание алгоритма.

Пусть имеется сообщение M длиной в l бит, хэш-функцию которого нужно вычислить.

Шаг 1. *Добавление дополнительных бит.*

К концу сообщения добавляется бит $'1'$, вслед за которым добавляются нулевые биты ($'0'$) до тех пор, пока длина L дополненного сообщения не станет удовлетворять условию: $L \equiv 448 \pmod{512}$. Таким образом, может быть добавлено от 1 до 512 бит.

Для *SHA–384* и *SHA–512*: $L \equiv 896 \pmod{1024}$, и соответственно может быть добавлено от 1 до 1024 бит.

Шаг 2. *Добавление исходной длины сообщения.*

К результату предыдущего шага добавляются младшие 64 бита (128 бит для *SHA–384* и *SHA–512*), взятые из побитного представления числа l (исходная длина сообщения). При этом для конвертирования последовательности бит в число используется порядок расположения бит big-endian. В итоге получается сообщение, длина которого кратна 512 битам (1024 битам). Например, если длина слова – 32 бита, то сообщение **"abc"**, имеющее длину 24 бита, будет дополнено до следующего сообщения (см. рисунок 1):

$$\underbrace{01100001}_{\text{"a"}}\ \underbrace{01100010}_{\text{"b"}}\ \underbrace{01100011}_{\text{"c"}}\ 1\ \overbrace{00\ldots00}^{423}\ \overbrace{00\ldots0\underbrace{11000}_{l=24}}^{64}$$

Рисунок 1 – Сообщение "abc", дополненное битами '1' и '0' и исходной длиной сообщения

Полученное сообщение разбивается на блоки длиной 512 (или 1024) бита, которые обозначаются $M^{(1)}, M^{(2)}, \ldots, M^{(N)}$. Каждый из блоков подразделяется на 16 32-битных (или 64-битных) слов, обозначаемых $M_0^{(i)}, M_1^{(i)}, \ldots, M_{15}^{(i)}$.

Дальнейшие шаги специфичны для каждого алгоритма

Алгоритм SHA–1

Шаг 3. *Определение констант.*

SHA–1 использует последовательность 80 32-битных слов $(K_0, K_1, \ldots, K_{79})$, где (в 16-ричном виде):

$$K_t = \begin{cases} 5a827999, & 0 \le t \le 19, \\ 6ed9eba1, & 20 \le t \le 39, \\ 8f1bbcdc, & 40 \le t \le 59, \\ ca62c1d6, & 60 \le t \le 79. \end{cases}$$

Начальное значение хэш-функции устанавливается следующими константами:

$H_0^{(0)}$ = 67452301, $H_1^{(0)}$ = efcdab89, $H_2^{(0)}$ = 98badcfe, $H_3^{(0)}$ = 10325476,
$H_4^{(0)}$ = c3d2e1f0.

Шаг 4. *Основной цикл.*

В следующем цикле последовательно обрабатываются все блоки, на которые было разделено дополненное сообщение:

```
For i = 1 to N
  {
   // 1. Подготовка списка преобразованных слов сообщения
```

