ГОСУДАРСТВЕННЫЙ СТАНДАРТ
РЕСПУБЛИКИ БЕЛАРУСЬ

СТБ 34.101.77-2016

Информационные технологии и безопасность
АЛГОРИТМЫ ХЭШИРОВАНИЯ

Інфармацыйныя тэхналогіі і бяспека
АЛГАРЫТМЫ ХЭШАВАННЯ

Госстандарт
Минск

УДК МКС 35.240.40 КП 05

Ключевые слова: криптографический алгоритм, контроль целостности, хэширование

Предисловие

Цели, основные принципы, положения по государственному регулированию и управлению в области технического нормирования и стандартизации установлены Законом Республики Беларусь «О техническом нормировании и стандартизации».

1 РАЗРАБОТАН учреждением Белорусского государственного университета «Научно-исследовательский институт прикладных проблем математики и информатики»

ВНЕСЕН Оперативно-аналитическим центром при Президенте Республики Беларусь

2 УТВЕРЖДЕН И ВВЕДЕН В ДЕЙСТВИЕ постановлением Госстандарта Республики Беларусь от 12 августа 2016 г. № 62

3 ВВЕДЕН ВПЕРВЫЕ

Издан на русском языке

Содержание

ГОСУДАРСТВЕННЫЙ СТАНДАРТ РЕСПУБЛИКИ БЕЛАРУСЬ

Информационные технологии и безопасность
АЛГОРИТМЫ ХЭШИРОВАНИЯ

Інфармацыйныя тэхналогіі і бяспека
АЛГАРЫТМЫ ХЭШАВАННЯ

Information technology and security
Hashing algorithms

Дата введения 2016-10-01

1 Область применения

Настоящий стандарт устанавливает семейство криптографических алгоритмов хэширования, которые используются для контроля целостности и других способов защиты информации при ее хранении, передаче и обработке.

Настоящий стандарт применяется при разработке средств криптографической защиты информации.

2 Нормативные ссылки

В настоящем стандарте использованы ссылки на следующие технические нормативные правовые акты в области технического нормирования и стандартизации (далее — ТНПА):

СТБ 34.101.31-2011 Информационные технологии и безопасность. Защита информации. Криптографические алгоритмы шифрования и контроля целостности

СТБ 34.101.45-2013 Информационные технологии и безопасность. Алгоритмы электронной цифровой подписи и транспорта ключа на основе эллиптических кривых

ГОСТ 34.973-91 (ИСО 8824-87) Информационная технология. Взаимосвязь открытых систем. Спецификация абстрактно-синтаксической нотации версии 1 (АСН.1)

Примечание — При пользовании настоящим стандартом целесообразно проверить действие ТНПА по каталогу, составленному по состоянию на 1 января текущего года, и по соответствующим информационным указателям, опубликованным в текущем году.

Если ссылочные ТНПА заменены (изменены), то при пользовании настоящим стандартом следует руководствоваться замененными (измененными) ТНПА. Если ссылочные ТНПА отменены без замены, то положение, в котором дана ссылка на них, применяется в части, не затрагивающей эту ссылку.

3 Термины и определения и сокращения

В настоящем стандарте применяют следующие термины с соответствующими определениями:

3.1 сообщение: Двоичное слово конечной длины.

3.2 хэш-значение: Двоичное слово фиксированной длины, которое определяется по сообщению без использования ключа и служит для контроля целостности сообщения и для представления сообщения в (необратимо) сжатой форме.

3.3 хэширование: Выработка хэш-значений.

3.4 целостность: Гарантия того, что сообщение не изменено при хранении или передаче.

В настоящем стандарте применяют следующее сокращение:

ЭЦП — электронная цифровая подпись.

