

ИНФОРМАЦИОННАЯ ТЕХНОЛОГИЯ  
КРИПТОГРАФИЧЕСКАЯ ЗАЩИТА  
ИНФОРМАЦИИ  
ФУНКЦИЯ ХЭШИРОВАНИЯ

Издание официальное

БЗ 3—94/131

ГОССТАНДАРТ РОССИИ  
Москва

## Предисловие

**1 РАЗРАБОТАН** Главным управлением безопасности связи Федерального агентства правительственной связи и информации и Всероссийским научно-исследовательским институтом стандартизации

**ВНЕСЕН** Техническим комитетом по стандартизации ТК 22 «Информационная технология» и Федеральным агентством правительственной связи и информации

**2 ПРИНЯТ И ВВЕДЕН В ДЕЙСТВИЕ** Постановлением Госстандарта России от 23.05.94 № 154

**3 ВВЕДЕН ВПЕРВЫЕ**

© Издательство стандартов, 1994

Настоящий стандарт не может быть полностью или частично воспроизведен, тиражирован и распространен в качестве официального издания без разрешения Госстандарта России

## СОДЕРЖАНИЕ

1 Область применения . . . . .	1
2 Нормативные ссылки . . . . .	1
3 Обозначения . . . . .	1
4 Общие положения . . . . .	2
5 Шаговая функция хэширования . . . . .	3
6 Процедура вычисления хэш-функции . . . . .	4
Приложение А Проверочные примеры . . . . .	6

## ВВЕДЕНИЕ

Расширяющееся применение информационных технологий при создании, обработке, передаче и хранении документов требует в определенных случаях сохранения конфиденциальности их содержания, обеспечения полноты и достоверности.

Одним из эффективных направлений защиты информации является криптография (криптографическая защита), широко применяемая в различных сферах деятельности в государственных и коммерческих структурах.

Криптографические методы защиты информации являются объектом серьезных научных исследований и стандартизации на национальных, региональных и международных уровнях.

Настоящий стандарт определяет процедуру вычисления хэш-функции для любой последовательности двоичных символов.

Функция хеширования заключается в сопоставлении произвольного набора данных в виде последовательности двоичных символов и его образа фиксированной небольшой длины, что позволяет использовать эту функцию в процедурах электронной цифровой подписи для сокращения времени подписывания и проверки подписи. Эффект сокращения времени достигается за счет вычисления подписи только под образом подписываемого набора данных.

**ГОСУДАРСТВЕННЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ**

Информационная технология

**КРИПТОГРАФИЧЕСКАЯ ЗАЩИТА ИНФОРМАЦИИ**

Функция хэширования

Information technology.  
Cryptographic Data Security.  
Hashing functionДата введения 1995—01—01**1 ОБЛАСТЬ ПРИМЕНЕНИЯ**

Настоящий стандарт определяет алгоритм и процедуру вычисления хэш-функции для любой последовательности двоичных символов, которые применяются в криптографических методах обработки и защиты информации, в том числе для реализации процедур электронной цифровой подписи (ЭЦП) при передаче, обработке и хранении информации в автоматизированных системах.

Определенная в настоящем стандарте функция хэширования используется при реализации систем электронной цифровой подписи на базе асимметричного криптографического алгоритма по ГОСТ Р 34.10.

**2 НОРМАТИВНЫЕ ССЫЛКИ**

В настоящем стандарте использованы ссылки на следующие стандарты:

ГОСТ 28147—89 Системы обработки информации. Защита криптографическая. Алгоритмы криптографического преобразования.

ГОСТ Р 34.10—94 Информационная технология. Криптографическая защита информации. Процедуры выработки и проверки электронной цифровой подписи на базе асимметричного криптографического алгоритма.

**3 ОБОЗНАЧЕНИЯ**

В настоящем стандарте используются следующие обозначения:

$V$  — множество всех конечных слов в алфавите  $V = \{0,1\}$ . Чтение слов и нумерация знаков алфавита (символов) осуществляются справа налево (номер правого символа в слове равен единице, второго справа — двум и т. д.).

