В. Ф. Антонов [V. F. Antonov]

УДК 004.9

АНАЛИЗ МЕТОДОВ ХЕШИРОВАНИЯ В ИНФОРМАЦИОННЫХ СИСТЕМАХ

ANALYSIS OF HASHING METHODS IN INFORMATION SYSTEMS

ФГАОУ ВО «Северо-Кавказский федеральный университет», Институт сервиса, туризма и дизайна (филиал) СКФУ в г. Пятигорске, Россия, e-mail: antonovpgtu@mail.ru
North Caucasus Federal University, Institute of service, tourism and design (branch) of NCFU in Pyatigorsk, Russia, e-mail: antonovpgtu@mail.ru

Аннотация. *Данная статья посвящена анализу современных методов хеширования и поиска коллизии, а также области и направления их применения в процедурах хеширования при верификации данных. В статье также рассматриваются использование различных алгоритмов MD5, SHA, ГОСТ Р 34.11-2012 при хешировании, их достоинства и недостатки.*

Материалы и методы. *При работе над статье были использованы язык программирования C#, современные методы и алгоритмы хеширования, поиска коллизии. В статье были использованы методы оценки хеш-функции на криптостойкость современных методов хеширования, при различных методах атак. Разработано приложение для оценки качества хеш-функции на языке программирования C#.*

Результаты. *Разработанное приложение дает возможность оценить качество хеш-функции при различных методах атак, с использованием методов семейства MD и SHA, а также российского стандарта ГОСТ Р 34.11-2012. Таким образом, проведенные исследования показали, что алгоритмы вычисления хеш-функции обеспечивают надежную защиту от атак.*

Заключение. *Необходимо отметить, тот факт, что ежегодно растут вычислительные мощности компьютерных систем, которые могут быть использованы злоумышленниками для поиска коллизии и впоследствии использование этих данных в корыстных целях и поиск более эффективных алгоритмов вычисления хеш-функции является актуальной задачей.*

Ключевые слова: хеширование, хеш-функция, атака, коллизии, хеш-таблица, элементарные преобразования.

Abstracts. *This article is devoted to the analysis of modern methods of hashing and collision search, as well as the scope and direction of their application in hashing procedures for data verification. The article also discusses the use of various algorithms MD5, SHA, GOST R 34.11-2012 for hashing, their advantages and disadvantages.*

Materials and methods. *When working on the article, the C # programming language modern methods and algorithms for hashing, collision search was used. The article used methods for evaluating the hash function on the cryptographic strength of modern hashing methods, with various attack methods. An application for evaluating the quality of hash functions in the C # programming language has been developed.*

Results. *The developed application makes it possible to evaluate the quality of the hash function for various attack methods, using the methods of the MD and SHA family, as well as the Russian standard GOST R 34.11-2012. Thus, the studies showed that hash function calculation algorithms provide reliable protection against attacks.*

Conclusion. *It should be noted that the fact that the computing power of computer systems is growing every year that can be used by attackers to search for collisions and subsequently using this data for personal gain and finding more efficient hash function calculation algorithms is an urgent task.*

Key words: hashing, hash function, attack, collisions, hash table, elementary transformations.

Введение. Хеширование (англ. hashing), это процедура преобразования входного сообщения, имеющая произвольный размер в выходную битовую последовательность, которая имеет фиксированную длину таким образом, что выходная последовательность никоим образом, не может быть использовано для определения входной последовательности.

Материалы и методы. Математически можно представить следующим образом, пусть некоторая последовательность входной информации x, имеющая произвольную длину, путем элементарных преобразований h этой произвольной последовательности в информационную последовательность h_x, имеющую фиксированную длину 64, 128 или 256 бит, при этом исходная последовательность x может быть любого размера. Данное преобразование h называется хеш-функцией, а результат преобразования - хеш-кодом, хеш-таблицей или дайджестом сообщения (англ. message digest). При этом дайджест используется для контроля целостности исходного сообщения и в ка-

честве контрольной суммы. Хеш-таблица формируется хеш-функией в определенном порядке, которая представляет с собой массив, хранящая пару вида «ключ-значение». Большинство алгоритмов при вычислении ключа в хеш-таблице ячейка оказывается заполненными, такую ситуацию принято называть коллилией. На рис. 1 представлен пример хеш-таблицы с коллизией.