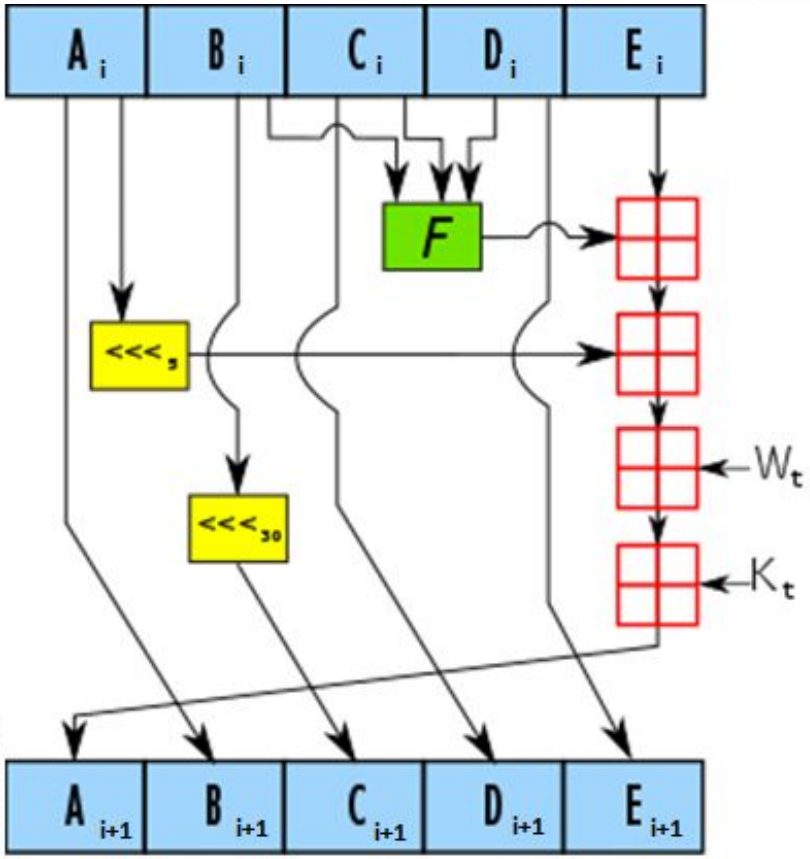
$$W_t = \begin{cases} M_t, & 0 \le t \le 15, \\ ROTL^1(W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16}), & 16 \le t \le 79. \end{cases}$$

```
   // 2. Инициализация рабочих переменных
```

a = $H_0^{(i-1)}$ b = $H_1^{(i-1)}$ c = $H_2^{(i-1)}$ d = $H_3^{(i-1)}$ e = $H_4^{(i-1)}$

```
// 3. Внутренний цикл
For t = 0 to 79
   {
    T = ROTL^5(a) + f_t(b,c,d) + e + K_t + W_t
    e = d
    d = c
    c = ROTL^30(b)
    b = a
    a = T
   }

// 4. Вычисление промежуточного значения хэш-функции
H_0^(i) = a + H_0^(i-1)     H_1^(i) = b + H_1^(i-1)     H_2^(i) = c + H_2^(i-1)
H_3^(i) = d + H_3^(i-1)     H_4^(i) = e + H_4^(i-1)
}
```



Шаг 5. *Результат.*
Результатом вычисления хэш-функции от всего сообщения является:

$$H_0^{(N)} \parallel H_1^{(N)} \parallel H_2^{(N)} \parallel H_3^{(N)} \parallel H_4^{(N)}.$$

‖ — конкатенация- операция склеивания объектов линейной структуры, обычно строк.
Конкатенация — бинарная операция, определённая на словах данного алфавита. Обозначения:

- A — алфавит, набор букв;
- α, β, γ — слова, составленные из букв;
- $a_1 \dots a_n$ и $b_1 \dots b_m$ — записанные подряд и пронумерованные буквы двух слов.

Если $\alpha = a_1 \dots a_n$ и $\beta = b_1 \dots b_m$ — слова в алфавите A, то конкатенацией слов α и β, которую обозначим в этой статье как $\alpha \cdot \beta$, будет слово γ в том же алфавите A, определяемое равенством

$$\gamma = \alpha \cdot \beta = a_1 \dots a_n b_1 \dots b_m.$$

Примеры значений хэш-функции (в 16-ричном виде):
SHA1 ("") = da39a3ee 5e6b4b0d 3255bfef 95601890 afd80709
SHA1 ("abc") = a9993e36 4706816a ba3e2571 7850c26c 9cd0d89d
SHA1 ("abcdbcdecdefdefgefghfghighijhijkijkljklmklmnlmnomnopnopq") = 84983e44 1c3bd26e baae4aa1 f95129e5 e54670f1