$|A|$  — длина слова  $A \in V^*$ .

$V_k(2)$  — множество всех бинарных слов длины  $k$ .

$A||B$  — конкатенация слов  $A, B \in V^*$  — слово длины  $|A|+|B|$ , в котором левые  $|A|$  символов образуют слово  $A$ , а правые  $|B|$  символов образуют слово  $B$ . Можно также использовать обозначение  $A||B = AB$ .

$A^k$  — конкатенация  $k$  экземпляров слова  $A$  ( $A \in V^*$ ).

$\leq N >_k$  — слово длины  $k$ , содержащее двоичную запись вычета  $N \pmod{2^k}$  неотрицательного целого числа  $N$ .

$\hat{A}$  — неотрицательное целое число, имеющее двоичную запись  $A$  ( $A \in V^*$ ).

$\oplus$  — побитовое сложение слов одинаковой длины по модулю 2.

$\oplus'$  — сложение по правилу  $A \oplus' B = \langle \hat{A} + \hat{B} \rangle_k$ , ( $k = |A| = |B|$ ).

$M$  — последовательность двоичных символов, подлежащая хэшированию (сообщение в системах ЭЦП),  $M \in V^*$ .

$h$  — хэш-функция, отображающая последовательность  $M \in V^*$  в слово  $h(M) \in V_{256}(2)$ .

$E_K(A)$  — результат зашифрования слова  $A$  на ключе  $K$  с использованием алгоритма шифрования по ГОСТ 28147 в режиме простой замены ( $K \in V_{256}(2)$ ,  $A \in V_{64}(2)$ ).

$H$  — стартовый вектор хэширования.

$e = g$  — присвоение параметру  $e$  значения  $g$ .

#### 4 ОБЩИЕ ПОЛОЖЕНИЯ

Под хэш-функцией  $h$  понимается зависящее от параметра стартового вектора хэширования  $H$ , являющегося словом из  $V_{56}(2)$  отображение

$$h: V^* \rightarrow V_{256}(2).$$

Для определения хэш-функции необходимы:

— алгоритм вычисления шаговой функции хэширования  $h$ , т. е. отображения

$$h: V_{256}(2) \times V_{256}(2) \rightarrow V_{256}(2);$$

— описание итеративной процедуры вычисления значения хэш-функции  $h$ .

## 5 ШАГОВАЯ ФУНКЦИЯ ХЭШИРОВАНИЯ

Алгоритм вычисления шаговой функции хэширования включает в себя три части, реализующие последовательно:

- генерацию ключей — слов длины 256 битов;
- шифрующее преобразование — зашифрование 64-битных подслов слова  $H$  на ключах  $K_i$  ( $i=1, 2, 3, 4$ ) с использованием алгоритма по ГОСТ 28147 в режиме простой замены;
- перемешивающее преобразование результата шифрования.

## 5.1 Генерация ключей.

Рассмотрим  $X = (b_{256}, b_{255}, \dots, b_1) \in V_{256}(2)$ .

$$\begin{aligned} \text{Пусть } X &= x_4 \| x_3 \| x_2 \| x_1 = \\ &= \eta_{16} \| \eta_{15} \| \dots \| \eta_1 = \\ &= \xi_{32} \| \xi_{31} \| \dots \| \xi_1, \end{aligned}$$

где  $x_i = (b_{i \times 64}, \dots, b_{(i-1) \times 64 + 1}) \in V_{64}(2)$ ,  $i = \overline{1, 4}$ ;  
 $\eta_j = (b_{j \times 16}, \dots, b_{(j-1) \times 16 + 1}) \in V_{16}(2)$ ,  $j = \overline{1, 16}$ ;  
 $\xi_k = (b_{k \times 8}, \dots, b_{(k-1) \times 8 + 1}) \in V_8(2)$ ,  $k = \overline{1, 32}$ .