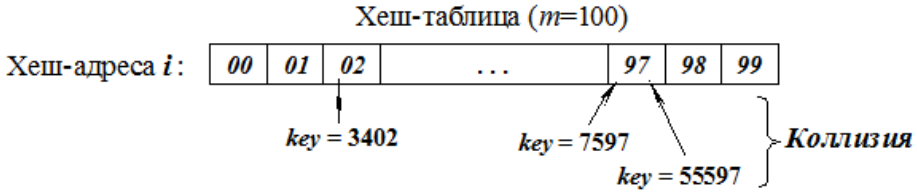


Рис. 1. Хеш-таблица с коллизией / Fig. 1. Hash table with collision

Схематично процесс хеширования можно представить в следующем виде (см. рис. 2):

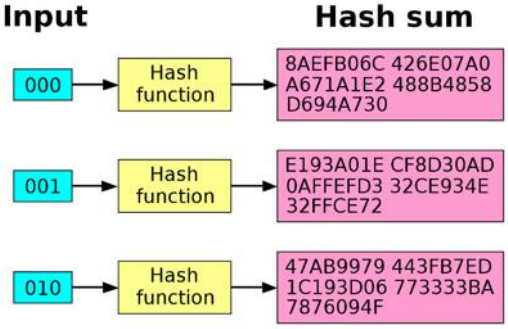


Рис. 2. Процесс хеширования / Fig. 2. The hashing process

На рис.3 приведена классификация криптографических методов. Как видно из рисунка процедура хеширования выделена в отдельную группу криптографических методов.

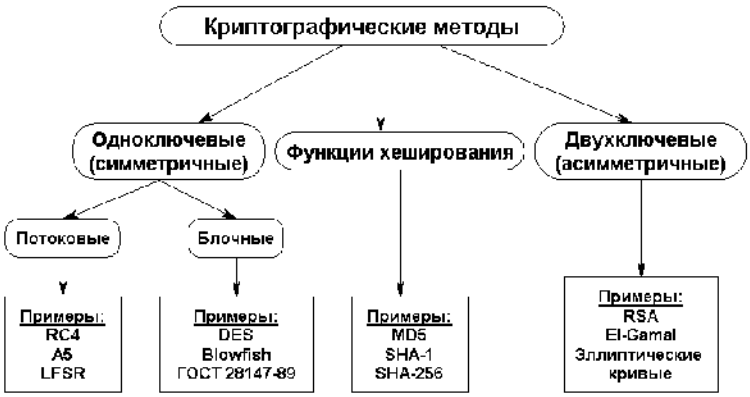


Рис. 3. Классификация криптографических методов / Fig. 3. Classification of cryptographic methods

На сегодняшний день известны и применяются в различных алгоритмах множество методов вычисления хеш-функции: метод свертывания, метод умножения (мультипликативный метод), метод деления.

Метод свертывания, как правило, используется для больших ключей, при том ключ разбивается на части длина которых равна длине требуемого адреса. Сумма этих частей и формирует адрес, перенос в старший разряд при этом игнорируется. Для понимания сути метода рассмотрим пример: пусть имеется ключ 3415768898: Разделим этот адрес на двух- трех и четырехцифровой адрес получим:

34+15+76+88+98=11

3+415+768+898=084

34+1576+8898=0508

Метод умножения (мультипликативный метод) выполняется в два этапа. На первом этапе ключ k умножается на константу A (0<A<1) и выделяется дробная часть полученного произведения. На втором этапе выделенную дробную часть умножают на m (количество хеш-значений), и к нему применяется $h(k)=[m\ (kA \bmod 1)]$.

Результаты и обсуждения. Вычисление хеш-функции методом деления осуществляется путем получения остатка от деления ключа ***k*** на ***m*** (при этом в качестве ***m*** нужно выбирать простые числа, достаточно далекие от степени 2).