Алгоритм SHA–256

Шаг 3. *Определение констант.*

SHA–256 использует последовательность 64 32-битных слов $\left(K_0^{\{256\}}, K_1^{\{256\}}, \ldots, K_{63}^{\{256\}}\right)$. Эти слова представляют собой первые 32 бита дробной части кубических корней, взятых от первых 64 простых чисел. В 16-ричном виде эти константы выглядят так:

```
428a2f98 71374491 b5c0fbcf e9b5dba5 3956c25b 59f111f1 923f82a4 ab1c5ed5
d807aa98 12835b01 243185be 550c7dc3 72be5d74 80deb1fe 9bdc06a7 c19bf174
e49b69c1 efbe4786 0fc19dc6 240ca1cc 2de92c6f 4a7484aa 5cb0a9dc 76f988da

983e5152 a831c66d b00327c8 bf597fc7 c6e00bf3 d5a79147 06ca6351 14292967
27b70a85 2e1b2138 4d2c6dfc 53380d13 650a7354 766a0abb 81c2c92e 92722c85
a2bfe8a1 a81a664b c24b8b70 c76c51a3 d192e819 d6990624 f40e3585 106aa070
19a4c116 1e376c08 2748774c 34b0bcb5 391c0cb3 4ed8aa4a 5b9cca4f 682e6ff3
748f82ee 78a5636f 84c87814 8cc70208 90befffa a4506ceb bef9a3f7 c67178f2
```

Начальное значение хэш-функции устанавливается следующими константами:

$H_0^{(0)}$ = 6a09e667, $H_1^{(0)}$ = bb67ae85, $H_2^{(0)}$ = 3c6ef372, $H_3^{(0)}$ = a54ff53a,
$H_4^{(0)}$ = 510e527f, $H_5^{(0)}$ = 9b05688c, $H_6^{(0)}$ = 1f83d9ab, $H_7^{(0)}$ = 5be0cd19.

Шаг 4. *Основной цикл.*

В следующем цикле последовательно обрабатываются все блоки, на которые было разделено дополненное сообщение.

```
For i = 1 to N
  {
   // 1. Подготовка списка преобразованных слов сообщения
```

$$W_t = \begin{cases} M_t, & 0 \le t \le 15, \\ \sigma_1^{\{256\}}(W_{t-2}) + W_{t-7} + \sigma_0^{\{256\}}(W_{t-15}) + W_{t-16}, & 16 \le t \le 63. \end{cases}$$

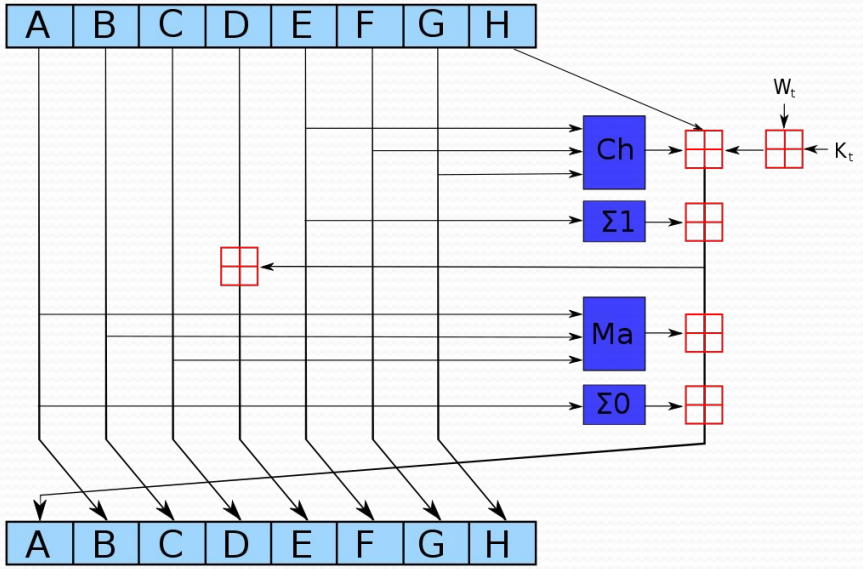
```
   // 2. Инициализация рабочих переменных
```

a = $H_0^{(i-1)}$ b = $H_1^{(i-1)}$ c = $H_2^{(i-1)}$ d = $H_3^{(i-1)}$

e = $H_4^{(i-1)}$ f = $H_5^{(i-1)}$ g = $H_6^{(i-1)}$ h = $H_7^{(i-1)}$

```
// 3. Внутренний цикл
For t = 0 to 63
  {
   T1 = h + Σ1{256}(e) + Ch(e,f,g) + Kt{256} + Wt
   T2 = Σ0{256}(a) + Maj(a,b,c)
   h = g
   g = f
   f = e
   e = d + T1
   d = c
   c = b
   b = a
   a = T1 + T2
  }
```

