ГОСТ Р 34.11 – 94. Функция хэширования. Краткий анализ

Шефановский Д.Б.[1]

Май 2001

Обозначения

$\{01\}^*$ - множество двоичных строк произвольной конечной длины;

$\{01\}^n$ - множество двоичных строк длиной n бит;

$\{0\}^n$ - двоичная строка из n нулей;

$|A|$ - длина слова A в битах;

$A \oplus B$ - побитное сложение слов A и B по $\bmod 2$, или попросту *XOR*;

$A \boxplus B$ - сложение слов A и B по $\bmod 2^{256}$;

$\leftarrow$ оператор присвоения;

|| - конкатенация;

$GF(2)$ - поле Галуа характеристики 2.

Введение

Криптографические хэш функции играют фундаментальную роль в современной криптографии. Говоря в общем хэш функция h отображает двоичные строки произвольной конечной длины в выходы небольшой (например, 64, 128, 160,192, 224, 256, 384, 512) фиксированной длины называемые *хэш величинами* за полиномиальное время:

$$h: \{01\}^* \to \{01\}^n,$$

где $\{01\}^* \in \bigcup_{i \in \mathbb{N}} \{01\}^i$ (в ГОСТ Р 34.11 – 94 $n = 256$). Основная идея криптографических функций состоит в том, чтобы хэш величины служили как компактный репрезентативный образ входной двоичной строки, и их можно использовать как если бы они однозначно отождествлялись с этой строкой, хотя для области определения D и области значений R с $|D| \gg |R|$ (имеются ввиду мощности множеств), функция типа “множество – один” подразумевает, что существование *столкновений* (пары входов с одинаковым выходом) неизбежно. Область применения хэш функции четко не оговорена: используется “для реализации процедур электронной цифровой подписи (ЭЦП), при передаче, обработке и хранении информации в автоматизированных системах”.

Сообщения с произвольной длины можно сжать используя хэш функцию с фиксированным размером входа при помощи двух методов:

- последовательного (итерационного);

[1] Учебный центр «ИНФОРМЗАЩИТА» 127018, г. Москва, ул. Образцова, 38 а/я 55
(095) 289-4232, 289-8998, 289-3162, 219-3188
shefanovski@infosec.ru

– параллельного.

Создатели ГОСТ Р 34.11 – 94 пошли по первому пути и использовали метод последовательного хэширования использующий хэш функцию с фиксированным размером входа $h:\{01\}^{2n}\to\{01\}^{n}$ (см. Рис. 1), то есть функцию сжатия с коэффициентом 2.

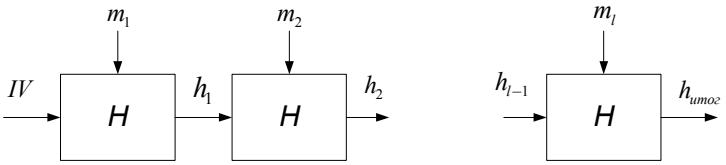

Рис. 1 Метод последовательного хэширования

Если необходимо хэшировать сообщение $m=(m_1,m_2,\ldots,m_l)$, то хэширование выполняется следующим образом:

$$h_0 \leftarrow IV,$$
$$h_i \leftarrow H(m_i,h_{i-1}), \quad для\, i=1,2,\ldots,l$$
$$h_{итог} \leftarrow h_l.$$

Здесь H_i - функция сжатия, а h_i - переменная сцепления.

Если последний блок меньше чем n бит, то он набивается одним из существующих методов до достижения длины кратной n. В отличие от стандартных предпосылок, что сообщение разбито на блоки и произведена набивка последнего блока (форматирование входа априори), если необходимо, до начала хэширования, то в ГОСТ Р 34.11 – 94 процедура хэширования ожидает конца сообщения (форматирование входного сообщения постериори). Набивка производится следующим образом: последний блок сдвигается вправо, а затем набивается нулями до достижения длины в 256 бит.