Обозначают  $A(X) = (x_1 \oplus x_2) \| x_4 \| x_3 \| x_2$ .

Используют преобразование  $P: V_{256}(2) \rightarrow V_{256}(2)$

слова  $\xi_{32} \| \dots \| \xi_1$  в слово  $\xi_{\varphi(32)} \| \dots \| \xi_{\varphi(1)}$ ,

где  $\varphi(i+1+4(k-1)) = 8i+k$ ,  $i=0 \div 3$ ,  $k=1 \div 8$ .

Для генерации ключей необходимо использовать следующие исходные данные:

- слова  $H, M \in V_{256}(2)$ ;
- параметры: слова  $C_i$  ( $i=2, 3, 4$ ), имеющие значения  
 $C_2 = C_4 = 0^{256}$  и  $C_3 = 1^8 0^8 1^{16} 0^{24} 1^{16} 0^8 (0^8 1^8)^2 1^8 0^8 (0^8 1^8)^4 (1^8 0^8)^4$ .

При вычислении ключей реализуется следующий алгоритм:

## 1 Присвоить значения

$$i := 1, U := H, V := M.$$

## 2 Выполнить вычисление

$$W = U \oplus V, K_i = P(W).$$

3 Присвоить  $i := i + 1$ .4 Проверить условие  $i = 5$ .

При положительном исходе перейти к шагу 7. При отрицательном — перейти к шагу 5.

## 5 Выполнить вычисление

$$U := A(U) \oplus C_i, V := A(A(V)), W := U \oplus V, K_i = P(W).$$

6 Перейти к шагу 3.

7 Конец работы алгоритма.

## 5.2 Шифрующее преобразование

На данном этапе осуществляется зашифрование 64-битных подслов слова  $H$  на ключах  $K_i$  ( $i = 1, 2, 3, 4$ ).

Для шифрующего преобразования необходимо использовать следующие исходные данные:

$$H = h_4 \| h_3 \| h_2 \| h_1, \quad h_i \in V_{64}(2), \quad i = \overline{1, 4}$$

и набор ключей  $K_1, K_2, K_3, K_4$ .

Реализуют алгоритм зашифрования и получают слова

$$s_i = E_{K_i}(h_i), \quad \text{где } i = 1, 2, 3, 4.$$

В результате данного этапа образуется последовательность

$$S = s_4 \| s_3 \| s_2 \| s_1.$$

## 5.3 Перемешивающее преобразование

На данном этапе осуществляется перемешивание полученной последовательности с применением регистра сдвига.

Исходными данными являются:

слова  $H, M \in V_{256}(2)$  и слово  $S \in V_{256}(2)$ .

Пусть отображение

$$\psi: V_{256}(2) \rightarrow V_{256}(2)$$

преобразует слово

$$\eta_{16} \| \dots \| \eta_1, \quad \eta_i \in V_{16}(2), \quad i = \overline{1, 16}$$

в слово

$$\eta_1 \oplus \eta_2 \oplus \eta_3 \oplus \eta_4 \oplus \eta_{13} \oplus \eta_{16} \| \eta_{16} \| \dots \| \eta_2.$$

Тогда в качестве значения шаговой функции хеширования принимается слово

$$x(M, H) = \psi^{61}(H \oplus \psi(M \oplus \psi^{12}(S))),$$

где  $\psi^i$  —  $i$ -я степень преобразования  $\psi$ .

## 6 ПРОЦЕДУРА ВЫЧИСЛЕНИЯ ХЭШ-ФУНКЦИИ

Исходными данными для процедуры вычисления значения функции  $h$  является подлежащая хешированию последовательность  $M \in B^*$ . Параметром является стартовый вектор хеширования  $H$  — произвольное фиксированное слово из  $V_{256}(2)$ .



