УДК 519.713

БЫСТРОДЕЙСТВУЮЩИЙ АЛГОРИТМ ХЕШИРОВАНИЯ С ВЫСОКОЙ ЗАЩИЩЕННОСТЬЮ ОТ ВСКРЫТИЯ

ШТАНЬКО И.А.

Предлагается вариант алгоритма хеширования, который обладает повышенным быстродействием, высокой защищенностью от вскрытия и длиной результата 128 и 256 бит. Алгоритм использует "технологию этажей", поэтому может состоять из различного количества циклов в зависимости от требований к стойкости и быстродействию.

1 Предпосылки разработки нового алгоритма хеширования

В современных компьютерных системах одной из наиболее актуальных задач является обеспечение целостности и подлинности информации на всем этапе ее существования. Одна из составных частей системы обеспечения целостности и подлинности информации – используемый алгоритм хеширования [1].

В настоящее время в системах защиты информации используются алгоритмы хеширования, разработанные в США и России. Наиболее часто используются алгоритмы MD4[2], MD5[3], SHA[4] и ГОСТ 34-11 94. Эти алгоритмы имеют много достоинств, однако содержат и некоторые недостатки: алгоритм ГОСТ 34-11 94 при достаточно высоких показателях криптостойкости обладает весьма невысоким быстродействием, на алгоритмы MD4 и MD5 разработаны успешные атаки, позволяющие получить коллизии за весьма незначительное время. Кроме того, всегда рекомендуется использовать алгоритмы "домашней" разработки по сравнению с зарубежными образцами.

В промышленно развитых странах в интересах обеспечения информационной безопасности создаются свои алгоритмы криптозащиты, а также алгоритмы хеширования. В США принят стандарт SHA (являющийся доработанной модификацией алгоритма MD5), в России принят ГОСТ 34-11 94.

Однако имеется вероятность "закладок" и "люков" в алгоритмах, разработанных за рубежом, что снижает общую устойчивость системы защиты информации от атак. Существует непроверенная информация, что спецслужбы при помощи имеющегося в их распоряжении оборудования могут взломать практически любую систему защиты. Такое возможно только в том случае, если существует возможность вскрытия системы защиты методом, отличающимся от полного перебора всех возможных вариантов, иными словами некоторых "закладок".

2 Требования, предъявляемые к разрабатываемому алгоритму хеширования

Прежде всего, алгоритм должен обеспечивать защиту от известных атак на алгоритмы хеширования [1], таких как:
– прямой перебор возможных вариантов выходных значений;
– атака "дня рождения" (подбор двух вариантов сообщения с одинаковым значением функции хеширования);
– построение алгоритма, позволяющего получить входное значение на основании его хеш-функции;
– выходное значение алгоритма должно зависеть от некоторых ключевых параметров для использования его в качестве КАС (кода аутентификации и сообщения).

Для обеспечения высокой защищенности алгоритма от криптоатак необходимо обеспечить ряд требований.

Во-первых, равновероятность получаемого результата. Выполнение данного условия не позволяет криптоаналитику выявлять более вероятные значения хеш-функции как для любого случайного пространства сообщений, так и для ограниченного определенными рамками.

Во-вторых, при разработке хеш-функции необходимо обеспечить невозможность получения результата хеширования иным способом, кроме как с помощью этой хеш-функции.

Изменение одного бита исходного сообщения должно повлечь изменение примерно половины бит результата хеширования. В частности, это касается некоторых видов сообщений, таких как пустое сообщение, сообщение, состоящее только из нулевых либо только из единичных бит. Кроме того, данное требование подразумевает невозможность получить значение хеширования для сообщения, незначительно отличающегося от другого, для которого оно известно. Выполнение этого требования возможно только в случае, если алгоритм обладает хорошим "лавинным" эффектом, иными словами, каждый бит выходного значения должен зависеть сложным образом от каждого бита входного сообщения.