На первый взгляд алгоритм хэширования по ГОСТ Р 34.11 – 94 можно классифицировать как устойчивый к столкновениям ($n=256$, следовательно атака по парадоксу дней рождения потребует приблизительно $2^{256/2}$ операций хэширования) код выявляющий модификации (*Collision Resistant Hash Function, CRHF*), см. Замечание 4. Нельзя забывать, что хэш функцию по ГОСТ Р 34.11 – 94 можно легко преобразовать в код аутентификации сообщения любым известным методом (например, HMAC [1], методом секретного префикса, суффикса, оболочки и т.д.). Однако конструкторы предусмотрели дополнительные меры защиты: параллельно рассчитываются контрольная сумма представляющая собой сумму всех блоков сообщения (последний суммируется уже набитым) по правилу $A+B \bmod 2^k$, где $k=|A|=|B|$, а $|A|$ и $|B|$ битовые длины слов A и B (далее на рисунках и в тексте эту операцию будем обозначать значком $\otimes$), и битовая длина хэшируемого сообщения приводимая по $\bmod 2^{256}$ (MD - усиление), которые в финальной функции сжатия используются для вычисления итогового хэша (см. Рис. 2).

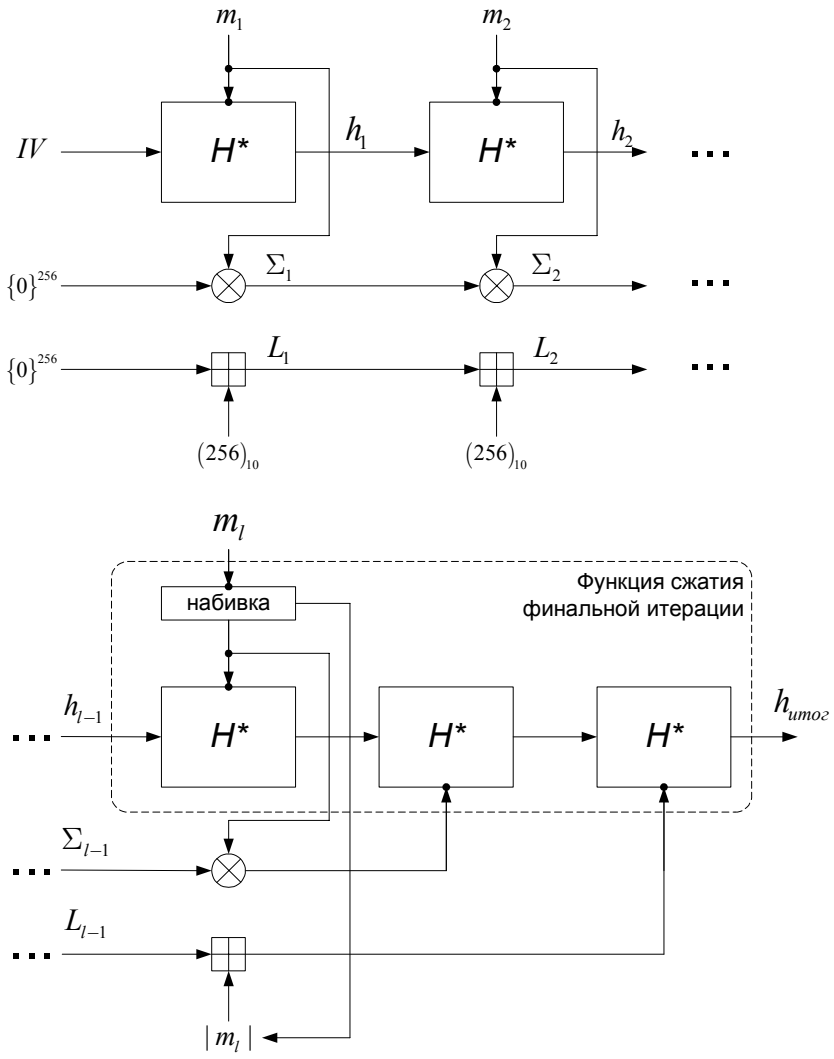

Рис. 2 Общая схема функции хэширования по ГОСТ Р 34.11 – 94

Замечание 1 (набивка) Указывать в передаваемом сообщении сколько было добавлено нулей к последнему блоку не требуется, так как длина сообщения участвует в хэшировании.

