УДК 681.3.06:519.248.681

В.Н. ВЕРВЕЙКО, А.И. ПУШКАРЕВ, Т.В. ЦЕПУРИТ

ФУНКЦИИ ХЭШИРОВАНИЯ: КЛАССИФИКАЦИЯ, ХАРАКТЕРИСТИКА И СРАВНИТЕЛЬНЫЙ АНАЛИЗ

Введение

Для решения задач обеспечения целостности наблюдаемости и подлинности информации применяются криптографические контрольные суммы. Методы формирования криптографических контрольных сумм можно разделить на два класса: на базе симметричных криптографических преобразований (коды аутентификации сообщений (КАС)) и использующие несимметричные преобразования (цифровые подписи) с применением секретных ключей. Такие функции могут применяться как непосредственно в качестве криптографической контрольной суммы, так и в других преобразованиях. Например, для формирования цифровой подписи необходима эффективная функция отображения сообщения в образ небольшой фиксированной длины (хэш-значение, хэш-код или просто хэш). Эти функции называют функциями хэширования или хэш-функциями.

1. Классификация функций хэширования

Функцией хэширования (в широком смысле) называется функция h, удовлетворяющая минимум двум требованиям [1]:

1. *Сжатие* – функция h отображает входное сообщение x произвольной конечной длины в хэш-значение $y = h(x)$ небольшой фиксированной длины, при этом входное сообщение будем называть прообразом.
2. *Простота вычисления* – для заданной функции h и сообщения x, $h(x)$ вычисляется не выше чем с полиномиальной сложностью.

Функции хэширования, используемые в криптографии, должны удовлетворять дополнительным требованиям, которые будут рассмотрены далее.

Все существующие функции хэширования можно разделить на два больших класса [1]: бесключевые хэш-функции, зависящие только от сообщения, и хэш-функции с секретным ключом, зависящие как от сообщения, так и от секретного ключа.

Подклассом бесключевых хэш-функций являются *коды обнаружения изменений* (modification detection codes, MDC-коды). В криптографии применяются специфические подклассы MDC-кодов, являющиеся однонаправленными и бесколлизионными хэш-функциями, которые получили широкое распространение в системах цифровой подписи.

Функции выработки кодов аутентификации сообщений (КАС) являются подклассом ключевых хэш-функций и обладают дополнительным свойством вычислительной стойкости.

По используемым внутренним преобразованиям функции хэширования можно разделить на:

- функции, использующие битовые логические преобразования. Эти функции применяют к входному сообщению побитовые нелинейные операции “И”, “ИЛИ”, “НЕ”, “ИСКЛЮЧАЮЩЕЕ ИЛИ”, различные сдвиги и, как правило, являются многоцикловыми;
- функции, использующие блочные симметричные шифры. Используются в основном для реализации функций выработки КАС;
- функции, использующие преобразования в группах, полях и кольцах с целочисленным или полиномиальным базисом;
- функции, использующие матричные преобразования.

2. Требования к применяемым в криптографии хэш-функциям

Анализ условий применения функций хэширования и практического их использования

позволил сформулировать требования, предъявляемые к применяемым в криптографии бесключевым хэш-функциям. Они состоят в следующем:

3. *Стойкость к вычислению прообраза* – невозможность нахождения неизвестного прообраза для любых предварительно заданных хэш-значений, т.е. для заданной хэш-функции h вычислительно невозможно найти неизвестный прообраз x при предварительно заданном хэш-значении $y = h(x)$ для любого значения y. Под термином "вычислительно невозможно" здесь и далее будем понимать, что алгоритм, выполняющий данное преобразование, обладает не менее чем экспоненциальной сложностью.
4. *Стойкость к вычислению второго прообраза* – невозможность нахождения любого другого прообраза, который давал бы такое же хэш-значение, как и заданный, т.е. для заданной хэш-функции h и прообраза x вычислительно невозможно найти другой прообраз $x' \neq x$, для которого выполнялось бы условие $h(x) = h(x')$.
5. *Стойкость к коллизиям* – невозможность нахождения двух прообразов для которых вырабатывалось бы одинаковое значение, т.е. для заданной хэш-функции h вычислительно невозможно найти два прообраза x и x', $x \neq x'$, для которых выполнялось бы условие $h(x) = h(x')$.