4 Обозначения

4.1 Список обозначений

$\{0,1\}^n$	множество всех слов длины n в алфавите $\{0,1\}$;
$\{0,1\}^*$	множество всех слов конечной длины в алфавите $\{0,1\}$ (включая пустое слово длины 0);
$\lvert u\rvert$	длина слова $u \in \{0,1\}^*$;
$\{0,1\}^{n*}$	множество всех слов из $\{0,1\}^*$, длина которых кратна n;
α^n	слово длины n из одинаковых символов $\alpha \in \{0,1\}$;
$\mathtt{Lo}_m(u)$	слово из первых m символов слова u, $m \leqslant \lvert u\rvert$;
$u \parallel v$	конкатенация $u_1u_2\ldots u_nv_1v_2\ldots v_m$ слов $u = u_1u_2\ldots u_n$ и $v = v_1v_2\ldots v_m$;
$\mathtt{01234...}_{16}$	представление $u \in \{0,1\}^{4*}$ шестнадцатеричным словом, при котором последовательным четырем символам u соответствует один шестнадцатеричный символ (например, $10100010 = \mathtt{A2}_{16}$);
$U \bmod m$	для целого U и натурального m остаток от деления U на m;
$u \oplus v$	для $u = u_1u_2\ldots u_n \in \{0,1\}^n$ и $v = v_1v_2\ldots v_n \in \{0,1\}^n$ слово $w = w_1w_2\ldots w_n \in \{0,1\}^n$ из символов $w_i = (u_i + v_i) \bmod 2$ (посимвольное исключающее ИЛИ);
$\neg u$	для $u \in \{0,1\}^n$ слово $u \oplus 1^n$ (посимвольное НЕ);
$u \wedge v$	для $u = u_1u_2\ldots u_n \in \{0,1\}^n$ и $v = v_1v_2\ldots v_n \in \{0,1\}^n$ слово $w = w_1w_2\ldots w_n \in \{0,1\}^n$ из символов $w_i = u_i * v_i$ (посимвольное И);
$u \vee v$	для $u = u_1u_2\ldots u_n \in \{0,1\}^n$ и $v = v_1v_2\ldots v_n \in \{0,1\}^n$ слово $w = w_1w_2\ldots w_n \in \{0,1\}^n$ из символов $w_i = (u_i * v_i + u_i + v_i) \bmod 2$ (посимвольное ИЛИ);
$\bar{u}$	а) для $u = u_1u_2\ldots u_8 \in \{0,1\}^8$ число $2^7u_1 + 2^6u_2 + \ldots + u_8$ и б) для $u = u_1 \parallel u_2 \parallel \ldots \parallel u_n$, $u_i \in \{0,1\}^8$, число $\bar{u}_1 + 2^8\bar{u}_2 + \ldots + 2^{8(n-1)}\bar{u}_n$;
$\langle U\rangle_{8n}$	для целого U слово $u \in \{0,1\}^{8n}$ такое, что $\bar{u} = U \bmod 2^{8n}$;
$\lfloor z\rfloor$	для вещественного z максимальное целое, не превосходящее z;
$\mathtt{ShLo}(u)$	для $u \in \{0,1\}^{8n}$ слово $\langle\lfloor\bar{u}/2\rfloor\rangle_{8n}$;
$\mathtt{ShHi}(u)$	для $u \in \{0,1\}^{8n}$ слово $\langle 2\bar{u}\rangle_{8n}$;
$\varphi^r(u)$	для слова u и преобразования φ результат r-кратного действия φ на u (например, $\mathtt{ShLo}^r(u)$ — результат r-кратного действия $\mathtt{ShLo}$);

$\texttt{RotHi}(u)$ для $u \in \{0,1\}^{8n}$ слово $\texttt{ShHi}(u) \oplus \texttt{ShLo}^{8n-1}(u)$;

$a \leftarrow u$ присвоение переменной a значения u.

4.2 Пояснения к обозначениям

4.2.1 Слова

Входными и выходными данными алгоритмов настоящего стандарта являются двоичные слова — последовательности символов алфавита $\{0,1\}$. Символы нумеруются слева направо от единицы. В настоящем подразделе в качестве примера рассматривается слово

$$w = 00000001001000110100010101100111100010011010101111001101111011111.$$

В этом слове первые семь символов нулевые, восьмой — 1, девятый — 0, ..., последний — 1.

Слова разбиваются на тетрады из четверок последовательных двоичных символов. Тетрады кодируются шестнадцатеричными символами по правилам, приведенным в таблице 1.