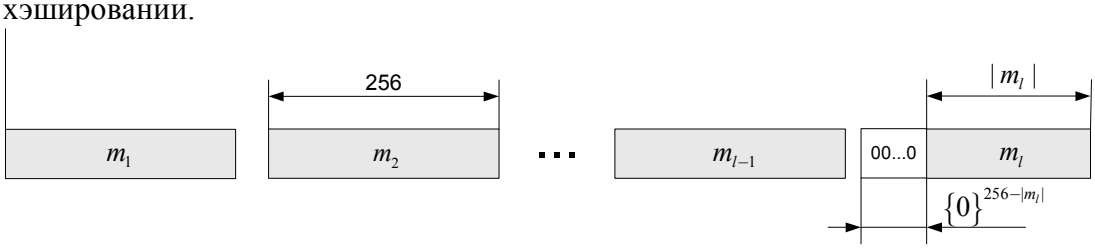

Рис. 3 Набивка сообщения

Замечание 2 (начальный вектор IV) Согласно ГОСТ Р 34.11 – 94 IV - произвольное фиксированное слово длиной 256 бит ($IV \in \{01\}^{256}$). В таком случае, если он априорно не известен верифицирующему целостность сообщения, то он должен передаваться вместе с сообщением с гарантией целостности. При небольших сообщениях можно усложнить задачу противнику, если IV выбирается из небольшого множества допустимых величин (но при этом увеличивается вероятность угадывания хэш величины противником). Также он может задаваться в рамках организации, домена как константа (обычно как в тестовом примере).

Детальное рассмотрение

1. ***Функция сжатия внутренних итераций (по ГОСТ "шаговая функция хэширования")***

Функция сжатия внутренних итераций χ отображает два слова длиной 256 бит в одно слово длиной 256 бит:

$$\chi : \{01\}^{256} \times \{01\}^{256} \to \{01\}^{256},$$

и состоит из трех процедур (см. Рис. 4):

- Перемешивающего преобразования (по существу модифицированной схемы Фейстеля),
- Шифрующего преобразования;
- Генерирования ключей.

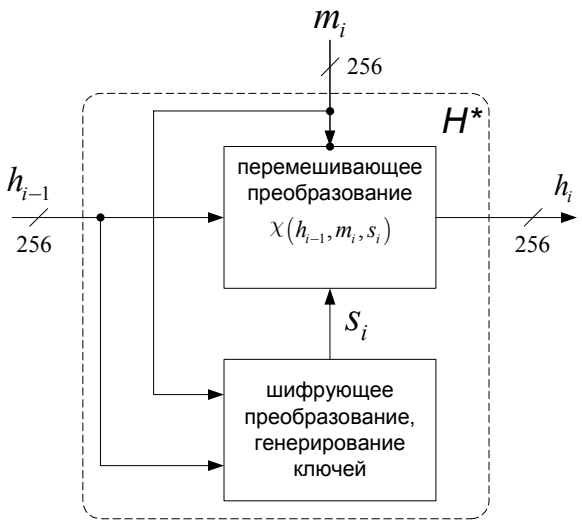

Рис. 4 Структура функции сжатия в ГОСТ Р 34.11 – 94

а) Перемешивающее преобразование

Здесь использована модифицированная для хэширования перестановка Фейстеля, где n-битный вход разделяется на k блоков по r бит, так чтобы $kl = n$:

$$F_p : \underbrace{\{01\}^r \times \cdots \times \{01\}^r}_{l} \to \underbrace{\{01\}^r \times \cdots \times \{01\}^r}_{l},$$

где p - входное сообщение. Пусть вход $A = (A_1, \ldots, A_l) \in \{01\}^n$, а выход $B = (B_1, \ldots, B_l) \in \{01\}^n$, тогда $F_p(A)$ описывается следующим образом:

$$\begin{cases} B_l \leftarrow A_1 \oplus f_p(A_2, \ldots, A_l), \\ B_{l-1} \leftarrow A_l, \ldots, B_1 \leftarrow A_2. \end{cases}$$

Схематически модифицированная функция Фейстеля представлена на Рис. 5 (без претензий на общность).

