5. *Лысов М.С., Стариков А.В., Стариков В.А.* Линеаризованная математическая модель синхронного электродвигателя при различных способах управления его скоростью // Вестн. сам. гос. техн. ун-та. Сер.: физ.-мат. науки. – 2008. – №1 (16). – С. 102-107.
6. *Florent Morel.* Permanent Magnet Synchronous Machine Hybrid Torque Control, Jean-Marie Retid, Xuefang Lin-Shi, and Clarie Valentin // IEEE Transactions on Industrial Electronics. – 2008. – Vol. 55, № 2.
7. *Estima J.O., Marques Cardoso A.J.* The Occurrence of Faults in Permanent Magnet Synchronous Motor Drives and its Effects on the Power Supply Quality // University of Coimbra, FCTUC/IT.

Статью рекомендовал к опубликованию д.т.н., профессор В.Х. Пшихопов.

Бошляков Андрей Анатольевич – МГТУ им. Н.Э. Баумана; e-mail: boshlyakov@mail.ru; 105005, г. Москва ул. 2-я Бауманская, 5; тел.: 84992636778; кафедра специальной робототехники и мехатроники; доцент.

Ковалев Владимир Вячеславович – кафедра специальной робототехники и мехатроники; аспирант.

Рубцов Василий Иванович – кафедра специальной робототехники и мехатроники; доцент.

Boshlyakov Andrey Anatolevich – BMSTU; e-mail: boshlyakov@mail.ru; 5, 2-ya Baumanskaya street, Moscow, 105005, Russia; phone: +74992636778; the department of robotics and mechatronics; associate professor.

Kovalev Vladimir Vyacheslavovich – the department of robotics and mechatronics; postgraduate student.

Rubtsov Vasily Ivanovich – the department of robotics and mechatronics; associate professor.

УДК 51-77

К.Д. Майзаков, Д.А. Эдель, В.А. Новосядлый

АЛГОРИТМИЧЕСКАЯ ОПТИМИЗАЦИЯ ВЫЧИСЛЕНИЯ ПРООБРАЗОВ НЕОБРАТИМЫХ ХЭШ-ФУНКЦИЙ MD5 И MD4

Рассматривается способ эффективного вычисления прообразов хэш-функции MD5 [1] и MD4 [2]. Обзор существующих способов атак – поиск коллизий, полный перебор – приведен в [3]. Рассмотрены существующие программные реализации [4, 5]. Показано, что данные реализации имеют возможность вычислять прообразы длиной до 16 байт и являются частным случаем предлагаемого. Приведены результаты численных экспериментов, показывающие, что предлагаемый способ в случае хэш-функции MD5 имеет возможность вычислять прообразы длиной до 55 байт и имеет скорость вычислений в среднем на 15 % выше существующих.

Хэш-функции; MD5; MD4; вычисление прообразов.

K.D. Mayzakov, D.A. Edel, V.A. Novosiadliy

ALGORITHMIC OPTIMIZATION OF INVERSE IMAGE COMPUTATION FOR MD5 AND MD4 DIGEST ALGORITHMS

Current paper reports on the method of efficient inverse image computation for MD5 [1] and MD4 [2] digest algorithm. A review of existing attack methods is given in [3]. Existing software implementations [4, 5] are discussed. They are shown to be able to compute inverse image of up to 16 bytes length and implement a special case of proposed method. The results of computational experiments are given, which show the possibility of inverse image calculation of up to 55 bytes length, with computational speed better than existing implementations up to 15 % in average.

Digest algorithm; MD5; MD4; inverse image computation.

Введение. В настоящее время однонаправленные хэш-функции получили большое распространение для обеспечения конфиденциальности пользовательской аутентификационной информации в среде сети Интернет, благодаря возможности хранить проверяющей стороной не сам пользовательский пароль, а результат хэширующего преобразования от данного пароля. Таким образом, при запросе пользовательского пароля, сравниваются не парольные фразы, а результаты хэширующих преобразований.

Одной из наиболее распространенных подобных функций является хэш-функция MD5 [1]. Данный алгоритм применяется для обеспечения конфиденциальности пользовательской аутентификационной информации в системах обмена сообщениями (ICQ, Skype), Интернет-форумах.

Существуют успешные атаки на алгоритм MD5, связанные с нахождением коллизий в бинарных документах [3, 6], но данные атаки сложно применимы к задаче восстановления парольной информации.