Таблица 1 — Шестнадцатеричные символы

тетрада	символ	тетрада	символ	тетрада	символ	тетрада	символ
0000	$\texttt{0}_{16}$	0001	$\texttt{1}_{16}$	0010	$\texttt{2}_{16}$	0011	$\texttt{3}_{16}$
0100	$\texttt{4}_{16}$	0101	$\texttt{5}_{16}$	0110	$\texttt{6}_{16}$	0111	$\texttt{7}_{16}$
1000	$\texttt{8}_{16}$	1001	$\texttt{9}_{16}$	1010	$\texttt{A}_{16}$	1011	$\texttt{B}_{16}$
1100	$\texttt{C}_{16}$	1101	$\texttt{D}_{16}$	1110	$\texttt{E}_{16}$	1111	$\texttt{F}_{16}$

Например, слово w кодируется следующим образом:

$$\texttt{0123456789ABCDEF}_{16}.$$

Пары последовательных тетрад образуют октеты. Последовательные октеты слова w имеют вид:

$$00000001 = \texttt{01}_{16},\ 00100011 = \texttt{23}_{16},\ 01000101 = \texttt{45}_{16},\ 01100111 = \texttt{67}_{16},$$
$$10001001 = \texttt{89}_{16},\ 10101011 = \texttt{AB}_{16},\ 11001101 = \texttt{CD}_{16},\ 11101111 = \texttt{EF}_{16}.$$

4.2.2 Слова как числа

Октету $u = u_1 u_2 \dots u_8$ ставится в соответствие байт — число $\bar{u} = 2^7 u_1 + 2^6 u_2 + \dots + u_8$. Например, октетам w соответствуют байты

$$1,\ 35 = 2^5 + 2^1 + 1,\ 69 = 2^6 + 2^2 + 1,\ 103 = 2^6 + 2^5 + 2^2 + 2^1 + 1,$$
$$137 = 2^7 + 2^3 + 1,\ 171 = 2^7 + 2^5 + 2^3 + 2^1 + 1,\ 205 = 2^7 + 2^6 + 2^3 + 2^2 + 1,$$
$$239 = 2^7 + 2^6 + 2^5 + 2^3 + 2^2 + 2^1 + 1.$$

Число ставится в соответствие не только октету, но и любому другому двоичному слову, длина которого кратна 8. При этом используется распространенное для многих современных процессоров соглашение «от младших к старшим» (little-endian): считается,

что первый байт является младшим, последний — старшим. Например, слову w соответствует число

$$\bar{w} = 1 + 2^8 \cdot 35 + 2^{16} \cdot 69 + 2^{24} \cdot 103 + 2^{32} \cdot 137 + 2^{40} \cdot 171 + 2^{48} \cdot 205 + 2^{56} \cdot 239 = \\ = 17279655951921914625.$$

При отождествлении слов с числами удобно представить себе гипотетический регистр, разрядность которого совпадает с длиной слова. В самый правый октет регистра загружается первый октет слова, во второй справа октет регистра — второй октет слова и т. д., пока, наконец, в самый левый октет регистра не загружается последний октет слова. Например, для w содержимое регистра имеет вид:

$$\texttt{EFCDAB8967452301}_{16} = 111011111100 \ldots 001100000001.$$

При таком представлении операции `ShLo`, `ShHi`, `RotHi` состоят в сдвигах содержимого регистра: `ShLo` — вправо (в сторону младших разрядов), `ShHi` — влево (в сторону старших разрядов) и `RotHi` — циклически влево, причем при сдвигах `ShLo` и `ShHi` в освободившиеся разряды регистров записываются нули. Например, предыдущий регистр изменяется при сдвигах следующим образом:

$$\begin{aligned} \texttt{ShLo}: &\ \texttt{77E6D5C4B3A29180}_{16} = 01110111 \ldots 10000000, \\ \texttt{ShHi}: &\ \texttt{DF9B5712CE8A4602}_{16} = 11011111 \ldots 00000010, \\ \texttt{RotHi}: &\ \texttt{DF9B5712CE8A4603}_{16} = 11011111 \ldots 00000011. \end{aligned}$$

Выгружая из регистра октеты слева направо, получаем следующие результаты:

$$\begin{aligned} \texttt{ShLo}(w) = &\ \texttt{8091A2B3C4D5E677}_{16}, \\ \texttt{ShHi}(w) = &\ \texttt{02468ACE12579BDF}_{16}, \\ \texttt{RotHi}(w) = &\ \texttt{03468ACE12579BDF}_{16}. \end{aligned}$$

Перестановки октетов при загрузке слова в регистр и при выгрузке из регистра в современных процессорах выполняются неявно.