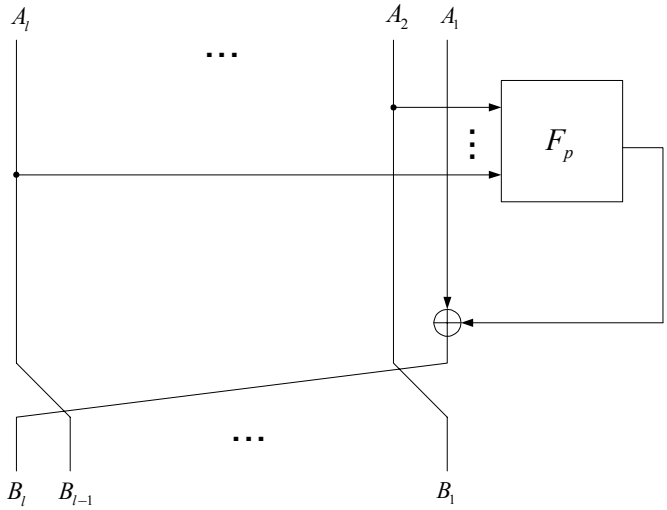

Рис. 5 Модифицированная схема Фейстеля для хэширования

Перемешивающее преобразование имеет вид

$$h_i = \chi\left(m_i, h_{i-1}, s_i\right) = \psi^{61}\left(h_{i-1} \oplus \psi\left(m_i \oplus \psi^{12}\left(s_i\right)\right)\right),$$

где ψ^j - j-я степень преобразования ψ. Схематически данное преобразование представлено на Рис. 6.

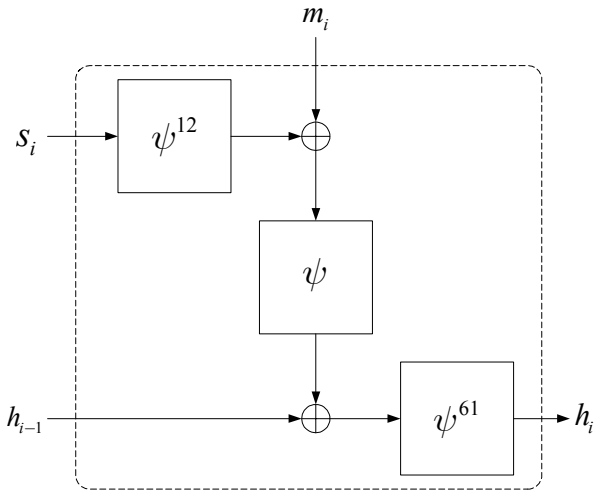

Рис. 6 Перемешивающее преобразование ГОСТ Р 34.11 – 94

Заметим (см. Рис. 6), что s_i выводится из h_{i-1} (см. Рис. 4). Функция $\psi : \{01\}^{256} \to \{01\}^{256}$ преобразует слово $\eta_{16} \| \cdots \| \eta_1$, $\eta_i \in \{01\}^{16}, i = \overline{1,16}$ в слово $\eta_1 \oplus \eta_2 \oplus \eta_3 \oplus \eta_4 \oplus \eta_{13} \oplus \eta_{16} \| \eta_{16} \| \eta_{15} \| \cdots \| \eta_2$ (см. Рис. 7).