Требование стойкости к коллизиям является более жестким, чем требование стойкости к вычислению второго прообраза, так как предполагает произвольный выбор двух прообразов.

Однонаправленной хэш-функцией называется функция h, удовлетворяющая требованиям сжатия, простоты вычисления, стойкости к вычислению прообраза и стойкости к вычислению второго прообраза.

Бесколлизионной хэш-функцией называется функция h, удовлетворяющая требованиям сжатия, простоты вычисления, стойкости к вычислению второго прообраза и стойкости к коллизиям.

На практике обычно используются хэш-функции, являющиеся одновременно бесколлизионными и однонаправленными.

Однонаправленные хэш-функции могут применяться для решения других задач, например, выработки ключей и псевдослучайных чисел. Для применения в таких задачах хэш-функция должна удовлетворять следующим требованиям [2]:

6. *Отсутствие корреляции* – входные и выходные биты не должны коррелировать, т.е. изменение любого входного бита приводит к большим непредсказуемым изменениям выходных бит.
7. *Стойкость к близким коллизиям* – для заданной однонаправленной функции h вычислительно невозможно найти два прообраза x и x', для которых хэш-значения $h(x)$ и $h(x')$ отличались бы на малое количество бит.
8. *Стойкость к частичной однонаправленности* – вычислительно невозможно восстановить любую часть входного сообщения так же, как и все сообщение. Более того, по любой известной части входного сообщения вычислительно невозможно восстановить оставшуюся часть (восстановление t неизвестных бит требует не менее чем 2^{t-1} операций).
9. *Возможность работы в режиме растягивания* – возможность вычисления хэш-функции при длине входного сообщения меньше чем длина хэш-значения.

Требование, предъявляемое к применяемым в криптографии хэш-функциям с секретным ключом, следующее:

вычислительная стойкость – невозможность нахождения хэш-значения для заданного сообщения без известного секретного ключа, т.е. для заданной ключевой хэш-функции h и одной или более корректных пар прообразов и хэш-значений $(x_i, h(x_i, k))$ и неиз-

вестном секретном ключе k вычислительно невозможно найти другую корректную пару $(x, h(x,k))$ для любого $x \neq x_i$.

Требование вычислительной стойкости предполагает выполнение требования стойкости ключа (по одной или более корректных пар прообразов и хэш-значения $(x_i, h(x_i,k))$ вычислительно невозможно восстановить секретный ключ k), однако, требование стойкости ключа не предполагает выполнение требования вычислительной стойкости.

Функция хэширования с секретным ключом h является *функцией выработки КАС*, если выполняются требования сжатия, вычислительной простоты (при известном сеансовом ключе) и вычислительной стойкости.

Следует различать функции выработки КАС и однонаправленные хэш-функции с секретным ключом, являющимся частью сообщения, так как они обладают различными свойствами. В функциях выработки КАС секретный ключ применяется к каждому блоку сообщения, а в однонаправленных хэш-функциях ключ используется префиксным (в начале сообщения), постфиксным (в конце сообщения) или комбинированным методом, что снижает стойкость функции.

3. Возможные атаки на функции хэширования

Проведенный анализ показал, что на однонаправленные и бесколлизионные хэш-функции возможны следующие атаки:

1) нахождение прообраза x по заданному значению $y = h(x)$. Такая атака особенно опасна для систем аутентификации, использующих хэш-значения паролей и секретных ключей;
2) нахождение прообраза x' по заданному прообразу x, для которого выполняется условие $h(x) = h(x')$. Эта атака может быть использована для фальсификации сообщения, подписанного цифровой подписью;
3) нахождение двух прообразов x и x', $x \neq x'$, для которых выполнялось бы условие $h(x) = h(x')$.