В-третьих, невозможность построения так называемого "обратного" алгоритма, т.е. алгоритма, который позволил бы получить исходное сообщение либо его вариант, хеш-функция которого будет совпадать с неким значением.

В-четвертых, стойкость алгоритма должна базироваться на наборе применяемых преобразований, а не на его недоступности для криптоаналитика.

Кроме выполнения требований к высокой защищенности разрабатываемого алгоритма, необходимо обеспечить его высокое быстродействие.

3 Подходы, применяемые при разработке представленного алгоритма хеширования

Как указывалось выше, при разработке алгоритма необходимо обеспечить высокую скорость "лавинного" эффекта. Выполнение этого требования возможно при помощи достаточно большого количества методов [5], однако определяющим является их быстродействие.

Высокое быстродействие хеш-функции может быть обеспечено за счет использования быстрых битовых операций над блоками сообщения [1]. В числе таких преобразований следует отметить сложение по модулю 2, 2^8, 2^{16} и 2^{32}, циклический сдвиг вправо (или влево) на n бит.

Однако прямое применение таких операций неоправданно с той точки зрения, что по выходным значениям криптоаналитик сможет делать предполо-

жения об исходном сообщении. А это неприемлемо из-за перечисленных выше требований.

Таким образом, необходима функция рандомизации, которая смогла бы скрыть от криптоаналитика исходное сообщение.

Рандомизация сообщения может обеспечиваться следующими методами:

– использование специальных констант;

– использование ГПСЧ;

– использование таблиц подстановки.

Первый метод подразумевает использование таких значений, битовое представление которых содержит примерно равновероятное распределение нулей и единиц. Например, в алгоритме MD4 используются следующие константы (приведены для наглядности в шестнадцатеричном и двоичном виде):

5A827999 = 01011010100000100111100110011001,

6ED9EBA1 = 01101110110110011111010111010000 1.

Можно заметить, что в этих константах примерно одинаковое число единичных и нулевых бит (в первом случае "1" – 15, "0" – 17, во втором "1" – 19, "0" – 13). Кроме того, взаимное расположение нулей и единиц достаточно случайно.

Использование ГПСЧ возможно только в случае его высокого быстродействия. Такой генератор должен давать сигнал, каждый последующий набор значений которого достаточно отличается от предыдущего, что практически сложно реализовать.

Использование таблиц подстановок подразумевает замену одних битовых последовательностей другими. В частности, таблица подстановок используется в алгоритме ГОСТ Р 34-11. Ее применение позволяет при достаточно невысоких затратах на вычисления получить высокую степень рандомизации. Однако при этом требования к таблице замены должны быть достаточно жесткие. Кроме того, изменение таблицы замен влечет за собой и изменение выходного значения алгоритма. Таким образом, таблица замен является еще одним ключом алгоритма.

Кроме перечисленных выше методов рандомизации сообщения, возможны варианты, состоящие из некоторого множества таких методов.

4 Обобщенная схема разрабатываемого алгоритма хеширования

При разработке алгоритма использовались следующие постулаты:

– для обеспечения высокого быстродействия, как указывалось выше, необходимо использовать только простые операции, выполняемые процессором за минимальное число тактов;

– общее количество циклов обработки блока сообщения должно быть сведено к минимуму;

– алгоритм должен иметь возможность формирования результата длиной 128 и 256 бит;

– рандомизация сообщения должна занимать как можно меньше времени.

Исходя из перечисленных выше требований и постулатов, разрабатываемый алгоритм должен строиться в виде следующей последовательности преобразований.

1. Расширение исходного сообщения до длины, кратной размеру обрабатываемого алгоритмом блока данных. Установка начальных значений используемых переменных.

2. Выполнение последовательности преобразований для получившейся последовательности блоков фиксированной длины.

3. Вычисление окончательного значения возвращаемого алгоритмом, как функции от значения, полученного на последнем этапе обработки.

Обобщенная схема алгоритма, описанного выше, представлена на рис. 1.