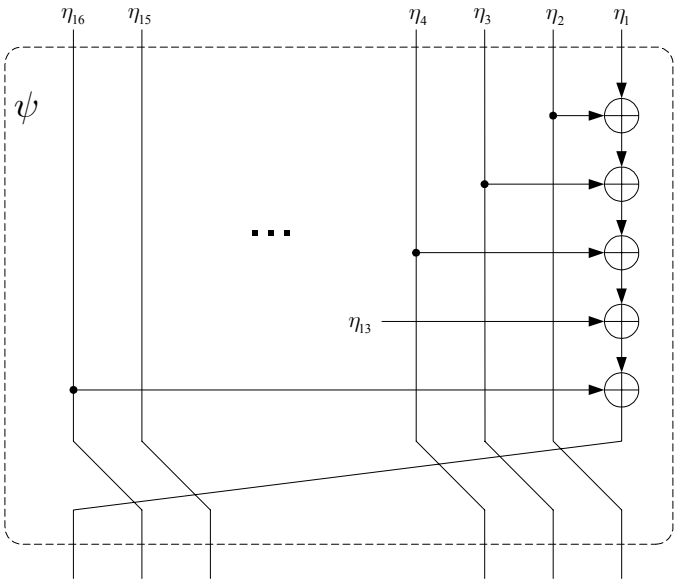

Рис. 7 Функция ψ перемешивающего преобразования

Замечание 3 Функция ψ перемешивающего преобразования не удовлетворяет следующим локальным критериям:

- 0-1 сбалансированности (гарантирует, что выходы функции будут 0 или 1 с вероятностью 0,5 когда вход функции выбирается случайно и равновероятно над всеми возможными векторами);
- большой нелинейности (Функция f называется *линейной функцией*, если f имеет вид $f(x_n, x_{n-1}, \dots, x_1) = a_n x_n \oplus a_{n-1} x_{n-1} \oplus \dots \oplus a_1 x_1 \oplus a_0$, где $a_i \in GF(2)$; Если обратить внимание на вид функции ψ, то очевидно, что это линейная функция);
- строгому лавинному критерию (Strict Avalanche Criterion, SAC) (говорят, что f удовлетворяет строгому лавинному критерию, если для каждого $1 \le i \le n$, дополнение x_i даст в результате выход f являющийся дополнением для 50% всех возможных входных векторов).

Кроме этого в перемешивающем преобразовании применяется только одна функция ψ, а следовательно оно не обладает следующими общими свойствами:

- линейной неэквивалентности структуры (две функции f и g *линейно эквивалентны по структуре*, если f можно преобразовать в g линейным преобразованием координат и дополнениями функций, т.е имеется неособенная $n \times n$ матрица A над $GF(2)$ а также вектор $B \in V_n$ такой что $f(xA \oplus B) = g(x)$, или $f(xA \oplus B) \oplus 1 = g(x)$, где $x = (x_n, x_{n-1}, \dots, x_1)$; этот критерий гарантирует, что функции используемые криптографическим алгоритмом не обладают схожестью структуры);
- взаимной некоррелированности выходов (функции f и *взаимно некоррелированы по выходу*, если f, g и $f \oplus g$ - 0-1 сбалансированные нелинейные функции; гарантирует, что последовательности функций не будут взаимно коррелированны или через линейные функции, или через смещение в выходных битах).

Остается вопрос, из каких критериев исходили конструктора?

б) Подпрограмма генерирования ключей

Данная подпрограмма использует следующие функции и константы:

- Константы

$$C_2, C_4 = \{0\}^{256},\ C_3 = 0xf0ff000ff00f0ff00f0f0f0ff0f0f0f0.$$

- Функции
  - $P: \{01\}^{256} \to \{01\}^{256}$. Пусть $X = \xi_{32} \| \xi_{31} \| \cdots \| \xi_1$, тогда
    $P(X) = \xi_{\varphi(32)} \| \xi_{\varphi(31)} \| \cdots \| \xi_{\varphi(1)}$, где $\varphi(i+1+4(k-1)) = 8i+k,\ i = \overline{0,3},\ k = \overline{1,8}.$
  - $A: \{01\}^{256} \to \{01\}^{256}$. Пусть $X = x_4 \| x_3 \| x_2 \| x_1$, тогда
    $A(X) = (x_1 \oplus x_2) \| x_4 \| x_3 \| x_2.$