На функции выработки КАС существует две атаки:

1) нахождение корректной пары прообраза и КАС $(x, h(x,k))$ по одной или более заданным корректным парам $(x_i, h(x_i,k))$ для любого $x \neq x_i$ при неизвестном секретном ключе k;
2) нахождение неизвестного сеансового ключа k по одной или более заданным корректным парам прообразов и кодов аутентификации $(x_i, h(x_i,k))$.

Атаки на функции КАС могут выполняться при следующих условиях:

1) *атака с известным текстом* – атакующему заданы только одна или несколько корректных пар прообразов и кодов аутентификации $(x_i, h(x_i,k))$;
2) *атака с выбираемым текстом* – атакующий имеет возможность получить корректные пары $(x_i, h(x_i,k))$ для выбранных x_i (атака на нахождение ключа);
3) *атака с адаптивным выбором текста* – атакующий может получить корректные пары $(x_i, h(x_i,k))$ для любых x_i, выбранных в зависимости от результатов предшествующих запросов (атака с целью нахождения ключа).

Все атаки на хэш-функции можно разделить на две группы: атаки, базирующиеся на уязвимости алгоритма преобразований (аналитические) и атаки, не зависящие от алгоритма.

Атаки, не зависящие от алгоритма: атака "грубой силой", атака методом "дня рождения", полный перебор ключей. К таким атакам уязвимы все алгоритмы, единственная возможность их избежать – увеличить длину хэш-значения, вырабатываемого однонаправленной или бесколлизионной хэш-функцией, и секретного ключа в функции выработки КАС.

Аналитические атаки: атака "встреча посередине", атака с коррекцией блока, атака с фиксированной точкой, атака на базовый алгоритм шифрования, дифференциальный анализ. Эти атаки основываются на недостатках внутренней структуры хэш-функций.

Атака "грубой силой" [3] может быть выполнена для нахождения прообраза по заданному хэш-значению или для нахождения прообраза, дающего заданное хэш-значение. Суть атаки заключается в последовательном или случайном переборе входных сообщений и сравнения результата выполнения хэш-функции с заданным. Сложность такой атаки оценивается 2^{l-1} операций вычисления хэш-значений, где l - длина хэш-значения в битах.

Атака методом "дня рождения" [4] выполняется для нахождения двух различных сообщений с одинаковыми хэш-значениями. Эта атака основана на парадоксе "дня рождения" и заключается в том, что в двух сгенерированных множествах хэш-значений, содержащих n_1 и n_2 элементов соответственно, вероятность нахождения совпадающих элементов между этими множествами оценивается следующей формулой:

$$P \approx 1 - e^{-\frac{n_1 n_2}{2^l}}.$$

В частности, при $n_1 = n_2 = 2^{\frac{l}{2}}$ сложность атаки оценивается как $2^{\frac{l}{2}+1}$ операций вычисления хэш-значений, а вероятность успеха равна

$$P \approx 1 - \frac{1}{e} \approx 0{,}63.$$

Атака полного перебора ключей осуществляется для нахождения неизвестного секретного сеансового ключа функции выработки КАС. Для нахождения ключа атакующий, имеющий не менее одной пары (сообщение, КАС), последовательно перебирает ключи. Так как пространство сообщений неоднозначно отображается в пространство хэш-значений, то может быть обнаружено множество подходящих значений ключей. Чтобы точно найти правильный ключ, необходимо выполнить проверку найденных ключей на большом множестве различных пар (сообщение, КАС). Как показано в [5] максимальное число попыток точного определения ключа составляет

$$m + \frac{2^k - 1}{1 - 2^{-l}},$$

где k - длина секретного ключа в битах, m - количество различных пар (сообщение, КАС). В среднем достаточно $\frac{k}{l}$ различных пар (сообщение, КАС) чтобы точно определить секретный ключ.