4.3 Запись перечислений

При записи разбиения $X = X_1 \parallel X_2 \parallel \ldots \parallel X_n$ допускается, что $n = 1$. В этом случае X состоит из единственного блока X_1.

Аналогично, при $n = 1$ цикл «для $i = 1, 2, \ldots, n$» выполняется один раз.

5 Общие положения

5.1 Назначение

Настоящий стандарт определяет семейство криптографических алгоритмов хэширования, предназначенных для контроля целостности и необратимого сжатия данных. Обрабатываемыми данными являются двоичные слова (сообщения).

Алгоритм хэширования по сообщению произвольной длины строит хэш-значение — слово фиксированной длины. Стороны могут организовать контроль целостности сообщений путем сравнения их хэш-значений с достоверными контрольными хэш-значениями.

Изменение сообщения с высокой вероятностью приводит к изменению соответствующего хэш-значения, и поэтому хэш-значения могут использоваться вместо самих сообщений, например в системах ЭЦП.

Алгоритмы хэширования могут дополнительно использоваться при построении систем имитозащиты, генераторов случайных и псевдослучайных чисел, протоколов аутентификации, доказательств вычислительной работы и др.

Примечание — Алгоритм хэширования установлен также в СТБ 34.101.31. Переход от алгоритма СТБ 34.101.31 к алгоритмам настоящего стандарта позволит увеличить скорость хэширования по крайней мере на паритетном уровне стойкости и на 64-разрядных аппаратных платформах. Кроме этого, алгоритмы хэширования настоящего стандарта поддерживают все три уровня стойкости алгоритмов ЭЦП, определенных в СТБ 34.101.45, в то время как алгоритм СТБ 34.101.31 поддерживает только первый уровень.

Примеры выполнения алгоритмов стандарта приведены в приложении А. Примеры можно использовать для проверки корректности реализаций алгоритмов.

Модуль абстрактно-синтаксической нотации версии 1 (АСН.1), определенной в ГОСТ 34.973, приведен в приложении Б. Модуль задает идентификаторы алгоритмов стандарта, в том числе в их связках с алгоритмами СТБ 34.101.45. Рекомендуется использовать модуль при встраивании алгоритмов в информационные системы, в которых также используется АСН.1.

5.2 Шаговая функция

Алгоритмы хэширования построены по схеме sponge (губка), описанной в [1]. Ядром схемы является шаговая функция, которая определяет сложное биективное преобразование слов большой длины.

В настоящем стандарте шаговая функция действует на слова длины 1536. Действие задается алгоритмом `bash-f`, определенным в 6.2.

Шаговая функция `bash-f` имеет самостоятельное значение и может использоваться за пределами настоящего стандарта для построения других криптографических алгоритмов.

5.3 Уровень стойкости

Алгоритмы хэширования настоящего стандарта отличаются уровнем стойкости l. Это натуральное число, кратное 16 и не превосходящее 256. Алгоритм уровня l вычисляет хэш-значения длины $2l$, обрабатывая входные слова блоками длины $1536 - 4l$. Уровни $l = 128$, $l = 192$ и $l = 256$ являются стандартными, им следует отдавать предпочтение.

При выборе l следует учитывать, что для определения сообщения с заданным хэш-значением требуется выполнить порядка 2^{2l} операций, а для определения двух различных сообщений с одинаковыми хэш-значениями требуется выполнить порядка 2^l операций.

Следует учитывать также, что с ростом l, кроме повышения стойкости, снижается быстродействие алгоритмов. В частности, хэширование на уровне $l = 256$ выполняется примерно в 2 раза медленнее, чем на уровне $l = 128$.

5.4 Хэш-значение

Длина хэш-значения регулируется уровнем стойкости l. Если при фиксированном l требуются не все, а $n < 2l$ символов хэш-значения, то должны использоваться первые n символов.

6 Вспомогательные алгоритмы

6.1 Алгоритм bash-s

6.1.1 Входные и выходные данные

Входными данными алгоритма bash-s являются слова $W_0, W_1, W_2 \in \{0,1\}^{64}$ и числа $m_1, n_1, m_2, n_2 \in \{1, 2, \ldots, 63\}$.

Выходными данными являются преобразованные слова W_0, W_1, W_2.