В данной статье рассмотрен способ ускорения вычисления прообразов хэш-функций MD5 и MD4 методом полного перебора ("грубой силы") [7], основанного на уменьшении числа итераций, требуемых для принятия решения о соответствии прообраза данному хэшу. Метод полного перебора подразумевает обход пространства прообразов и вычисления соответствующих хэшей для каждого кандидата в прообраз.

Алгоритм хэширования MD5. Рассмотрим упрощенную схему хэширования MD5. Преобразования выполняются последовательно над 16 переменными типа "двойное слово", из которых первые 14 содержат байты хэшируемых данных, а две последние – размер данных в битах.

Ограничим область рассматриваемых прообразов таким образом, чтобы все вычисления можно было выполнить за единственный проход алгоритма MD5. Следовательно, максимальная длина восстанавливаемых данных составляет 55 байт, принимая во внимание наличие дополнительного байта выравнивания (0x80) и восемь байт, содержащих длину хэшируемых данных.

Рассмотрим развернутый основной цикл алгоритма MD5 [1], состоящий из четырех раундов.

Пусть F, G, H, I – функции раундов, A, B, C, D – переменные, до цикла инициализируемые заданными константами, а после – содержащие MD5 хэш, T [1 … 64] – таблица констант, используемых при вычислении, w[0 … 15] – массив хэшируемых данных, <<< – операция циклического сдвига.

На первом раунде пусть [*abcd k s i*] означает следующую операцию: $a = b + ((a + F(b,c,d) + w[k] + T[i]) <<< s)$. Произведем следующие 16 операций, которые пронумеруем от 0 до 15:

0. [ABCD 0 7 0],
1. [DABC 1 12 1],
2. [CDAB 2 17 2],
3. [BCDA 3 22 3],
4. [ABCD 4 7 4],
5. [DABC 5 12 5],
6. [CDAB 6 17 6],
7. [BCDA 7 22 7],
8. [ABCD 8 7 8],

9. [DABC 9 12 9],

10. [CDAB 10 17 10],

11. [BCDA 11 22 11],

12. [ABCD 12 7 12],

13. [DABC 13 12 13],

14. [CDAB 14 17 14],

15. [BCDA 15 22 15].

На втором раунде пусть [*abcd k s i*] означает следующую операцию: $a = b + ((a + G(b,c,d) + w[k] + T[i]) <<< s)$. Произведем следующие 16 операций, которые пронумеруем от 16 до 31:

16. [ABCD 1 5 16],

17. [DABC 6 9 17],

18. [CDAB 11 14 18],

19. [BCDA 0 20 19],

20. [ABCD 5 5 20],

21. [DABC 10 9 21],

22. [CDAB 15 14 22],

23. [BCDA 4 20 23],

24. [ABCD 9 5 24],

25. [DABC 14 9 25],

26. [CDAB 3 14 26],

27. [BCDA 8 20 27],

28. [ABCD 13 5 28],

29. [DABC 2 9 29],

30. [CDAB 7 14 30],

31. [BCDA 12 20 31].

На третьем раунде пусть [*abcd k s i*] означает следующую операцию: $a = b + ((a + H(b,c,d) + w[k] + T[i]) <<< s)$. Произведем следующие 16 операций, которые пронумеруем от 32 до 47:

32. [ABCD 5 4 32],

33. [DABC 8 11 33],

34. [CDAB 11 16 34],

35. [BCDA 14 23 35],

36. [ABCD 1 4 36],

37. [DABC 4 11 37],

38. [CDAB 7 16 38],

39. [BCDA 10 23 39],

40. [ABCD 13 4 40],

41. [DABC 0 11 41],

42. [CDAB 3 16 42],

43. [BCDA 6 23 43],

44. [ABCD 9 4 44],

45. [DABC 12 11 45],

46. [CDAB 15 16 46],

47. [BCDA 2 23 47].

На четвертом раунде пусть [*abcd k s i*] означает следующую операцию: $a = b + ((a + I(b,c,d) + w[k] + T[i]) <<< s)$. Произведем следующие 16 операций, которые пронумеруем от 48 до 63:

48. [ABCD 0 6 48],

49. [DABC 7 10 49],

50. [CDAB 14 15 50],

51. [BCDA 5 21 51],

52. [ABCD 12 6 52],

53. [DABC 3 10 53],

54. [CDAB 10 15 54],

55. [BCDA 1 21 55],

56. [ABCD 8 6 56],

57. [DABC 15 10 57],

58. [CDAB 6 15 58],

59. [BCDA 13 21 59],

60. [ABCD 4 6 60],

61. [DABC 11 10 61],

62. [CDAB 2 15 62],

63. [BCDA 9 21 63].