Процедура вычисления функции  $h$  на каждой итерации использует следующие величины:

$M \in B^*$  — часть последовательности  $M$ , не прошедшая процедуры хэширования на предыдущих итерациях;

$H \in V_{256}(2)$  — текущее значение хэш-функции;

$\Sigma \in V_{256}(2)$  — текущее значение контрольной суммы;

$L \in V_{256}(2)$  — текущее значение длины обработанной на предыдущих итерациях части последовательности  $M$ .

Алгоритм вычисления функции  $h$  включает в себя этапы:

#### Этап 1

Присвоить начальные значения текущих величин

1.1  $M := M$

1.2  $H := H$

1.3  $\Sigma := 0^{256}$

1.4  $L := 0^{256}$

1.5 Переход к этапу 2

#### Этап 2

2.1 Проверить условие  $|M| > 256$ .

При положительном исходе перейти к этапу 3.

В противном случае выполнить последовательность вычислений:

2.2  $L := \langle L + |M| \rangle_{256}$

2.3  $M' := 0^{256 - |M|} \| M$

2.4  $\Sigma := \Sigma \oplus M'$

2.5  $H := x(M', H)$

2.6  $H := x(L, H)$

2.7  $H := x(\Sigma, H)$

2.8 Конец работы алгоритма

#### Этап 3

3.1 Вычислить подслово  $M_s \in V_{256}(2)$  слова  $M$  ( $M = M_p \| M_s$ ).  
Далее выполнить последовательность вычислений:

3.2  $H := x(M_s, H)$

3.3  $L := \langle L + 256 \rangle_{256}$

3.4  $\Sigma := \Sigma \oplus M_s$

3.5  $M := M_p$

3.6 Перейти к этапу 2.

Значение величины  $H$ , полученное на шаге 2.7, является значением функции хэширования  $h(M)$ .

Проверочные примеры для вышеизложенной процедуры вычисления хэш-функции приведены в приложении А.

**ПРИЛОЖЕНИЕ А**  
(справочное)

**ПРОВЕРОЧНЫЕ ПРИМЕРЫ**

Заполнение узлов замены  $\pi_1, \pi_2, \dots, \pi_8$  и значение стартового вектора хэширования  $H$ , указанные в данном приложении, рекомендуется использовать только в проверочных примерах для настоящего стандарта

**А1 Использование алгоритма ГОСТ 28147**

В качестве шифрующего преобразования в приводимых ниже примерах используется алгоритм ГОСТ 28147 в режиме простой замены

При этом заполнение узлов замены  $\pi_1, \pi_2, \dots, \pi_8$  блока подстановки  $\pi$  следующее.

	8	7	6	5	4	3	2	1
0	1	D	4	6	7	5	E	4
1	F	B	B	C	D	8	B	A
2	D	4	A	7	A	1	4	9
3	0	1	0	1	1	D	C	2
4	5	3	7	5	0	A	6	D
5	7	F	2	F	8	3	D	8
6	A	5	1	D	9	4	F	0
7	4	9	D	8	F	2	A	E
8	9	0	3	4	E	E	2	6
9	2	A	6	A	4	F	3	B
10	3	E	8	9	6	C	8	1
11	E	7	5	E	C	7	1	C
12	6	6	9	0	B	6	0	7
13	B	8	C	3	2	0	7	F
14	8	2	F	B	5	9	5	5
15	C	C	E	2	3	B	9	3

В столбце с номером  $j, j=1,8$ , в строке с номером  $i, i=0,15$ , приведено значение  $\pi_j(i)$  в шестнадцатеричной системе счисления

**А2 Представление векторов**

Последовательности двоичных символов будем записывать как строки шестнадцатеричных цифр, в которых каждая цифра соответствует четырем знакам ее двоичного представления