- $\Sigma_0^{\{256\}}(x) = ROTR^2(x) \oplus ROTR^{13}(x) \oplus ROTR^{22}(x);$
- $\Sigma_1^{\{256\}}(x) = ROTR^6(x) \oplus ROTR^{11}(x) \oplus ROTR^{25}(x);$



// 4. Вычисление промежуточного значения хэш-функции

$H_0^{(i)} = a + H_0^{(i-1)}$ $H_1^{(i)} = b + H_1^{(i-1)}$ $H_2^{(i)} = c + H_2^{(i-1)}$ $H_3^{(i)} = d + H_3^{(i-1)}$

$H_4^{(i)} = e + H_4^{(i-1)}$ $H_5^{(i)} = f + H_5^{(i-1)}$ $H_6^{(i)} = g + H_6^{(i-1)}$ $H_7^{(i)} = h + H_7^{(i-1)}$

}

Примеры значений хэш-функции (в 16-ричном виде):

SHA256 ("") = e3b0c442 98fc1c14 9afbf4c8 996fb924 27ae41e4 649b934c a495991b 7852b855

SHA256 ("abc") = ba7816bf 8f01cfea 414140de 5dae2223 b00361a3 96177a9c b410ff61 f20015ad

SHA256 ("abcdbcdecdefdefgefghfghighijhijkijkljklmklmnlmnomnopnopq") = 248d6a61 d20638b8 e5c02693 0c3e6039 a33ce459 64ff2167 f6ecedd4 19db06c1

MD4

MD4 - это однонаправленная хэш-функция, изобретенная Роном Ривестом. MD обозначает Message Digest (краткое изложение сообщения), алгоритм для входного сообщения выдает 128-битовое хэш-значение, или краткое изложение сообщения.

Ривест описал цели, преследуемые им при разработке алгоритма:

Безопасность. Вычислительно невозможно найти два сообщения с одинаковым хэш-значением. Вскрытие грубой силой является самым эффективным.

Прямая безопасность. Безопасность MD4 не основывается на каких-либо допущениях, например, предположении о трудности разложения на множители.

Скорость. MD4 подходит для высокоскоростных программных реализаций. Она основана на простом наборе битовых манипуляций с 32-битовыми операндами.

Простота и компактность. MD4 проста, насколько это возможна, и не содержит больших структур данных или сложных программных модулей.

Удачна архитектура. MD4 оптимизирована для микропроцессорной архитектуры (особенно для микропроцессоров Intel), для более крупных и быстрых компьютеров можно выполнить любые необходимые изменения.

После первого появления алгоритма Берт ден Боер и Антон Босселаерс (Antoon Bosselaers) достигли успеха при криптоанализе последних двух из трех этапов алгоритма. Ральфу Мерклу совершенно независимо удалось вскрыть первые два этапа. Хотя все эти вскрытия не были распространены на полный алгоритм, Ривест усилил свою разработку. В результате появилась MD5.

MD5

англ. Message Digest 5 – 128-битный алгоритм хеширования, разработанный профессором Рональдом Л. Ривестом из Массачусетского технологического института (Massachusetts Institute of Technology, MIT) в 1991 г. Предназначен для создания «отпечатков» или «дайджестов» сообщений произвольной длины. Является улучшенной в плане безопасности версией MD4. Используется для проверки подлинности опубликованных сообщений посредством сравнения дайджеста сообщения с опубликованным хэшем. Эту операцию называют «проверкой хэша». Алгоритм MD5 разработан таким образом, чтобы быть достаточно быстрым для выполнения на 32-разрядном процессоре. Алгоритм не требует больших таблиц подстановок и может быть закодирован весьма компактно

Алгоритм вычисления хеша.