Атака "встреча посередине" [5] является модификацией атаки методом "дня рождения" и используется для хэш-функций с циклической структурой, если цикловая функция $f()$ инвертируема по отношению к промежуточному значению X или блоку сообщения M_i. Эта атака по сложности сопоставима с атакой методом "дня рождения".

Атака с коррекцией блока используется в случае, если атакующий обладает сообщением и хочет изменить в нем один или более блоков без изменения хэш-значения. Один цикл MD5 уязвим к этой атаке: атакующий берет блок сообщения M_i (16 слов по 32 бита), оставляет 11 слов, модифицирует одно слово, и вычисляет оставшиеся 4. В результате получается блок M_i', отображающийся в то же самое хэш-значение, что и M_i. Полная версия MD5 не уязвима к этой атаке.

Атака с фиксированной точкой [5] может применяться при условии, что цикловая функция f имеет одну или несколько фиксированных точек. Фиксированной точкой называется блок сообщения M_i, для которого выполняется $f(X_i, M_i) = X_i$, т.е. существует блок сообщения M_i, не изменяющий промежуточный результат X_i. Таким образом, в сообщение M можно добавлять или удалять блоки M_i без изменения хэш-значения. Защитой от таких атак служит вычисление длины сообщения и добавления ее в конце сообщения.

Атака на базовый алгоритм шифрования [6] используется для атаки на хэш-функции, базирующиеся на блочных симметричных шифрах. Так как алгоритмы шифрования разрабатывались как двунаправленные (поддерживают обратное преобразование), то это может увеличить уязвимость в функцию сжатия с их применением.

Дифференциальный анализ исследует зависимости между входными и выходными значения цикловой функции или функции сжатия с целью определения статистических аномалий. Дифференциальный анализ применим к различным криптографическим системам, включая функции хэширования [7, 8].

Оценочные значения вычислительной стойкости хэш-функций приведены в табл. 1.

Таблица 1

Тип хэш-функции	Цель атаки	Идеальная стойкость
однонаправленная хэш-функция	нахождение прообраза	2^l
	нахождение 2-го прообраза	2^l
бесколлизионная хэш-функция	нахождение любой коллизии	$2^{\frac{l}{2}}$
функция выработки КАС	точное нахождение ключа	$\left\lceil \frac{k}{l} \right\rceil + \frac{2^k - 1}{1 - 2^{-l}}$
	Подделка сообщения	$P_m = \max(2^{-k}, 2^{-l})$

В таблице используются следующие обозначения: l - длина хэш-значения или кода аутентификации, k - длина секретного ключа, P_m - вероятность успешной подделки сообщения.

4. Анализ современных хэш-функций

Рассмотрим однонаправленные и бесколлизионные функции хэширования и функции выработки КАС, используемые в настоящее время в криптографических системах защиты информации, а также некоторые перспективные: ГОСТ 34311-95, HAVAL, SHA-1, RIPEMD-160, MD4, MD5, ГОСТ 28147-89 режим 4, Rijndael CBC–MAC, Whirlpool, SHA-2, UMAC. Все бесключевые хэш-функции, являются однонаправленными и бесколлизионными, поэтому дальнейшем будем их называть однонаправленными.

В качестве примера рассмотрим новую перспективную функцию хэширования SHA-2, в которой в качестве основных преобразований используются нелинейные битовые преобразования. Новые требования к стойкости криптографических алгоритмов, выдвинутые в ходе разработки проектов AES и NESSIE [9], предполагают три основных уровня стойкости: 2^{128}, 2^{192} и 2^{256} (ключи длиной 128, 192 и 256 бит). Для обеспечения таких уровней защищенности минимальная длина хэш-значения должна составлять соответственно 256, 384 и 512 бит. Широко используемый в настоящее время алгоритм хэширования SHA-1 вырабатывает хэш-значение длиной 160 бит и обладает стойкостью к коллизиям 2^{80}. В связи с тем, что на алгоритм SHA-1 на текущий момент не найдена ни одна аналитическая атака, было решено доработать его для возможности формирования хэш-значений длиной 256, 384 и 512 бит. Соот-