6.1.2 Переменные

Используются переменные $T_0, T_1, T_2 \in \{0,1\}^{64}$.

6.1.3 Шаги алгоритма

Преобразование слов W_0, W_1, W_2 состоит в выполнении следующих шагов:

1 $T_0 \leftarrow \texttt{RotHi}^{m_1}(W_0)$.
2 $W_0 \leftarrow W_0 \oplus W_1 \oplus W_2$.
3 $T_1 \leftarrow W_1 \oplus \texttt{RotHi}^{n_1}(W_0)$.
4 $W_1 \leftarrow T_0 \oplus T_1$.
5 $W_2 \leftarrow W_2 \oplus \texttt{RotHi}^{m_2}(W_2) \oplus \texttt{RotHi}^{n_2}(T_1)$.
6 $T_0 \leftarrow \neg W_2$.
7 $T_1 \leftarrow W_0 \vee W_2$.
8 $T_2 \leftarrow W_0 \wedge W_1$.
9 $T_0 \leftarrow T_0 \vee W_1$.
10 $W_1 \leftarrow W_1 \oplus T_1$.
11 $W_2 \leftarrow W_2 \oplus T_2$.
12 $W_0 \leftarrow W_0 \oplus T_0$.
13 Возвратить (W_0, W_1, W_2).

6.2 Алгоритм bash-f

6.2.1 Входные и выходные данные

Входными данными алгоритма bash-f является слово $S \in \{0,1\}^{1536}$.

Выходными данными является преобразованное слово S.

Слово S записывается в виде $S = S_0 \parallel S_1 \parallel \ldots \parallel S_{23}$, $S_i \in \{0,1\}^{64}$.

6.2.2 Переменные

Используются переменные $C \in \{0,1\}^{64}$ и $m_1, n_1, m_2, n_2 \in \{1, 2, \ldots, 63\}$.

6.2.3 Шаги алгоритма

Преобразование слова S состоит в выполнении следующих шагов:

1 $C \leftarrow \texttt{B194BAC80A08F53B}_{16}$.

2 Для $i = 1, 2, \ldots, 24$ выполнить:

1) $(m_1, n_1, m_2, n_2) \leftarrow (8, 53, 14, 1)$;

2) для $j = 0, 1, \ldots, 7$:

(a) $(S_j, S_{8+j}, S_{16+j}) \leftarrow \texttt{bash-s}(S_j, S_{8+j}, S_{16+j}, m_1, n_1, m_2, n_2)$;

(b) $(m_1, n_1, m_2, n_2) \leftarrow (7\,m_1 \bmod 64, 7\,n_1 \bmod 64, 7\,m_2 \bmod 64, 7\,n_2 \bmod 64)$;

3) $S \leftarrow S_{15} \| S_{10} \| S_9 \| S_{12} \| S_{11} \| S_{14} \| S_{13} \| S_8 \| S_{17} \| S_{16} \| S_{19} \| S_{18} \| S_{21} \| S_{20} \| S_{23} \| S_{22} \| S_6 \| S_3 \| S_0 \| S_5 \| S_2 \| S_7 \| S_4 \| S_1$;

4) $S_{23} \leftarrow S_{23} \oplus C$;

5) если $\bar{C}$ — четное, то $C \leftarrow \texttt{ShLo}(C)$;

иначе $C \leftarrow \texttt{ShLo}(C) \oplus \texttt{AED8E07F99E12BDC}_{16}$.

3 Возвратить S.

7 Алгоритмы хэширования

7.1 Входные и выходные данные

Входными данными алгоритма хэширования уровня стойкости l является сообщение $X \in \{0,1\}^*$.

Выходными данными является слово $Y \in \{0,1\}^{2l}$ — хэш-значение сообщения X.

7.2 Вспомогательные преобразования и переменные

Используются алгоритм `bash-f`, определенный в 6.2, и переменная $S \in \{0,1\}^{1536}$.

7.3 Шаги алгоритма

Хэширование сообщения X на уровне стойкости l состоит в выполнении следующих шагов:

1 Дописать к X сначала слово 01, а затем t символов 0, где t — минимальное неотрицательное целое, для которого $|X| + 2 + t$ кратно $1536 - 4l$.