**А3 Примеры вычисления значения хэш-функции**

В качестве стартового вектора хэширования принимают, например, нулевой вектор

$H=00000000\ 00000000\ 00000000\ 00000000$   
 $00000000\ 00000000\ 00000000\ 00000000$

**А3.1 Пусть необходимо выполнить хэширование сообщения**

$M=73657479\ 62203233\ 3D687467\ 6E656C20$   
 $2C656761\ 7373656D\ 20736920\ 73696854$

Выполняют присвоение начальных значений:  
 текста

$M=73657479\ 62203233\ 3D687467\ 6E656C20$   
 $2C656761\ 7373656D\ 20736920\ 73696854$

## хэш-функции

H = 00000000 00000000 00000000 00000000  
 00000000 00000000 00000000 00000000

## суммы блоков текста

$\Sigma$  = 00000000 00000000 00000000 00000000  
 00000000 00000000 00000000 00000000

## длины текста

L = 00000000 00000000 00000000 00000000  
 00000000 00000000 00000000 00000000

Так как длина сообщения, подлежащего хэшированию, равна 256 битам (32 байтам),

I = 0C900000 00000000 00000000 00000000  
 00000000 00000000 00000000 00000100

M' = M = 73657479 62203233 3D687467 6E656C20  
 2C656761 7373656D 20736920 73696854, то

нет необходимости дописывать текущий блок нулями,

$\Sigma$  = M' = 73657479 62203233 3D687467 6E656C20  
 2C656761 7373656D 20736920 73696854

Переходят к вычислению значения шаговой функции хэширования  $\kappa$  (M, H).  
 Вырабатывают ключи

K<sub>1</sub> = 733D2C20 65686573 74746769 79676120  
 626E7373 20657369 326C6568 33206D54

K<sub>2</sub> = 110C733D 0D166568 130E7474 06417967  
 1D0C626E 161A2065 090D326C 4D393320

K<sub>3</sub> = 80B111F3 730DF216 850013F1 C7E1F941  
 620C1DFF 3ABAE91A 3FA109F2 F513B239

K<sub>4</sub> = A0E2894E FF1B73F2 ECE27A00 E7B8C7E1  
 EE1D620C AC0CC5BA A804C05E A18B0AEC

Осуществляют зашифрование 64-битных подслов блока H с помощью алгоритма по ГОСТ 28147.

Блок h<sub>1</sub> = 00000000 00000000 зашифровывают на ключе K<sub>1</sub> и получают s<sub>1</sub> = 42ABVCSCE 32BC0B1B.

Блок h<sub>2</sub> = 00000000 00000000 зашифровывают на ключе K<sub>2</sub> и получают s<sub>2</sub> = 5203EBC8 5D9BCFFD.

Блок h<sub>3</sub> = 00000000 00000000 зашифровывают на ключе K<sub>3</sub> и получают s<sub>3</sub> = 8D345899 00FF0E28.

Блок h<sub>4</sub> = 00000000 00000000 зашифровывают на ключе K<sub>4</sub> и получают s<sub>4</sub> = E7860419 0D2A562D.

Получают

S = E7860419 0D2A562D 8D345899 03FF0E28  
 5203EBC8 5D9BCFFD 42ABVCSCE 32BC0B1B

Выполняют перемешивающее преобразование с применением регистра сдвига и получают

$\Sigma = \kappa(M, H) =$  CF9A8C65 E35967A4 68A03B8C 42DE7624  
 D99C4124 883DA687 561C7DE3 3315C034

Полагают  $H = \Xi$ , вычисляют  $\kappa(L, H)$ .

$K_1 =$	CF68D956 50428833	9AA09C1C 59DE3D15	8C3B417D 6776A6C1	658C24E3 A4248734
$K_2 =$	8FCF68D9 BB504288	809AA9C 2859DE3D	3C8C3B41 666676A6	C7658C24 B3A42487
$K_3 =$	4C70CF97 CABB50BD	3C8C65A0 E3D7A6DE	853C8CC4 D1936788	57389A8C 5CB35B24
$K_4 =$	584E70CF EDCABB50	C53C8065 78E3D7A6	48853C8C EED19867	1657389A 7F5CB35B
$S =$	66B70F5E E5EC8A37	F163F461 3FD42279	468A9528 3CD1602D	61D60593 DD783E86
$\Xi =$	2B6EC233 DD3848D1	C7BC89E4 C6AC997A	2ABC2692 24F74E2B	5FEA7285 09A3AEF7