Для вычисления хеш-функции применяются множество алгоритмов различной сложности. Среди самых распространенных алгоритмы MD5, SHA-1, SHA-2 и ее разновидности, а также отечественные алгоритмы, изложенный в ГОСТ Р 34.11-94, ГОСТ Р 34.11-2012.

Одним из наиболее распространенных алгоритмов является алгоритмы семейства MD (message digest), в частности алгоритм MD5. Реализация алгоритма осуществляется в пять этапов.

Этап 1. Добавление битов заполнения. В исходное сообщение в конец добавляется один бит, имеющий значение 1, а затем биты имеющие значение 0 до размера конгруэнтного 448 по модулю 512. В конечном счете, количество бит, добавляемых в сообщение не может быть меньше 1 и больше 512.

Этап 2. Добавление размера сообщения. Далее в конец предыдущего этапа добавляется 64-битовое представление b (размер исходного сообщения в битах). При этом размер полученного сообщения будет кратен 512 битам. В результате получается сообщение, содержащее целое число блоков по 16 слов.

Этап 3. Инициализация буфера MD. Перед обработкой создается четыре 32-битовых регистра, содержащий буфер из 4 слова (A,B,C,D). Каждому регистру присваивается определенное первоначальное значение.

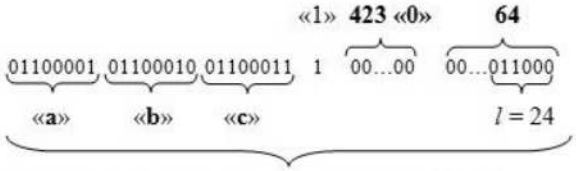
A: 01234567
B: 89abcdef
C: fedcba98
D: 76543210

Этап 4. Обработка сообщения. При обработке сообщения используется четыре функции с тремя аргументами, являющимися 32 битовыми словами. При этом результатом вычисления является 32-битовое слово. Вычисление функции A1(X,Y,Z), **B**1(X,Y,Z), C1(X,Y,Z) и D1(X,Y,Z) осуществляется с помощью элементарных логических операций («логическое умножение», «исключающее ИЛИ», «И», «НЕ», «ИЛИ»). На данном этапе строится 64-элементная таблица с использованием синусоидальной функции.

Этап 5. Вывод. Полученные значения A, B, C, D являются конечным результатом и выводятся, начиная с младшего байта и закачивается старшим байтом.

Другим наиболее эффективным алгоритмом вычисления хеш-функции является алгоритмы семейства SHA (SHA Secure Hash Algorithm – алгоритм ,безопасног хэширования). Реализация алгоритма осуществляется в четыре этапа.

1 этап. Дополнение сообщения. Конец исходного сообщения ***M*** заполняется значением «1», а потом далее нулями - в количестве *i*, таким образом, чтобы размер полученного сообщения был на 64 разряда меньше числа, кратного 512. Затем, к полученному результату добавляется 64-битовое представление размера исходного сообщения M.. В итоге получим 512-битовое сообщение вида:



Размер дополненного сообщения - 512 бит

2 этап. Разбиение дополненного сообщения на M-битные блоки. Полученное в результате дополнения сообщение разбивается на N блоков по 512 бит: M(1), M(2) … M(N). Таким образом, блоки по 512 бит можно представить как шестнадцать 32-битных слов, тогда первый блок сообщения обозначим M0(i), следующие M1(i), и последнее M15(i).

3 этап. Установка инициализирующих значений. Перед тем, как вычислить значение хеш-функции провести инициализацию H, т.е присвоить первоначальные значения. Четыре 32-битных слова.

H0 = 0x67452301
H1 = 0xefcdab89
H2 = 0x98badcfe
H3 = 0x10325476
H4 = 0xc3d2e1f0

4 этап. Вычисление хэша

Цикл от 1 до N

{

1. Преобразование 16 слов размером 32 бит (с *M0(i)* по *M15(i)*) в 80 слов размером 32бита (с *W0* по *W79*):

Wt = *Mt*, при t = 0..15

Wt = **ROT L**(*Wt-3 XOR Wt-8 XOR Wt-14 XOR Wt-16, 1*), при t = 16..79

(*ROT L(x, 1)*- циклический сдвиг влево)

2. Инициализация переменных a,b,c,d,e.