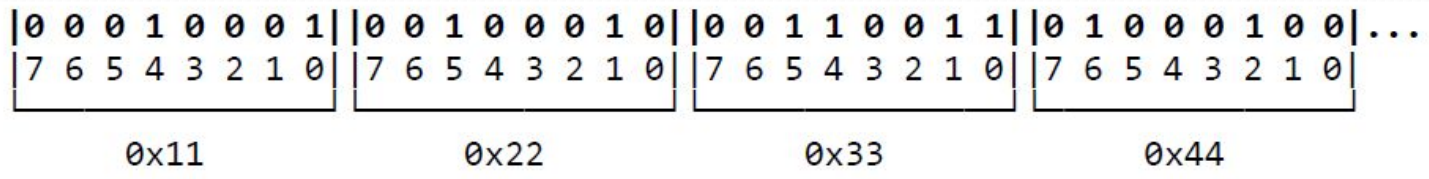
1. Выравнивание потока.

В конец исходного сообщения, длиной L, дописывают единичный бит, затем необходимое число нулевых бит так, чтобы новый размер L' был сравним с 448 по модулю 512 (L’ mod 512 = 448).

2. Добавление длины сообщения.

К модифицированному сообщению дописывают 64-битное представление длины данных (количество бит в сообщении). Т.е. длина сообщения T становится кратной 512 (T mod 512 = 0). Если длина исходного сообщения превосходит 2^{64} - 1, то дописывают только младшие 64 бита. Кроме этого, для указанного 64-битного представления длины вначале записываются младшие 32 бита, а затем старшие 32 бита.

Последовательность байт может быть интерпретирована как последовательность 32-битных слов, где каждая последовательная группа из 4 байт представляет собой 1 слово. Внутри слова байты располагаются следующим образом: сначала идут наименее значимые байты, затем – наиболее. Такой порядок расположения бит (или байт) называется ***little-endian*** (***порядок от младшего к старшему***). Например, пусть есть последовательность бит (выделена полужирным шрифтом):



Тогда она может быть интерпретирована как 4 подряд расположенных байта (0x11, 0x22, 0x33, 0x44 – шестнадцатеричные числа) или как одно слово 0x44332211.

3. Инициализация буфера.

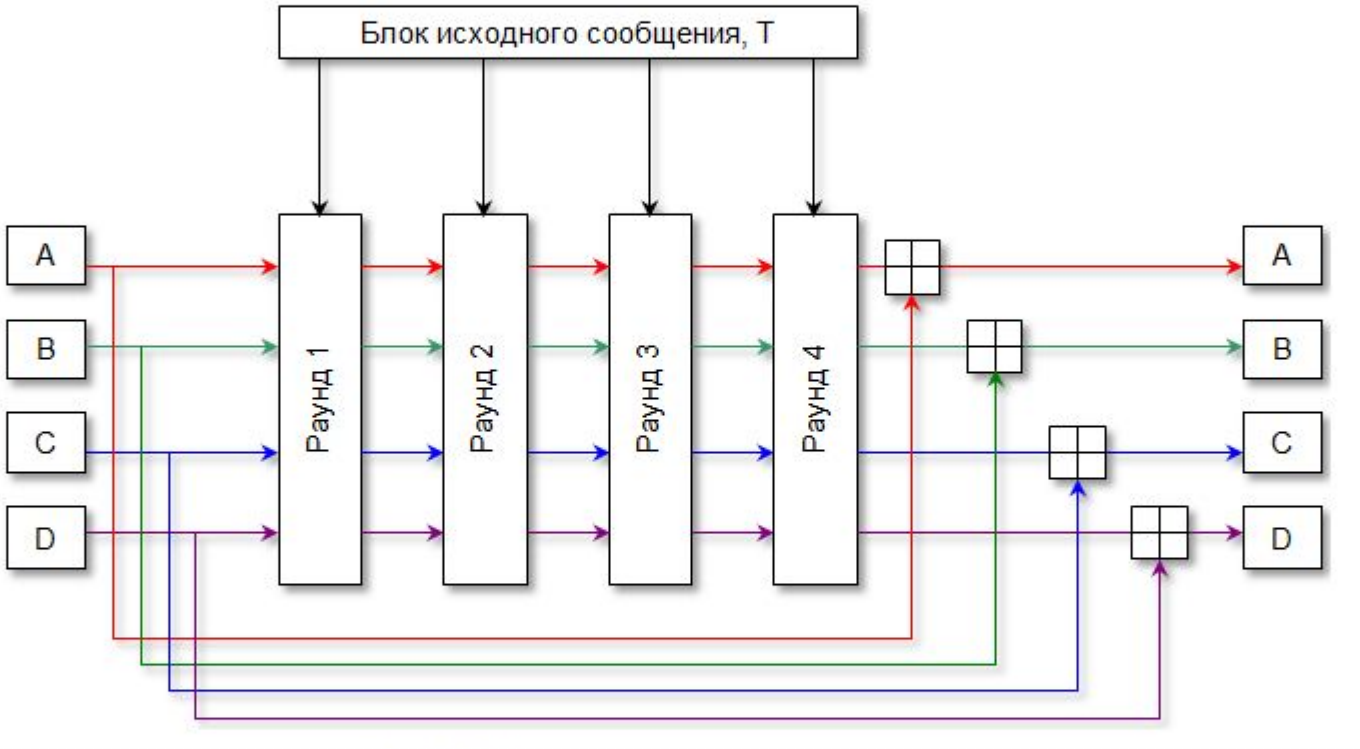
Для вычислений инициализируются 4 переменных размером по 32 бита и задаются начальные значения (шестнадцатеричное представление):