Можно заметить, что на четвертом раунде для шагов с номерами 49–63 не происходит обращения к первым четырем байтам хэшируемых данных (элемент $w[0]$ массива данных). Отсюда вытекает идея ускорения опробования прообразов хэш-функции MD5.

Ускорение опробования прообразов хэш-функции MD5. Пусть четверка чисел $M_h = \{a_h, b_h, c_h, d_h\}$ есть представление MD5 хэша, для которого необходимо вычислить прообраз, в виде четырех четырехбайтовых чисел. Последовательно применим к данному набору преобразования, обратные к преобразованиям четвертого раунда главного цикла алгоритма MD5 с номерами 49–63.

Полученную четверку чисел $M_p = \{a_p, b_p, c_p, d_p\}$ назовем предхэшем.

Пространство перебора будем обходить, очевидно, последовательно изменяя *w*[0]. Для каждого кандидата в прообраз производим шаги с номерами 0–47 (первых трех раундов полностью, и первый шаг четвертого раунда) и получаем четверку чисел $M_t = \{a_t, b_t, c_t, d_t\}$. Если $M_p = M_t$, то, выполнив дополнительно шаги от номера 49 до номера 63, проверяем, совпадает ли полученный хэш с заданным.

Таким образом, в результате данной оптимизации сокращается количество шагов, требуемое для проверки кандидата в прообраз.

Частный случай данного подхода реализован в ПО BarsWF [4], Elcomsoft Lightning Hash Cracker [5]. В данных программных реализациях возможно восстановление прообразов длиной до 15 байт, поскольку в основном цикле алгоритма хэширования значения массива данных заменены нулями, начиная с номера 4. Также, поскольку в данных реализациях изменяются два первых байта из четырех возможных в величине w[0], скорость опробования данных программных решений меньше теоретически возможной.

Алгоритм хэширования MD4. Покажем, что рассуждения, приведенные для алгоритма хэширования MD5, применимы также для более раннего алгоритма хэширования – алгоритма MD4. Рассмотрим алгоритм MD4 как упрощенную версию алгоритма MD5. Алгоритм MD4 также, как и алгоритм MD5, состоит из четырех раундов.

Пусть *F*, *G*, *H* – функции раундов, *A*, *B*, *C*, *D* – переменные, до цикла инициализируемые заданными константами, а после – содержащие MD4 хэш, *w*[0 .. 15] – массив хэшируемых данных.

Развернутый основной цикл алгоритма можно представить следующим образом [2]:

На первом раунде пусть [*abcd w s*] означает следующую операцию: $a = (a + F(b,c,d) + X[k]) <<< s$. Произведем 16 следующих операций:

0. [ABCD 0 3],
1. [DABC 1 7],
2. [CDAB 2 11],
3. [BCDA 3 19],
4. [ABCD 4 3],
5. [DABC 5 7],
6. [CDAB 6 11],
7. [BCDA 7 19],
8. [ABCD 8 3],
9. [DABC 9 7],
10. [CDAB 10 11],
11. [BCDA 11 19],
12. [ABCD 12 3],
13. [DABC 13 7],
14. [CDAB 14 11],
15. [BCDA 15 19].

На втором раунде пусть [*abcd w s*] означает следующую операцию: $a = (a + G(b,c,d) + w[k] + 5A827999) <<< s$. Производим 16 следующих операций:

16. [ABCD 0 3],
17. [DABC 4 5],
18. [CDAB 8 9],
19. [BCDA 12 13],
20. [ABCD 1 3],
21. [DABC 5 5],
22. [CDAB 9 9],
23. [BCDA 13 13],
24. [ABCD 2 3],
25. [DABC 6 5],
26. [CDAB 10 9],
27. [BCDA 14 13],
28. [ABCD 3 3],
29. [DABC 7 5],
30. [CDAB 11 9],
31. [BCDA 15 13].

На третьем раунде пусть [*abcd w s*] означает следующую операцию: $a = (a + H(b,c,d) + w[k] + 6ED9EBA1) <<< s$. Производим 16 следующих операций:

32. [ABCD 0 3],
33. [DABC 8 9],
34. [CDAB 4 11],
35. [BCDA 12 15],
36. [ABCD 2 3],
37. [DABC 10 9],
38. [CDAB 6 11],
39. [BCDA 14 15],
40. [ABCD 1 3],
41. [DABC 9 9],
42. [CDAB 5 11],
43. [BCDA 13 15],
44. [ABCD 3 3],
45. [DABC 11 9],
46. [CDAB 7 11],
47. [BCDA 15 15].

