УДК 681.3.06

О МОДИФИКАЦИИ АЛГОРИТМА MD5

ABOUT MODIFICATION OF THE MD5 ALGORITM

А.Д. Буханцов, И.В. Дружкова
A.D. Bukhantsov, I.V. Druzhkova

Белгородский государственный национальный исследовательский университет, Россия, 308015, Белгород, ул. Победы, 85

Belgorod State National Research University, 85 Pobeda St, Belgorod, 308015, Russia

e-mail: bukhantsov@bsu.edu.ru, 984546@bsu.edu.ru

Аннотация. В данной статье предлагается вариант модификации известной хэш-функции с целью повышения ее стойкости. Показано, что при сохранении требований по быстродействию вычисления дайджеста сообщения на основе модифицированного алгоритма, вероятность коллизии второго рода существенно снижается.

Resume. This article proposes a modification of the well-known hash functions with the aim of increasing its durability.Shows that while maintaining the performance requirements of the message digest calculation based on modified algorithm the probability of collision of the second kind is greatly reduced.

Ключевые слова: хэш-функция, вероятность коллизии, атака, парадокс дней рождения.
Keywords: the hash function, the probability of collisions, attack, birthday paradox.

В настоящее время вопросы исследования путей дальнейшего снижения уязвимости криптографических хэш-функций для различных приложений достаточно актуальны, так как при постоянном возрастании мощности и быстродействия современных вычислительных средств вероятность успешной атаки также возрастает. Алгоритмы хеширования, используемые в современных стандартах на электронную подпись в других смежных приложениях [1-3], имеют достаточно высокую стойкость к коллизиям, однако усложнение алгоритма часто приводит к увеличению вычислительной сложности и затрат на его реализацию. Поэтому относительно простые хэш-алгоритмы, используемые в приложениях, не требующих высокого уровня стойкости, могут оставаться практически полезными. Например, хэш-функции MD5 и SHA-1 до сих пор продолжают использоваться в некоторых практических приложениях, несмотря на обнаруженные уязвимости, так как обладают высоким быстродействием. Таким образом, исследование возможности модификации таких криптографических протоколов с целью повышения их стойкости по-прежнему может представлять не только научный, но и практический интерес.

В данной работе предлагается вариант модификации алгоритма MD5. Если в существующем протоколе выходная хэш-сумма составляет 128 бит, то предлагаемый алгоритм хэширует текст произвольной длины, превращая его в 256-битную последовательность. На рисунке 1 показана блок-схема модифицированного алгоритма.

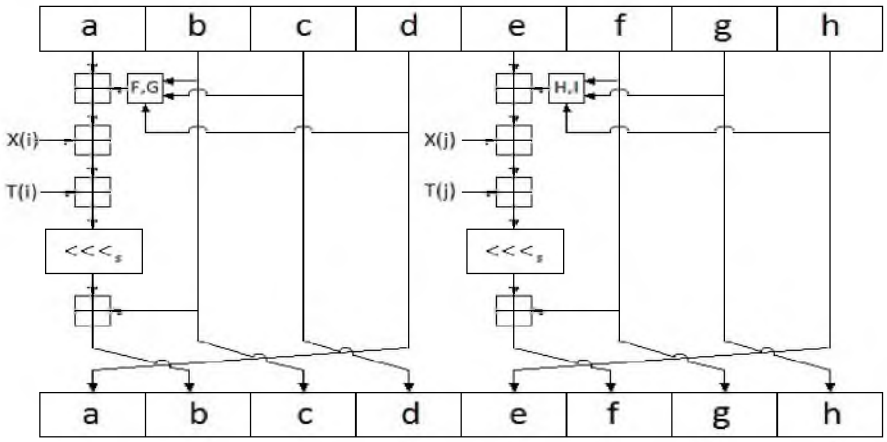

Рис. 1. Блок-схема измененного алгоритма MD5
Fig. 1. A block diagram of the modified algorithm MD5

Описание модифицированного алгоритмаMD5

Шаг 1. Сначала добавляют единичный байт в конец потока. Затем входные данные выравниваются так, чтобы их новый размер L был сравним с 448 по модулю 512. Выравнивание происходит с помощью дописывания необходимого числа нулевых бит.

Шаг 2. В конец сообщения добавляют 64-битное представление длины данных до выравнивания. Сначала записывают младшие 4 байта, затем старшие. Если длина данных превосходит 2^{64}-1, то дописывают только младшие биты. В результате длина потока кратна 512.

Шаг 3. Для вычислений инициализируются 4 переменных размером по 32 бита и задаются начальные значения десятичными числами:

a=1732584193;
b=4023233417;
c=2562383102;
d=271733878;
e=1374568893;
f=2578663458;
g=741235123;
h=8012563941.