Рис. 1. Обобщенная схема разрабатываемого алгоритма хеширования

Цикл преобразований каждого блока исходного сообщения может строиться с использованием следующих подходов:

– применение последовательности преобразований, каждое из которых используется один раз;

– применение цикла преобразований, использующих разные константы;

– применение цикла преобразований, параметры которого вычисляются отдельно для обработки каждого блока сообщения.

Следует заметить, что здесь под циклом преобразований подразумевается последовательность действий, которая, возможно, также содержит в себе подциклы.

Рассмотрев перечисленные выше подходы к построению цикла обработки, отметим значительное усложнение алгоритма в первом случае, сложность выбора констант во втором и необходимость обеспечения надежного алгоритма формирования параметров в третьем.

Исходя из сказанного выше, при разработке алгоритма был выбран подход, основанный на использовании так называемой "технологии этажей".

5 Технология этажей

Сущность такого подхода заключается в том, что алгоритм работы внутри является постоянным и не изменяется в случае изменения параметров. Это позволяет, кроме упрощения алгоритма, минимизировать объем программного кода, что в некоторых

случаях может оказаться существенным (в частности, для аппаратной реализации алгоритма).

Содержание этажей остается постоянным, их возможно использовать различное количество в зависимости от конкретных требований применения данного алгоритма. Технология этажей схематически изображена на рис. 2.

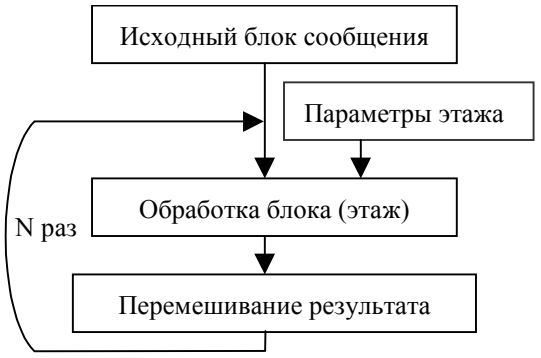


Рис. 2. Цикл обработки блока данных

6. Математическое описание алгоритма

Математическое описание работы алгоритма выглядит следующим образом:

$$M_0 = MAP_0 \oplus K; \quad (1)$$

$$h_0 = (f(M_0)) \lll 3,$$

где K – ключ рандомизации;

M_0 – начальный блок исходного сообщения;

<< – циклический сдвиг влево.

Далее преобразования производятся в соответствии с (2):

$$M_i = MAP_i \oplus K; \quad (2)$$

$$h_i = (f(M_i)) \lll 3.$$

После преобразований (1) и (2) следуют преобразования на этажах (3):

$$A_i = ((f_n(A_{i-1}\ [+]\ K_4)) \lll 3) \oplus B_{i-1};$$

$$B_i = ((f_n(B_{i-1} \oplus K_3)) \lll 11) \oplus A_i; \quad (3)$$

$$C_i = ((f_n(C_{i-1}\ [**]\ K_2)) \lll 13) \oplus D_i;$$

$$D_i = ((f_n(D_{i-1}\ [*]\ K_1)) \lll 7) \oplus C_{i-1},$$

где [+] – сложение по модулю 2^{32};

[*] – сложение по модулю 2^{16};

[**] – сложение по модулю 2^8;

$\oplus$ – сложение по модулю 2.

Формирование ключа рандомизации K производится путем использования таблицы подстановок. Она представляет собой матрицу размера 16*16 (для 128-битного варианта) и 16*32 (для 256-битного).