A = 67 45 23 01;
B = EF CD AB 89;
C = 98 BA DC FE;
D = 10 32 54 76.

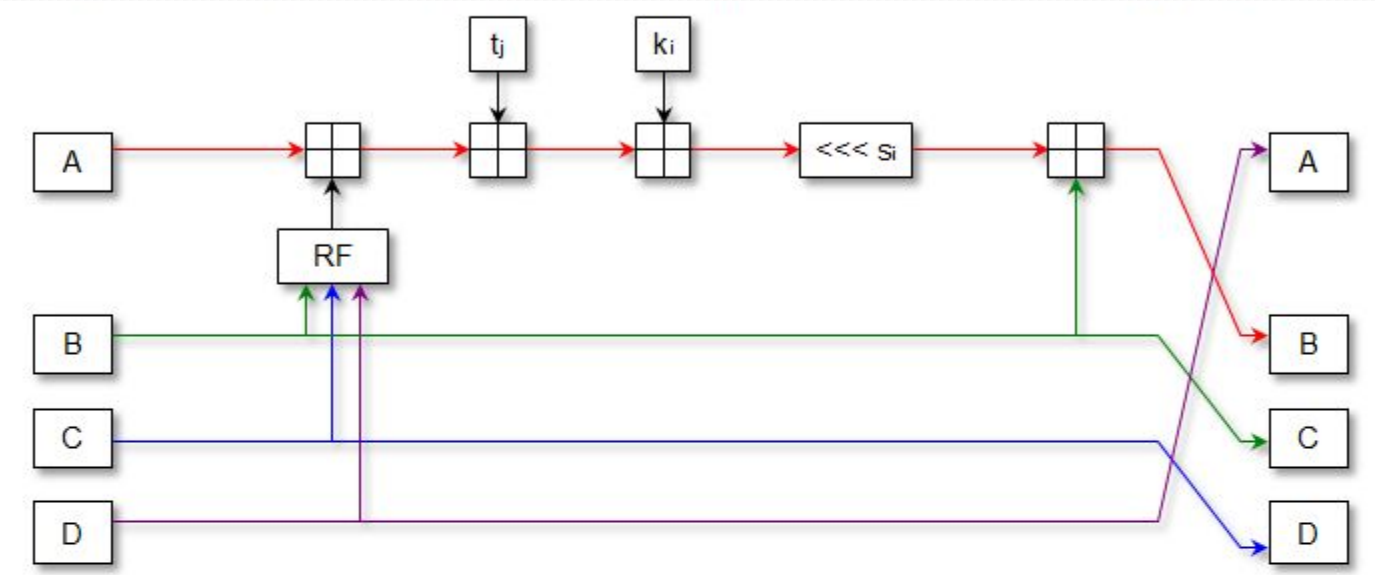
В этих переменных будут храниться результаты промежуточных вычислений. Начальное состояние ABCD называется инициализирующим вектором

4. Вычисление хеша в цикле.