Алгоритм 1 *Подпрограмма генерирования ключей*

---

1. $U \leftarrow m_i,\ V \leftarrow h_{i-1},\ W \leftarrow U \oplus V,\ K_1 \leftarrow P(W)$
2. Для j от 2 от 4 выполняем следующее:
   2.1. $U \leftarrow A(U) \oplus C_j,\ V \leftarrow A(A(V)),\ W \leftarrow U \oplus V,\ K_j \leftarrow P(W)$
3. Выход (K_1, K_2, K_3, K_4)

в) Шифрующее преобразование

Основным функциональным предназначением является получение s_i из h_{i-1}. Пусть $h_{i-1} = h_{i-1}^4 \| h_{i-1}^3 \| h_{i-1}^2 \| h_{i-1}^1$, где $h_{i-1}^j \in \{01\}^{64},\ j = \overline{1,4}$, а $s_i = s_i^4 \| s_i^3 \| s_i^2 \| s_i^1$, где $s_i^j \in \{01\}^{64},\ j = \overline{1,4}$. Тогда

$$s_i^j \leftarrow E_{K_j}\left(h_{i-1}^j\right),$$

где $j = \overline{1,4}$, E_K - шифрование по ГОСТ 28147 – 89 в режиме простой замены. Схематически это изображено на Рис. 8.

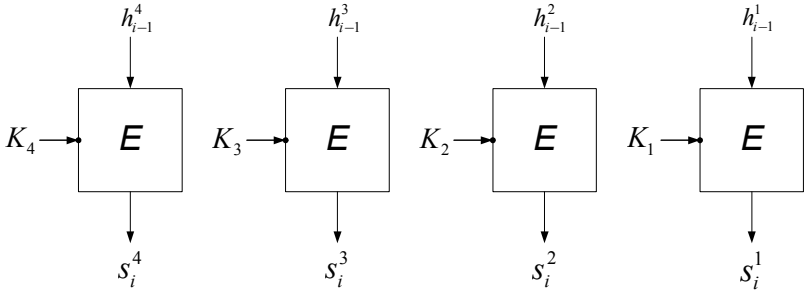

Рис. 8 Шифрующее преобразование

Замечание 4 Как видно $s_i = s_i^4 \| s_i^3 \| s_i^2 \| s_i^1$ - конкатенация 64 битных слов, полученных в результате шифрования. Следовательно, в свете замечания 3, s_i можно было бы атаковать по частям, но это предотвращается казуальным генерированием ключей (в данном случае m_i и h_{i-1} выступают в роли ключа для выведения ключей).

Замечание 5 (S-блоки при шифровании в режиме простой замены) Согласно стандарта ГОСТ 28147 – 89 S-блоки не определены, но указано, что они должны назначаться вышестоящим органом. Следовательно они должны учитываться как данные инициализации функции хэширования, и в определенной степени влияют на ее свойства.

*2. **Функция сжатия финальной итерации***

Функция сжатия финальной итерации производит итоговую хэш величину, зависящую от результата хэширования последовательным методом, контрольной суммы по $\bmod 2$ и длины сообщения в битах приведенной по $\bmod 2^{256}$:

$$m'_l \times h_{l-1} \times \Sigma_l \times L_l \to h_{итог},$$

где $m'_l, h_{l-1}, \Sigma_l, L_l \in \{01\}^{256}$, m'_l - последний набитый нулями блок (если необходимо, см. Рис. 3). Сначала вычисляется битовая длина последнего (ненабитого) блока сообщения $|m_l| \le 256$ бит. Если $|m_l| < 256$, то производится его набивка справа нулями до достижения длины 256 бит и получается новый блок $m'_l \leftarrow \{0\}^{256-|m_l|} \| m_l$. Вычисляются итоговая контрольная сумма $\Sigma_l \leftarrow \Sigma_{l-1} \otimes m'_l$ и длина всего сообщения $L_l \leftarrow L_{l-1} + |m'| \left(\bmod 2^{256}\right)$. Параллельно выполняется последняя итерация $h_l \leftarrow \chi(m', h_{l-1})$. Затем вычисляются перемешивающие преобразования $h_{l+1} \leftarrow \chi(L_l, h_l)$ и $h_{итог} \leftarrow \chi(\Sigma_l, h_{l+1})$, давая в результате итоговую хэш величину $h_{итог}$.