Вновь полагают  $H = \Xi$  и вычисляют  $\kappa(\Sigma, H)$

$K_1 =$	5817F104 A531B57A	0BD45D84 9C8FDFCA	B6522F27 BB1EFCC6	4AF5B00B D7A517A3
$K_2 =$	E82759E0 D2C73DA8	C278D950 19A6CAC9	15CC523C 3E8440F5	FC72EBB6 C0DDDB65A
$K_3 =$	77483AD9 FBC3DAA0	F7C29CAA 7CB555F0	EB06D1D7 D4968080	841BCAD3 0A9E56BC
$K_4 =$	A1157965 7684ADCB	2D9FBC9C FA4ACA06	088C7CC2 53EFF7D7	46FB3DD2 C0748708
$S' =$	2AFBFA76 C31E7435	A85FB57D 4930FD05	6F164DE9 1F8A4942	2951A581 550A582D
$\Xi =$	FAFF37A6 E09525F3	15A81669 9F811983	1CFF3EF8 2EB81975	B68CA247 D366C4B1

Таким образом, результат хэширования есть

$H =$	FAFF37A6 E09525F3	15A81669 9F811983	1CFF3EF8 2EB81975	B68CA247 D366C4B1
-------	----------------------	----------------------	----------------------	----------------------

А 3.2 Пусть необходимо выполнить хэширование сообщения

$M =$	7365 73656D20	74796220 6C616E69	3035203D 6769726F	20687467 20656874	6E656C20 2065736F	73616820 70707553	65676173
-------	------------------	----------------------	----------------------	----------------------	----------------------	----------------------	----------

Так как длина сообщения, подлежащего хэшированию, равна 400 битам (50 байтам), то разбивают сообщение на два блока и второй (старший) блок дописывают нулями. В процессе вычислений получают:

#### ШАГ 1

$H =$	00000000 00000000	00000000 00000000	00000000 00000000	00000000 00000000
$M =$	73616820 6769726F	65676173 20656874	73656D20 2065736F	6C616E69 70707553
$K_1 =$	73736720 656C2070	61656965 67616570	636D7273 616E6875	20206F6F 73697453

$K_2 =$	14477373 4C50656C	0C0C6165 04156761	1F01686D 061D616E	4F002020 1D277369
$K_3 =$	CBFF14B8 35094CAF	6D04F30C 72F9FB15	96051FFE 7CF006E2	DFFFBB00 AB1AE221
$K_4 =$	EBACCB00 BA1C3509	F7006DFB FD118DF9	E5E16905 F61B830F	B0B0DFFF F8C554E5
$S =$	FF41797C EDDC2210	EEAADAC2 1EE1ADF9	43C9B1DF FA67E757	2E14681C DAFE3AD9
$\Sigma =$	F0CEEA4E A93BEFBD	368B5A60 2634F0AD	C63D96C1 CBBB69CE	E5B51CD2 ED2D5D9A

### ШАГ 2

$H =$	F0CEEA4E A93BEFBD	368B5A60 2634F0AD	C63D96C1 CBBB69CE	E5B51CD2 ED2D5D9A
$M' =$	00000000 74796220	00000000 3035203D	00000000 20687467	00C07365 6E656C20
$K_1 =$	F0C6DDEB 36E51683	CE3D42D3 8BB50148	EA968D1D 5A6FD031	4EC19DA9 60B790BA
$K_2 =$	16A4C6A9 FB68E526	F9DF3D3B 2CDBB534	E4FC96EF FE161C83	5339C1BD 6F7DD2C8
$K_3 =$	C49D846D 9DCB0644	1780482C D1E641E5	9086887F A02109AF	C48C9186 9D52C7CF
$K_4 =$	BDB0C9F0 1CAD9536	756E9131 F4E4B674	E1F290EA 99F31E29	50E4CBB1 70C52AFA
$S =$	62A07EA5 6881EB66	EF3C3309 F5C7959F	2CE1B076 63FCA1F1	173D48CC D33C31B8
$\Sigma =$	95BEA0BE B8287CB6	88D5AA02 2CBC135B	FE3C9D45 3E339EFE	436CE821 F6576CA9