Исходное сообщение разбивается на блоки T, длиной 512 бит. Для каждого блока в цикле выполняется процедура, приведенная на рис. Результат обработки всех блоков исходного сообщения в виде объединения 32-битных значений переменных ABCD и будет являться хешем.



Шаг основного цикла вычисления хеша



В каждом раунде над переменными ABCD и блоком исходного текста T в цикле (16 итераций) выполняются однотипные преобразования по следующей схеме.

Условные обозначения.
1) RF - раундовая функция, определяемая по следующей таблице.
2) tj - j-ая 32-битовая часть блока исходного сообщения T с обратным порядком следования байт;

3) ki - целая часть константы, определяемой по формуле
ki = 232 * | sin(i + 16 * (r - 1)) |,
где i – номер итерации цикла (i = 1..16);
r – номер раунда (r = 1..4).
Аргумент функции sin измеряется в радианах.
4) ⊞ – сложение по модулю 232.
5) <<< si – циклический сдвиг влево на si разрядов.
Используемая 32-битовая часть блока исходного сообщения tj и величина циклического сдвига влево si зависят от номера итерации и приведены в следующей таблице.

Раундовые функции RF

№ раунда	Обозначение функции	Формула расчета
1	F	$F(B, C, D) = (B \wedge C) \vee (\neg B \wedge D)$
2	G	$G(B, C, D) = (B \wedge D) \vee (\neg D \wedge C)$
3	H	$H(B, C, D) = B \oplus C \oplus D$
4	I	$I(B, C, D) = C \oplus (\neg D \vee B)$

Величины, используемые на шаге цикла раунда

№ итерации		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Раунд 1	t_j	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}	t_{13}	t_{14}	t_{15}	t_{16}
	s_i	7	12	17	22	7	12	17	22	7	12	17	22	7	12	17	22
Раунд 2	t_j	t_2	t_7	t_{12}	t_1	t_6	t_{11}	t_{16}	t_5	t_{10}	t_{15}	t_4	t_9	t_{14}	t_3	t_8	t_{13}
	s_i	5	9	14	20	5	9	14	20	5	9	14	20	5	9	14	20
Раунд 3	t_j	t_6	t_9	t_{12}	t_{15}	t_2	t_5	t_8	t_{11}	t_{14}	t_1	t_4	t_7	t_{10}	t_{13}	t_{16}	t_3
	s_i	4	11	16	23	4	11	16	23	4	11	16	23	4	11	16	23
Раунд 4	t_j	t_1	t_8	t_{15}	t_6	t_{13}	t_4	t_{11}	t_2	t_9	t_{16}	t_7	t_{14}	t_5	t_{12}	t_3	t_{10}
	s_i	6	10	15	21	6	10	15	21	6	10	15	21	6	10	15	21

После 4 раундов новое (модифицированное) значение каждой из переменных ABCD складывается (⊞) с исходным (значением переменной до 1-го раунда).

Результатом вычисления хэш-функции являются биты слов A, B, C, D, т.е. биты результата начинаются с младшего байта слова A и заканчиваются старшим битом слова D.

Примеры значений хэш-функции (в 16-ричном виде):
MD5 ("") = d41d8cd9 8f00b204 e9800998 ecf8427e
MD5 ("abc") = 90015098 3cd24fb0 d6963f7d 28e17f72
MD5 ("12345678901234567890123456789012345678901234567890123456789012345678901234567890") = 57edf4a2 2be3c955 ac49da2e 2107b67a