ветственно алгоритм SHA-2 подразделяется на три алгоритма: SHA-256, SHA-384 и SHA-512. Хэш-значение вычисляется по следующей итеративной формуле

$$H_i = H_{i-1} + C(M_i, H_{i-1}), \tag{1}$$

где H_0 - фиксированный начальный вектор хэширования, C - цикловая функция сжатия, H_n - хэш-значение сообщения M, $+$ - операция сложения 32-х битовых слов по модулю 2^{32} для SHA-256 или 64-х битовых слов по модулю 2^{64} для SHA-384 и SHA-512.

Функция сжатия алгоритма SHA-256 отображает входное 512-и битовое значение в промежуточное значение длиной 256 бит. Перед сжатием выполняется расширение сообщения до 64-х 32-х разрядных блоков. Алгоритм SHA-2 использует 6 нелинейных функций:

$$\begin{aligned}
&Ch(x,y,z) = (x \wedge y) \oplus (\neg x \wedge z),\\
&Maj(x,y,z) = (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z),\\
&\sum_0(x) = S^2(x) \oplus S^{13}(x) \oplus S^{22}(x),\\
&\sum_1(x) = S^6(x) \oplus S^{11}(x) \oplus S^{25}(x),\\
&\sigma_0(x) = S^7(x) \oplus S^{18}(x) \oplus R^3(x),\\
&\sigma_1(x) = S^{17}(x) \oplus S^{19}(x) \oplus R^{10}(x),
\end{aligned} \tag{2}$$

где $\oplus$ - сложение по модулю 2, $\wedge$ - логическое И, $\neg$ - логическое НЕ, S^n - циклический сдвиг вправо на n разрядов, R^n - сдвиг вправо на n разрядов. Промежуточное значение разбивается на восемь 32-х битовых слов, над которыми 64 раза выполняется преобразование, показанные на рис. 1.

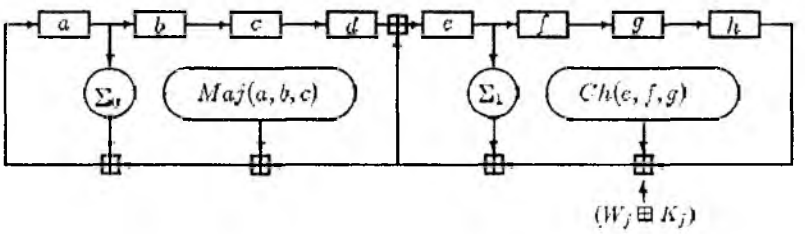

Рис. 1

На рис. 1 приняты следующие обозначения: + - сложение по модулю 2^{32}, K_j - цикловые константы j-го цикла, W_j - 32-х битовый элемент расширенного сообщения j-го цикла. Расширенное сообщение W получается из сообщения M с использованием преобразования, изображенного на рис. 2. Начальным заполнением регистра является сообщение M.

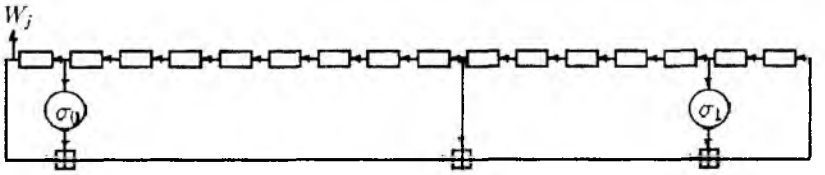

Рис. 2

Алгоритм SHA-512 аналогичен алгоритму SHA-256, но все операции в нем выполняются с 64-х битовыми блоками (сложение по модулю 2^{64}), количество циклов и элементов расширенного сообщения – 80, на вход поступает 1024-х битовый блок, а на выходе формирует-