Теперь дадим формальное описание алгоритма:

***Алгоритм 2** Вычисление хэш функции по* ГОСТ Р 34.11 – 94

Вход: двоичное сообщение M, блоки замен для шифрования в режиме простой замены по ГОСТ 28147-89, начальный вектор $IV \in \{01\}^{256}$

Выход: хэш величина $h_{итог}$ для сообщения M.

1. ***Инициализация алгоритма*** $h \leftarrow IV, \Sigma \leftarrow \{0\}^{256}, L \leftarrow \{0\}^{256}$.

2. ***Функция сжатия внутренних итераций*** Пока $|M| > 256$ выполняем следующее:

   2.1. $h \leftarrow \chi(m_s, h)$ (итерация метода последовательного хэширования),

   2.2. $L \leftarrow L + 256 \left(\bmod 2^{256}\right)$ (итерация вычисления длины сообщения),

   2.3. $\Sigma \leftarrow \Sigma \otimes m_s$ (итерация вычисления контрольной суммы).

3. Иначе (***функция сжатия финальной итерации***)

   3.1. $L \leftarrow L + |m| \left(\bmod 2^{256}\right)$ (вычисление полной длины сообщения),

   3.2. $m' \leftarrow \{0\}^{256-|m|} \| m$ (набивка последнего блока),

   3.3. $\Sigma \leftarrow \Sigma \otimes m'$ (вычисление контрольной суммы сообщения),

   3.4. $h \leftarrow \chi(m', h)$,

   3.5. $h \leftarrow \chi(L, h)$ (MD - усиление),

   3.6. $h_{итог} \leftarrow \chi(\Sigma, h)$.

4. ***Выход*** ($h_{итог}$)

Заключение Из приведенного выше описания хэш функции по ГОСТ Р 34.11 – 94 можно сделать следующие выводы:

- Булева функция перемешивающего преобразования линейна и ее применение неоправданно – это приводит к снижению скорости обработки данных из-за большого числа повторений перемешивающего преобразования для достижения “заданного” уровня безопасности (в первую очередь SAC);
- Шифрующее преобразование, при определенных допущениях, невозможно атаковать по частям, а, следовательно, функцию сжатия можно считать стойкой к столкновениям (CRHF);
- Алгоритм хэширования является методом последовательного хэширования с MD – усилением (коэффициент сжатия 2);
- Используется постериорное форматирование сообщения;
- Стойкость хэш функции в известной мере зависит от выбора блоков замен в шифрующем преобразовании, к тому же они один из параметров инициализации алгоритма хэширования;
- IV в стандарте не фиксирован, а это подразумевает, что необходимо выработать правила его использования (см. Замечание 2), к тому же имеется большой класс атак на псевдостолкновения при не фиксированном IV [3];
- скорость обработки данных хэш функцией значительно меньше чем у аналогичных по внешним параметрам HAVAL, SHA-256, а тем более остального MD-семейства, из-за попыток ликвидировать очевидные оплошности конструирования усложнением функции сжатия;
- приблизительная скорость реализации 4/5 от скорости реализации лежащего в основе алгоритма шифрования;
- зависимость получаемой хэш величины от длины исходного сообщения исключает возможность проведения атаки на длинные сообщения для второго прообраза;
- нет обоснования выбора конструкции, функций, констант.

1. ГОСТ Р 34.11 – 94. "Информационная технология. Криптографическая защита информации. Функция хэширования".
2. *M. Bellare, R. Canetti, and H. Krawczyk.* **Keying hash functions for message authentication.** *Advances in Cryptology - Crypto 96 Proceedings, Lecture Notes in Computer Science Vol. 1109, N. Koblitz ed,* Springer-Verlag, 1996.
3. *Alfred J. Menezes, Paul C. van Oorschot and Scott A. Vanstone* **Handbook of Applied Cryptography** *CRC Press ISBN: 0-8493-8523-7 October 1997, 816 pages*