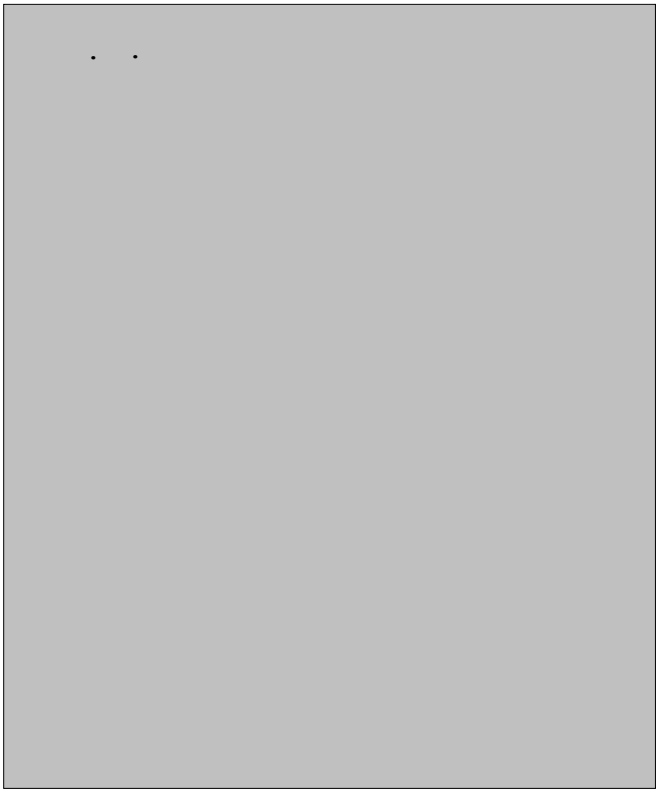
Дальнейшие преобразования заключаются в том, что подблок делится на 4 по 32 или 8 по 32 бита и эти блоки образуют этаж (параллельный этаж).

На рис. 3 изображена обобщенная схема работы алгоритма (рассмотрен случай 128-битной хеш-функции), приведен только один его этаж (выполнение остальных происходит аналогично).

7. Оценка криптостойкости предложенного алгоритма хеширования

Как указывалось выше, одним из основных требований к разрабатываемому алгоритму является его высокая криптостойкость, для обеспечения которой необходимым условием служат хорошие статистические свойства получаемого результата. В табл. 1 приведены статистические характеристики разработанного алгоритма (длина блока 128 бит) по этажам.

Для сравнения в табл. 2 приведены аналогичные характеристики для алгоритма SHA (длина блока 160 бит), который выбран не случайно. Для этого алго-



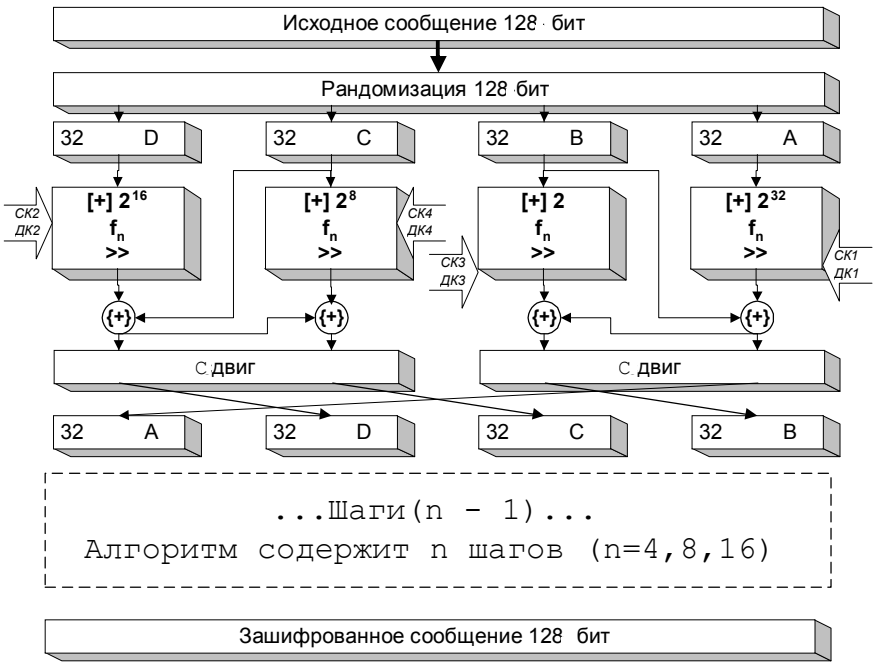


Рис.3. Схема работы алгоритма хеширования

ритма на данный момент не найдено ни одной успешной атаки, несмотря на то, что он уже давно является общедоступным.