ся 512-и битовое промежуточное значение, а также используются следующие логические функции:

$$
\begin{aligned}
&Ch(x,y,z) = (x \wedge y) \oplus (\neg x \wedge z),\\
&Maj(x,y,z) = (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z),\\
&\sum\nolimits_0(x) = S^{28}(x) \oplus S^{34}(x) \oplus S^{39}(x),\\
&\sum\nolimits_1(x) = S^{14}(x) \oplus S^{18}(x) \oplus S^{41}(x),\\
&\sigma_0(x) = S^{1}(x) \oplus S^{8}(x) \oplus R^{7}(x),\\
&\sigma_1(x) = S^{19}(x) \oplus S^{61}(x) \oplus R^{6}(x).
\end{aligned}
\qquad (3)
$$

Алгоритмы SHA-384 и SHA-512 одинаковы, за исключением того, что используются различные начальные заполнения, и используются только 384 бита полученного 512-и битового значения.

Некоторые характеристики хэш-функций представлена в табл. 2.

Таблица 2

Функция хэширования	Класс функции	Базовые преобразования	Длина хэш-значения, бит
Whirlpool	однонаправленная	в конечных полях и матрицах	512
SHA-2	однонаправленная	логические и арифметические	256, 384, 512
ГОСТ 34311-95	однонаправленная	блочный симметричный шифр	256
HAVAL	однонаправленная	логические и арифметические	128, 160, 192, 256
SHA-1	однонаправленная	логические и арифметические	160
RIPEMD-160	однонаправленная	логические и арифметические	160
MD5	однонаправленная	логические и арифметические	128
MD4	однонаправленная	логические и арифметические	128
UMAC	однонаправленная и выработка КАС	в кольцах	128, 64
Rijndael CBC–MAC	выработка КАС	блочный симметричный шифр	128
ГОСТ 28147-89 (режим 4)	выработка КАС	блочный симметричный шифр	64

Для оценки сложности реализации хэш-функций была измерена скорость работы официальных реализаций на языке C, а также для сравнения некоторые оптимизированные версии с использованием языка assembler. В качестве компилятора использовался Microsoft Visual C++ 6.0. Тестирование скорости работы проводилось на компьютерах Celeron 600 MHz, 128 MB RAM и Pentium III 1000 MHz, 256 MB RAM под управлением операционной системы Microsoft Windows 2000 Professional. Результаты тестирования скорости работы алгоритмов хэширования представлены в табл. 3.

Таблица 3

Функция хэширования	Количество циклов	Язык реализации	Скорость работы на Celeron 600 MHz	Скорость работы на Pentium III 1000 MHz
Whirlpool	10	C	28,013 Мбит/с	46,961 Мбит/с
SHA-2 (512)	80	C	41,159 Мбит/с	68,701 Мбит/с
SHA-2 (256)	64	C	81,308 Мбит/с	135,557 Мбит/с
ГОСТ 34311-95	-	C+Assembler	49,408 Мбит/с	83,056 Мбит/с
HAVAL	96(128, 160)	C	337,842 Мбит/с	564,809 Мбит/с
SHA-1	80	C Assembler	206,285 Мбит/с 361,581 Мбит/с	344,433 Мбит/с 605,558 Мбит/с
RIPEMD-160	160	C	147,465 Мбит/с	246,568 Мбит/с
MD5	64	C	278,715 Мбит/с	574,635 Мбит/с
MD4	48	C	344,086 Мбит/с	467,793 Мбит/с
UMAC	-	C C+Assembler	989,371 Мбит/с 3518,900 Мбит/с	1648,953 Мбит/с 5885,057 Мбит/с
Rijndael CBC–MAC	14	C	139,376 Мбит/с	231,255 Мбит/с
ГОСТ 28147-89 (режим 4)	16	C+Assembler	189,559 Мбит/с	315,270 Мбит/с

Заключение

В качестве основных критериев оценки функций хэширования можно использовать стойкость и вычислительную сложность (скорость) вычисления значения хэш-функции.