2 Полученное слово $X \| 01 \| 0^t$ записать в виде $X_1 \| X_2 \| \ldots \| X_n$, $X_i \in \{0,1\}^{1536-4l}$.

3 $S \leftarrow 0^{1472} \| \langle l/4 \rangle_{64}$.

4 Для $i = 1, 2, \ldots, n$ выполнить:

1) $\texttt{Lo}_{1536-4l}(S) \leftarrow X_i$;

2) $S \leftarrow \texttt{bash-f}(S)$.

5 $Y \leftarrow \texttt{Lo}_{2l}(S)$.

6 Возвратить Y.

Приложение А
(справочное)
Проверочные примеры

А.1 Алгоритм bash-s

В таблице А.1 представлен пример выполнения алгоритма bash-s с входными параметрами $(m_1, n_1, m_2, n_2) = (8, 53, 14, 1)$.

Таблица А.1 — Алгоритм bash-s

Шаг	Слово	Вычисляется как	Значение
	W_0		$\texttt{B194BAC80A08F53B}_{16}$
	W_1		$\texttt{E12BDC1AE28257EC}_{16}$
	W_2		$\texttt{E9DEE72C8F0C0FA6}_{16}$
1	T_0	$\texttt{RotHi}^{m_1}(W_0)$	$\texttt{3BB194BAC80A08F5}_{16}$
2	W_0	$W_0 \oplus W_1 \oplus W_2$	$\texttt{B96181FE6786AD71}_{16}$
3	T_1	$W_1 \oplus \texttt{RotHi}^{n_1}(W_0)$	$\texttt{CDFB23D652B779DB}_{16}$
4	W_1	$T_0 \oplus T_1$	$\texttt{F64AB76C9ABD712E}_{16}$
5	W_2	$W_2 \oplus \texttt{RotHi}^{m_2}(W_2) \oplus \texttt{RotHi}^{n_2}(T_1)$	$\texttt{F1401A7713A9DFD3}_{16}$
6	T_0	$\neg W_2$	$\texttt{0EBFE588EC56202C}_{16}$
7	T_1	$W_0 \vee W_2$	$\texttt{F9619BFF77AFFFF3}_{16}$
8	T_2	$W_0 \wedge W_1$	$\texttt{B040816C02842120}_{16}$
9	T_0	$T_0 \vee W_1$	$\texttt{FEFFF7ECFEFF712E}_{16}$
10	W_1	$W_1 \oplus T_1$	$\texttt{0F2B2C93ED128EDD}_{16}$
11	W_2	$W_2 \oplus T_2$	$\texttt{41009B1B112DFEF3}_{16}$
12	W_0	$W_0 \oplus T_0$	$\texttt{479E76129979DC5F}_{16}$

А.2 Шаговая функция

В таблице А.2 представлен пример выполнения алгоритма bash-f. Приводится результат выполнения $i = 1, 2, 3$ итераций шага 2 алгоритма, а также окончательное выходное значение.