В этих переменных будут храниться результаты промежуточных вычислений. Начальное состояние “abcdefgh” называется инициализирующим вектором.

Шаг 4. Потребуются 4 функции для четырех раундов. Введем функции от трех параметров:

$$F(X,Y,Z) = (X \wedge Y) \vee (\neg X \wedge Z) \quad (1)$$

$$G(X,Y,Z) = (X \wedge Z) \vee (\neg Z \wedge Y) \quad (2)$$

$$H(X,Y,Z) = X \oplus Y \oplus Z \quad (3)$$

$$I(X,Y,Z) = Y \oplus (\neg Z \vee X) \quad (4)$$

Шаг 5. Определим таблицу констант t — 64-элементная таблица данных, построенная следующим образом:

$$t(i) = \text{int}\left(2^{32} \cdot |\sin(i)|\right), \quad (5)$$

где $i = 1,\ldots,6$.

Шаг 6. Последовательность бит полученная после всех выравниваний представляется в виде n-блоков по 16 бит. Каждый блок x (16 бит) проходит 4 раунда преобразований.

Вычисление происходит в цикле от 1 до n. В каждом цикле преобразования происходят над i-тым блоком. Сохраняются переменные a, b, c, d, e, f, g, h, оставшиеся после операций над предыдущими блоками (или их начальные значения, если блок первый). Далее происходят преобразования с помощью следующих 4 раундов:

Раунд 1.

[abcdksi] a = b + ((a + F(b,c,d) + x[k] + t[i]) <<<s)
[abcd 0 7 1][dabc 1 12 2][cdab 2 17 3][bcda 3 22 4]
[abcd 4 7 5][dabc 5 12 6][cdab 6 17 7][bcda 7 22 8]
[abcd 8 7 9][dabc 9 12 10][cdab 10 17 11][bcda 11 22 12]
[abcd 12 7 13][dabc 13 12 14][cdab 14 17 15][bcda 15 22 16]

Раунд 2.

[abcd k s i] a = b + ((a + G(b,c,d) + x[k] + i[i]) <<< s)
[abcd 1 5 17][dabc 6 9 18][cdab 11 14 19][bcda 0 20 20]
[abcd 5 5 21][dabc 10 9 22][cdab 15 14 23][bcda 4 20 24]
[abcd 9 5 25][dabc 14 9 26][cdab 3 14 27][bcda 8 20 28]
[abcd 13 5 29][dabc 2 9 30][cdab 7 14 31][bcda 12 20 32]

Раунд 3.

[efgh k s i] e = f + ((e + H(f,g,h) + x[k] + t[i]) <<< s)
[efgh5 4 33][hefg 8 11 34][ghef 11 16 35][fgha 14 23 36]
[efgh 1 4 37][hefg 4 11 38][ghef 7 16 39][fgha 10 23 40]
[efgh 13 4 41][hefg 0 11 42][ghef 3 16 43][fgha 6 23 44]
[efgh 9 4 45][hefg 12 11 46][ghef 15 16 47][fgha 2 23 48]

Раунд 4.

[efgh k s i] e = f + ((e + H(f,g,h) + x[k] + t[i]) <<< s)
[efgh 0 6 49][hefg 7 10 50][ghef 14 15 51][fgha 5 21 52]
[efgh12 6 53][hefg 3 10 54][ghef 10 15 55][fgha 1 21 56]
[efgh 8 6 57][hefg 15 10 58][ghef 6 15 59][fgha 13 21 60]

[efgh 4 6 61][hefg 11 10 62][ghef 2 15 63][fgha 9 21 64]

Затем переменные суммируются с результатом предыдущего цикла.

После окончания цикла необходимо проверить, есть ли еще блоки для вычислений. Если да, то переходим к следующему блоку и повторяем цикл.

Алгоритм реализован на языке C++ с использованием прикладной программы Microsoft Visual Studio.

Вычислительные эксперименты

При тестировании программного кода было показано, что модифицированный алгоритм имеет тот же порядок вычислительной сложности, что и стандартный, то есть время вычисления хэш-суммы одного и того же текста с помощью обоих алгоритмов практически не отличается (таблица 1). Это является важным фактором, так как основным преимуществом хэш-функций такого вида и является быстродействие.

Таблица 1
Table 1

Время вычисления хэша
A time hash calculation

Количество входных данных (количество символов)	Время вычисления стандартного MD5 на компьютере с частотой 2ГТЦ (мс)	Время вычисления Модифицированного MD5 на компьютере с частотой 2ГТЦ (мс)
10	0.038	0.041
50	0.045	0.044

Также хэш-функция должна удовлетворять такому условию, что при изменении одного бита достигается «лавинный эффект», то есть хэш значительно различается для хешируемых входных данных. Пример сравнения таких сообщений и соответствующих сверток приведен в таблице 2.