Минимальные требования по стойкости, определенные при разработке стандарта AES, соответствуют сложности атаки, равной 2^{128}. Таким образом, для новых криптографических систем можно рекомендовать только однонаправленные хэш-функции, имеющие стойкость к коллизиям не менее 2^{128}, т.е. функции с длиной хэш-значения не менее 256 бит. Однонаправленные функции хэширования с длиной хэш-значения, равного 128 битам, следует по возможности исключать из применения, по крайней мере, в перспективных системах. Для функций выработки КАС требования несколько другие. Для будущего применения длина ключа и хэш-значения должна составлять не менее 128 бит, однако, в настоящее время возможно использование функций с длиной ключа не менее 128 бит и длиной кода аутентификации не менее 64-х бит.

Исходя из вышесказанного, алгоритм SHA-2 (512) имеет большой запас стойкости и достаточное быстродействие, т.е. может применяться в системах, в которых наиболее важным требованием является стойкость в течение длительного времени. Алгоритм Whirlpool имеет стойкость, сравнимую с SHA-2 (512), но невысокое быстродействие и может применяться в системах, где стойкость намного важнее скорости.

В качестве рекомендуемого решения для большинства криптографических систем можно рекомендовать функции хэширования с хэш-значением длиной 256 бит. В системах, для которых необходима высокая скорость работы, рекомендуется применять HAVAL, в остальных - SHA-2 (256). В Украине в качестве государственного стандарта принят алгоритм ГОСТ 34311-95, который должен использоваться в системах, подлежащих сертификации. Следует отметить, что ГОСТ 34311-95 является одним из наиболее медленных алгоритмов.

Отдельно следует рассмотреть алгоритм UMAC. Этот алгоритм поддерживает два режима работы: однонаправленная хэш-функция и функция выработки КАС. Хотя для однонаправленной функции он обладает невысокой стойкостью, но зато имеет наивысшее быстродействие, приближающееся к 6 Гбит/с на процессоре Pentium III 1000 MHz. Эта функция может быть использована для систем, где требуется очень высокое быстродействие, например, для систем защиты сетевого трафика в высокоскоростных сетях.

Список литературы: 1. *A. Menezes, P. van Oorschot, and S. Vanstone.* Handbook of Applied Cryptography. Chapter 9. CRC Press, 1996. 2. *R. Anderson.* The classification of hash functions. Proc. of the IMA Conference on Cryptography and Coding, Cirencester, December 1993, Oxford University Press, 1995, pp. 83-95. 3. *ANSI X9.30 (PART 2)*, "American National Standard for Financial Services – Public key cryptography using irreversible algorithms for the financial services industry – Part 2: The secure hash algorithm (SHA)", ASC X9 Secretariat – American Bankers Association, 1993. 4. *K. Ohta and K. Koyama.* Meet-in-the-Middle Attack on Digital Signature Schemes. In Abstract of AUSCRYPT '90, pages 110-121, 1990. 5. *B. Preneel.* Analysis and Design of Cryptographic Hash Functions. PhD thesis, Katholieke University Leuven, January 1993. 6. *J.-J. Quisquater and J.-P. Delescaille.* How Easy is Collision Search. New results and applications to DES. In Advances in Cryptology, Proceedings of CRYPTO'89, pages 408-415, 1990. 7. *E. Biham.* On the Applicability of Differential Cryptoanalysis to Hash Functions. In E.I.S.S Workshop on Cryptographic Hash Functions, pages 25-27, March 1992. 8. *E. Biham and A. Shamir.* Differential cryptoanalysis of FEAL and N-Hash. In Advances in Cryptology – Eurocrypt '91, pages 1-16, 1991. 9. *http://www.cryptonessie.org.*

Харьковский национальный университет радиоэлектроники

Поступила в редколлегию 22.04.2002