### ШАГ 3

$H =$	95BEA0BE B8287CB6	88D5AA02 2CBC135B	FE3C9D45 3E339EFE	436CE821 F6576CA9
$L =$	00000000 00000000	00000000 00000000	00000000 00000190	
$K_1 =$	95FEB83E 88432CF6	BE3C2833 D56CBC57	A09D7C9E AAE8136D	BE45B6FE 02215B39
$K_2 =$	8695FEB8 DA88432C	1BBE3C28 EBD56CBC	E2A09D7C 7FABE813	48BE45B6 F292215B
$K_3 =$	B9799501 6FDA88BC	141B413C D0142A6C	1EE2A062 FA80AA16	0CB74145 15F2FDB1
$K_4 =$	94B97995 346FDA88	7D141B41 46D0142A	C21EE2A0 BDF8A1AA	040CB741 DC1562FD
$S =$	D42336E0 9FDDFF20	2A0A6998 48C8E863	6C65478A 941D9D6D	3D08A1B9 F776A7AD

E =	47E26AFD A3D97E7E	3E7278A1 A744CB43	7D473785 08AA4C24	06140773 3352C745
-----	----------------------	----------------------	----------------------	----------------------

## ШАГ 4

H =	47E26AFD A3D97E7E	3E7278A1 A744CB43	7D473785 08AA4C24	06140773 3352C745
Σ =	73616820 DBE2D48F	65676173 509A88B1	73656D20 40CDE7D6	6C61E1CE DED5E173
K <sub>1</sub> =	340E7848 5B6AF7ED	83223B67 1575DE87	025AAAAB 19E64326	DDA5F1F2 D2BDF236
K <sub>2</sub> =	03DC0ED0 A8B063CB	F4CD26BC ED3D7325	8B595F13 6511662A	F5A4A55E 7963008D
K <sub>3</sub> =	C954EF19 4A9D0277	D0779A68 78EF765B	ED37D3FB C4731191	7DA5ADDC 7EBB21B1
K <sub>4</sub> =	6D12BC47 F2137F37	D9363D19 64E4C18B	1E3C696F 69CCFBF8	28F2DC02 EF72B7E3
S =	790DD7A1 25EF9645	066544EA EE2C05DD	2829563C A5ECAD92	3C39D781 2511A4D1
E =	0852F562 EAFBC135	3B89DD57 0613763A	AEB4781F 0D770AA6	E54DF14E 57BA1A47

Таким образом, результат хэширования есть

H =	0852F562 EAFBC135	3B89DD57 0613763A	AEB4781F 0D770AA6	E54DF14E 57BA1A47
-----	----------------------	----------------------	----------------------	----------------------

---

УДК 681.3.06:006.354

П85

ОКСТУ 5002

Ключевые слова: информационная технология, криптографическая защита информации, электронная цифровая подпись, асимметричный криптографический алгоритм, системы обработки информации, защита сообщений, подтверждение подписи, хэш-функция, функция хэширования

---

Редактор **Л. В. Афанасенко**  
Технический редактор **Н. С. Гришанова**  
Корректор **А. С. Черноусова**

Сдано в наб 24 06 94      Подп в печ 19 08 94      Усл п л 0 93      Усл кр.-отт **0,98.**  
Уч -изд л 0,84      Тираж 300 экз      С 1585

---

Ордена «Знак Почета» Издательство стандартов, 107076, Москва, Колодезный пер., 14.  
Тип «Московский печатник» Москва, Лялин пер, 6      Зак 208