Сравнение показывает, что статистические свойства разработанного алгоритма ничуть не уступают аналогичным свойствам алгоритма SHA.

За счет использования 4 методов битового биения (сложение по разным модулям) уменьшается число циклов преобразования и увеличивается скорость

Таблица 2

. .	Асимметр.

работы алгоритма. Как видно из табл. 1, уже на первом шаге статистические свойства алгоритма указывают на высокую степень случайности получаемого результата.

Исследования, проведенные над алгоритмом, показали, что он обладает прекрасными статистическими характеристиками. С изменением в исходном сообщении одного бита хеш-функция изменяется на половину своих бит. Это значит, что алгоритм обладает хорошим "лавинным" эффектом. Изменение ключа на один бит приводит к аналогичному результату. Алгоритм осуществляет качественное "перемешивание" бит, при этом дисперсия указывает на то, что измененные разряды распределены по равномерному закону.

Чтобы оценить криптостойкость алгоритма, необходимо вычислить $t_б$, $H(K)$, L_0 для алгоритмов с выходной хеш-функцией 128 и 256 бит: $t_б = 1,1*10^{19}$ лет для 128-битного и $t_б = 3*10^{57}$ лет для 256-битного; энтропия ключа $H(K)_{128} = 128$ и $H(K)_{256} = 256$; расстояние единственности $L_0 = 256$ и $L_0 = 512$.

8. Сравнительный анализ алгоритма с известными аналогами

Высокая скорость "лавинного" эффекта предложенного алгоритма позволила уменьшить общее число циклов обработки блока данных без ущерба его криптостойкости. Это, в свою очередь, положительно сказалось на быстродействии всего алгоритма.

Сравнение представленного алгоритма с известными аналогами показывает более высокое его быстродействие (см. табл. 3) при хороших статистических характеристиках.

Таблица 3

, .	Мбайт/сек

Исходя из сказанного выше, можно сделать вывод о том, что предложенный алгоритм обладает хорошими показателями быстродействия и криптостойкости. Таким образом, разработанная функция хеширования может использоваться в системах защиты информации.

Кроме того, предложенный подход к реализации криптографических преобразований может быть использован и при разработке других алгоритмов защиты информации, в частности, алгоритмов блочного шифрования.

Литература: 1. *Горбенко И.Д., Штанько И.А.* Функции хеширования. Понятие, требования, классификация, свойства и применение // Радиоэлектроника и информатика. 1998. №1. С. 64-69. 2. *Rivest R.L.* The MD4 Message Digest Algorithm. Advances in Cryptology CRYPTO'90 Proceedings. Apr. 1992. P. 30-35. 3. *Rivest R.L.* The MD5 Message Digest Algorithm. Advances in Cryptology CRYPTO'90 Proceedings. Oct . 1990. P. 40-44. 4. FIPS 180-1 Secure hash standard. NIST, US Department of Commerce. Washington D.C. Apr. 1995. P. 25. 5. *Preneel B.* Analysis and Design of Cryptographic Hash Function. Ph.D. dissertation. Katholieke Universiteit Leuven. Jan. 1993. P. 124.

Поступила в редколлегию 30.09.98

Рецензент: д-р техн. наук Стасев Ю.В.

Штанько Игорь Анатольевич, аспирант кафедры ПОЭВМ ХТУРЭ. Научные интересы: защита информации. Увлечения и хобби: программирование. Адрес: Украина, 310726, Харьков, пр. Ленина, 14. тел. 63-95-33.