Таблица А.2 — Шаговая функция

S	B194BAC80A08F53B 366D008E584A5DE4 8504FA9D1BB6C7AC 252E72C202FDCE0D 5BE3D61217B96181 FE6786AD716B890B 5CB0C0FF33C356B8 35C405AED8E07F99 E12BDC1AE28257EC 703FCCF095EE8DF1 C1AB76389FE678CA F7C6F860D5BB9C4F F33C657B637C306A DD4EA7799EB23D31 3E98B56E27D3BCCF 591E181F4C5AB793 E9DEE72C8F0C0FA6 2DDB49F46F739647 06075316ED247A37 39CBA38303A98BF6 92BD9B1CE5D14101 5445FBC95E4D0EF2 682080AA227D642F 2687F93490405511$_{16}$
$S\ (i = 1)$	E2B6A7F6F035D3F2 39480309210BEE8D DED2F39B17FE7C73 4ECA319DCCB1FF76 7BC40A127CF4877A E7FB536FE9390C54 99F34A34D10940B3 0F2B2C93ED128EDD EEB12106DC4F0DFD 41009B1B112DFEF3 BC6D797961DEC912 60E31EF060BE55EB C45AFC52E748DC91 2CAFCA63316F4885 51293EE80CC2D263 22368797C4123CC4 D7C509C309827DE3 2C98DECE4BC4A759 479E76129979DC5F 08C16DF28F6305A6 9D17224CB6817E27 F5823D9AFB05B086 C917D78B6ECAA711 EB72E1BF436E40E7$_{16}$
$S\ (i = 2)$	BA9659361A0C4CEE 4E3D7DBEA2105A0F C013BAF75A0D25A7 B75E9FD11911F45D FC69D759AECDE7C3 03EF0B29E992C6F8 8B9DE3850D8DFE0C 1BDDCE12F8D6FA9A AF72F482DF11C7CD E5BF7296886D1FAB 4752419560C91DB8 5FB21DB9B8FDE868 C6DC94B8011B4EA1 AE7B7EAD5C8259EA 22DAD6B09B827CD1 D93F3E3D9AB7A83D FF1D5681C46C4A06 9D57FC71FC5A5544 25A032DE52434699 43B5DBE9DDA545A8 94EE1EB7B0B6DEC9 1B02E5748F9141C1 7B2C3572CAC28A7B DDAFB4BA42799C9C$_{16}$
$S\ (i = 3)$	DFCF8BEE927CFE37 5D9C4D5CAF40D3CB B9D88D53C69035BB 5731D745CC819EBA E2997B65309B248A 84D02D7449D95208 0B501107F1758917 D088BB8CB4CC72C1 EB04E3084DA79297 E636CC72732EFD58 1F31744F59995332 28C3061400E0C34B 9EAE60469BB4F1B6 1E37FA5B319F90FF D4B7D3F007592688 6EBB6B818BB9BAC4 2904D6B8AAABE559 56B7D63B932FA660 D5068CCACE824E9A 43696F09544AA03A 559E325797384232 3435388ADDBF17C4 479570E8E01E18EE 1BE353ABA3EA17EC$_{16}$
$\texttt{bash-f}(S)$	8FE727775EA7F140 B95BB6A200CBB28C 7F0809C0C0BC68B7 DC5AEDC841BD94E4 03630C301FC255DF 5B67DB53EF65E376 E8A4D797A6172F22 71BA48093173D329 C3502AC946767326 A2891971392D3F70 89959F5D61621238 655975E00E2132A0 D5018CEEDB17731C CD88FC50151D37C0 D4A3359506AEDC2E 6109511E7703AFBB 014642348D8568AA 1A5D9868C4C7E6DF A756B1690C7C2608 A2DC136F5997AB8F BB3F4D9F033C87CA 6070E117F099C409 4972ACD9D976214B 7CED8E3F8B6E058E$_{16}$

А.3 Хэширование

В таблице А.3 представлены примеры хэширования. В таблице для различных уровней стойкости l приводятся хэш-значения сообщения $\texttt{Lo}_{8n}(X)$, где X — слово из таблицы А.2.

Таблица А.3 — Хэширование

n	Хэш-значение
	$l = 128$
0	114C3DFAE373D9BC BC3602D6386F2D6A 2059BA1BF9048DBA A5146A6CB775709D$_{16}$
127	3D7F4EFA00E9BA33 FEED259986567DCF 5C6D12D51057A968 F14F06CC0F905961$_{16}$
128	D7F428311254B8B2 D00F7F9EEFBD8F30 25FA87C4BABD1BDD BE87E35B7AC80DD6$_{16}$
135	1393FA1B65172F2D 18946AEAE576FA1C F54FDD354A0CB297 4A997DC4865D3100$_{16}$
	$l = 192$
95	64334AF830D33F63 E9ACDFA184E32522 103FFF5C6860110A 2CD369EDBC04387C 501D8F92F749AE4D E15A8305C353D64D$_{16}$
96	D06EFBC16FD6C088 0CBFC6A4E3D65AB1 01FA82826934190F AABEBFBFFEDE93B2 2B85EA72A7FB3147 A133A5A8FEBD8320$_{16}$
108	FF763296571E2377 E71A1538070CC0DE 88888606F32EEE6B 082788D246686B00 FC05A17405C55176 99DA44B7EF5F55AB$_{16}$
	$l = 256$
63	2A66C87C189C12E2 55239406123BDEDB F19955EAF0808B2A D705E249220845E2 0F4786FB6765D0B5 C48984B1B16556EF 19EA8192B985E423 3D9C09508D6339E7$_{16}$
64	07ABBF8580E7E5A3 21E9B940F667AE20 9E2952CEF557978A E743DB086BAB4885 B708233C3F5541DF 8AAFC3611482FDE4 98E58B3379A6622D AC2664C9C118A162$_{16}$
127	526073918F97928E 9D15508385F42F03 ADE3211A23900A30 131F8A1E3E1EE21C C09D13CFF6981101 235D895746A4643F 0AA62B0A7BC98A26 9E4507A257F0D4EE$_{16}$
192	8724C7FF8A2A83F2 2E38CB9763777B96 A70ABA3444F214C7 63D93CD6D19FCFDE 6C3D3931857C4FF6 CCCD49BD99852FE9 EAA7495ECCDD96B5 71E0EDCF47F89768$_{16}$