Таблица 2
Table 2

Выходные данные модифицированного MD5
Output data of changed MD5

Входной текст	Полученная хэш
hello	cbd02904ff8ad1f3dee3c92bc2418c2f7b214cc3769299558fb5a423f74b2bb2
iello	e39b1e6cd4f6e7e18103e0cae8eb667ec4a3bdc4970887048e6c45209d4814cb

В повышении стойкости MD5 после процедуры модификации можно убедиться, рассмотрев основные атаки на хэш-функции и сравнив основной и усовершенствованный алгоритмы.

Атака «дней рождения» [4] – один из методов поиска коллизий хэш-функций на основе парадокса дней рождения. Для заданной криптографической хэш-функции f-целью атаки является поиск коллизии второго рода. Для этого вычисляются значения f для случайно выбранных блоков входных данных до тех пор, пока не будут найдены два блока, имеющие один и тот же хэш. Таким образом, если f имеет N различных равновероятных выходных значений и N является достаточно большим, то из парадокса дней рождения следует, что в среднем после перебора $1{,}25 \cdot \sqrt{N}$ различных входных значений будет найдена искомая коллизия. Если же хэш-функция генерирует n-битное значение, то число случайных входных данных, для которых хэш-коды с большой вероятностью дадут коллизию, равно не 2^n, а только около $2^{\frac{n}{2}}$.

Таким образом, для MD5 разной размерности хэша существует разное количество входных данных для сопоставимой вероятности коллизии (таблица 3).

Таблица 3
Table 3

Вероятность коллизии
The probability of collision

Размерность хэш (n)	Количество равновероятных выходных значений (2n)	Количество входных данных, при которых коллизия будет с заданной вероятностью				
		10-6 %	0,1 %	1 %	25 %	50 %
128	3.4×1038	2.6×1016	8.3×1017	2.6×1018	1.4×1019	2.2×1019
256	1.2×1077	4.8×1035	1.5×1037	4.8×1037	2.6×1038	4.0×1038

Атака «грубой силой» [5] может быть выполнена для нахождения первого прообраза по заданному хэш-значению или для нахождения второго прообраза, дающего такое же хэш-значение, как и заданное сообщение. Суть атаки заключается в последовательном или случайном переборе входных сообщений и сравнении результата выполнения хэш-функции с ее заданным значением. Сложность такой атаки оценивается 2^{n-1} операций вычисления хэш-значения, где n–длина хэш-значения. Из табл. 3 видно, что модифицированный алгоритм требует в $3{,}5 \cdot 10^{38}$ раз больше операций, чем стандартный.

Выводы

Предложенный вариант модификации алгоритма MD5 позволяет фактически при том же быстродействии существенно снизить вероятность коллизии как первого, так и второго рода. Использование таких алгоритмов, например, в приложениях ускоренного поиска данных в больших массивах, где коллизия приводит не к нарушению целостности сообщения, а всего лишь к ошибке поиска, может значительно повысить эффективность работы таких систем за счет снижения вероятности такой ошибки.

Список литературы
References

1. ГОСТ Р 34.11-2012. Информационная технология. Криптографическая защита информации. Функция хэширования.

GOST R 34.11-2012. Informacionnaja tehnologija. Kriptograficheskaja zashhita informacii. Funkcija hjeshirovanija.

2. Халимов Г.З. Универсальное хеширование по максимальной кривой третьего рода / Научные ведомости БелГУ. Сер. История. Политология. Экономика. Информатика. - 2011. - № 1 (96), - Вып. 17/1. - С. 137–145.

Halimov G.Z. Universal'noe heshirovanie po maksimal'noj krivoj tret'ego roda / Nauchnye vedomosti BelGU. Ser. Istorija. Politologija. Jekonomika. Informatika. - 2011. - № 1 (96), - Vyp. 17/1. - S. 137–145.

3. Халимов О.Г., Буханцов А.Д., Халимов Г.З. Построение кривых Гурвица для универсального хеширования/ Научные ведомости БелГУ. Сер. История. Политология. Экономика. Информатика. - 2014. - № 1 (172), - Вып. 29/1. - С. 153–160.

Halimov O.G., Buhancov A.D., Halimov G.Z. Postroenie krivyh Gurvica dlja universal'nogo heshirovanija/ Nauchnye vedomosti BelGU. Ser. Istorija. Politologija. Jekonomika. Informatika. - 2014. - № 1 (172), - Vyp. 29/1. - S. 153–160.

4. K. Ohta and K. Koyama. Meet-in-the-Middle Attack on Digital Signature Schemes. In Abstract of AUSCRYPT'90, pages 110-121, 1990.

K. Ohta and K. Koyama. Meet-in-the-Middle Attack on Digital Signature Schemes. In Abstract of AUSCRYPT'90, pages 110-121, 1990.

5. 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.

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.