Таким образом, начиная с шага номер 33 хэширующих преобразований элемент $w[0]$ не участвует в вычислениях. Следовательно, к алгоритму MD4 применимы те же рассуждения, что и к алгоритму хэширования MD5.

Результаты экспериментальных вычислений. Исследования проводились на тестовом стенде Intel Core i7 3GHz 4C, 2 x Nvidia GTX460, 8GB RAM. Экспериментальный модуль реализован с поддержкой вычислений на видеоускорителях Nvidia посредством технологии Nvidia CUDA. Сравнение производилось с существующими реализациями ПО BarsWF [4], Elcomsoft Lightning Hash Cracker [5], который также поддерживают вычисления на видеоускорителях Nvidia. Следует отметить, что как ПО BarsWF, так и Elcomsoft Lightning Hash Cracker поддерживают размер кандидатов в прообразы только до 16 байт, что не позволяет использовать их, например, для восстановления паролей к CMS Joomla, которые вычисляются с использованием соли длиной 32 байта.

С целью установления практических результатов увеличения скорости переборного восстановления прообразов был проведен ряд инструментов по сравнению существующего программного обеспечения с экспериментальным программным модулем, реализующим описываемый в статье алгоритм.

Результаты численных экспериментов приведены в табл. 1.

Таблица 1

Программное обеспечение	Скорость опробования, миллионов кандидатов в прообразы в секунду
BarsWF	1397
Lightning Hash Cracker	560
Экспериментальный модуль	1547

Экспериментальный модуль обеспечивает прирост в скорости апробации кандидатов в прообразы 10 %, при этом максимальная длина кандидата в прообразы составляет 55 байт.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Rivest R.* MD5 Algorithm. Network Working Group, Request for Comments: 1321. [Электронный ресурс] // MIT Laboratory for Computer Science and RSA Data Security, Inc., 1992 / Режим доступа: http://tools.ietf.org/html/rfc1321.
2. *Rivest R.* MD4 Algorithm. Network Working Group, Request for Comments: 1320. [Электронный ресурс] // MIT Laboratory for Computer Science and RSA Data Security, Inc., 1992 / Режим доступа: http://tools.ietf.org/html/rfc1320.
3. *Sasaki A., Aoki K.* Finding Preimages in Full MD5 Faster Than Exhaustive Search // EUROCRYPT 2009, LNCS 5479. – 2009. – P. 134-152.
4. *Сваричевский М.А.* Программное обеспечение Bars WF [Электронный ресурс] // Сваричевский М.А. 2010. Режим доступа: http://3.14.by/ru/.
5. Lightning Hash Cracker [Электронный ресурс] / ООО "Элкомсофт". 2010. Режим доступа: http://www.elcomsoft.ru/lhc.html.
6. *Sotirov Alexander, Marc Stevens, Jacob Appelbaum, Arjen Lenstra, David Molnar, Dag Arne Osvik, Benne de Weger*. MD5 considered harmful today // http://www.win.tue.nl/hashclash/rogue-ca.
7. *Adleman LM, Rothemund PW, Roweis S, Winfree E.* On applying molecular computation to the data encryption standard // J. Comput. Biol. – 1999 Spring. – Vol. 6, № 1. – P. 53-63.

Статью рекомендовала к опубликованию д.ф.-м.н., профессор Г.В. Муратова.

Майзаков Константин Дмитриевич – Федеральное государственное научное учреждение “Научно-исследовательский институт «Специализированные вычислительные устройства защиты и автоматика»” в г. Ростов-на-Дону; e-mail: k.mayzakov@niisva.org; 344002, г. Ростов-на-Дону, пер. Газетный, 51; тел.: 88632012817; научный сотрудник.

Эдель Дмитрий Александрович – e-mail: d.edel@niisva.org; научный сотрудник.

Новосядлый Василий Александрович – e-mail: v.novosiadliy@niisva.org; зав. лабораторией.

Mayzakov Konstantin Dmitrievich – Federal State-Owned Science Agency “Research Institute «Specvuzavtomatika»”; e-mail: k.mayzakov@niisva.org; 51, Gazetnij, 344002, Rostov-on-Don, Russia; phone: +78632012817; scientist.

Edel Dmitry Alexandrovich – e-mail: d.edel@niisva.org; scientist.

Novosiadliy Vasily Alexandrovich – e-mail: v.novosiadliy@niisva.org; head the laboratory.