Приложение Б
(рекомендуемое)
Модуль ACH.1

В модуле ACH.1 определяются идентификаторы следующих алгоритмов:

`bash256`	алгоритм хэширования уровня стойкости $l = 128$;
`bash384`	алгоритм хэширования уровня стойкости $l = 192$;
`bash512`	алгоритм хэширования уровня стойкости $l = 256$;
`bash-f`	алгоритм вычисления значений шаговой функции (пункт 6.2).

Алгоритм хэширования уровня l определяет функцию хэширования $h\colon \{0,1\}^* \to \{0,1\}^{2l}$. Эта функция может использоваться в алгоритмах ЭЦП СТБ 34.101.45, уточняя их. Правила использования h определены в СТБ 34.101.45 (пункт 5.5). Уточненным алгоритмам ЭЦП присваиваются следующие идентификаторы:

`bign-with-bash256`	алгоритмы ЭЦП СТБ 34.101.45 (пункт 7.1) с функцией хэширования, заданной алгоритмом `bash256`;
`bign-with-bash384`	алгоритмы ЭЦП СТБ 34.101.45 с функцией хэширования, заданной алгоритмом `bash384`;
`bign-with-bash512`	алгоритмы ЭЦП СТБ 34.101.45 с функцией хэширования, заданной алгоритмом `bash512`;
`bign-ibs-with-bash256`	алгоритмы идентификационной ЭЦП СТБ 34.101.45 (приложение В, пункт В.1) с функцией хэширования, заданной алгоритмом `bash256`;
`bign-ibs-with-bash384`	алгоритмы идентификационной ЭЦП СТБ 34.101.45 с функцией хэширования, заданной алгоритмом `bash384`;
`bign-ibs-with-bash512`	алгоритмы идентификационной ЭЦП СТБ 34.101.45 с функцией хэширования, заданной алгоритмом `bash512`.

Модуль ACH.1 имеет следующий вид:

```
Bash-module-v1 {iso(1) member-body(2) by(112) 0 2 0 34 101 77 module(1) ver1(1)}
DEFINITIONS ::=
BEGIN
  IMPORTS
    bign
      FROM Bign-module-v2 {iso(1) member-body(2) by(112) 0 2 0 34 101 45
        module(1) ver2(2)};

  bash OBJECT IDENTIFIER ::= {iso(1) member-body(2) by(112) 0 2 0 34 101 77}

  bash256 OBJECT IDENTIFIER ::= {bash 11}
  bash384 OBJECT IDENTIFIER ::= {bash 12}
```

```
  bash512 OBJECT IDENTIFIER ::= {bash 13}
  bash-f OBJECT IDENTIFIER ::= {bash 101}

  bign-with-bash256 OBJECT IDENTIFIER ::= {bign 13}
  bign-with-bash384 OBJECT IDENTIFIER ::= {bign 14}
  bign-with-bash512 OBJECT IDENTIFIER ::= {bign 15}
  bign-ibs-with-bash256 OBJECT IDENTIFIER ::= {bign 73}
  bign-ibs-with-bash384 OBJECT IDENTIFIER ::= {bign 74}
  bign-ibs-with-bash512 OBJECT IDENTIFIER ::= {bign 75}
END
```

Библиография

[1] Bertoni G., Daemen J., Peeters M., Van Assche G. Cryptographic sponge functions Avail. at `http://sponge.noekeon.org/CSF-0.1.pdf`, 2011
(Бертони Г., Дэмен Дж., Питерс М., ван Аш Г. Криптографические sponge-функции)