A1 = *H0(i-1)*

B1 = *H1(i-1)*

C1 = *H2(i-1)*

D1 = *H3(i-1)*

E1 = *H4(i-1)*

3. Главный цикл функции сжатия

Цикл от 1 до N

TEMP1 = **ROTL**(*A!, 5*) + *ft(B1,C1,D1)* + *E1* + *Wt* + *Kt*

E1 = *D1*

D1 = *C1*

C1 = **ROT L**(*B1, 30*)

B1 = *A1*

A1 = *TEMP1*

4. Вычисление промежуточного значения хэш-функции:

H0(i) = *(H0(i-1)* + *A1)*

H1(i) = *(H1(i-1)* + *B1)*

H2(i) = *(H2(i-1)* + *C1)*

H3(i) = *(H3(i-1)* + *D1)*

H4(i) = *(H4(i-1)* + *D1)*

В результате получается хэш-значение 5 слов * 32 бита = 160 бит.

В ходе работы над данной статьей было разработано программное приложение, обеспечивающее вычисление хеш-функции различными алгоритмами. Программа позволяет вычислить хеш-функции с использованием различных алгоритмов, а также произвести оценку эффективности того или иного алгоритма.

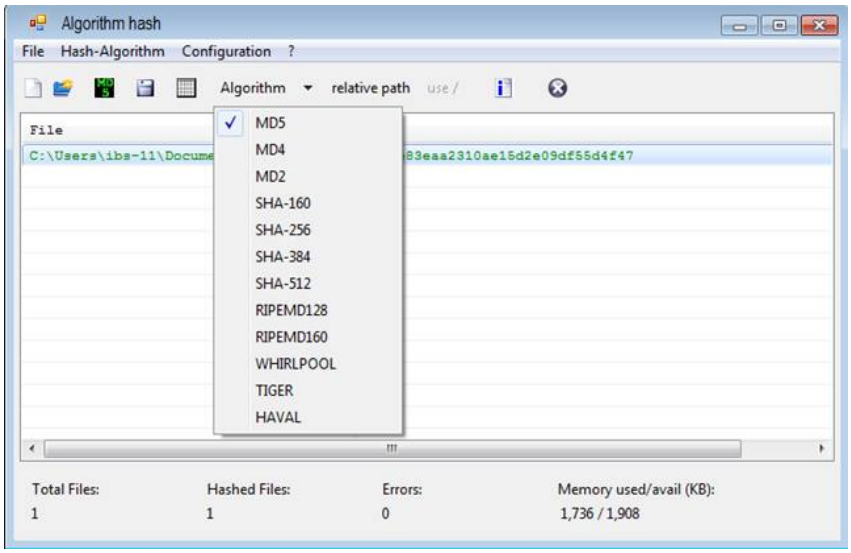


Рис. 4. Интерфейс приложения криптографических алгоритмов вычисления хеш-функции /
Fig. 4. Application interface of cryptographic algorithms for computing a hash function

Еще одним из алгоритмов вычисления хеш-функции является российский стандарт ГОСТ Р 34.11-2012 («Стрибог»). Структура его отличается от структуры алгоритмов семейства MD и SHA. Размер блоков сообщения и внутреннего состояния, создаваемого алгоритмом ГОСТ Р 34.11-2012, равна 512 битам. Данный алгоритм используется для обеспечения целостности, аутентичности, электронно-цифровой подписи, при обработке, хранении и передачи информации в автоматизированных информационных системах органов государственной власти.

Еще одним из важных вопросов является методы разрешения коллизии. Существует несколько методов устранения коллизий, но при этом нагрузка на систему возрастает. Это известные методы: метод цепочек и метод открытой адресации. Для разрешения коллизии методом цепочек формируется массив, содержащий список (цепочку) пар ключ-значение, имеющих одинаковые хеш-значения ключа. Для поиска и удаления используется значение ключа в этом массиве, которые содержит эти линейные цепочки. При открытой адресации все записи хранятся в хеш-таблице. Удаление записей данным методом несколько затруднителен, для этого создается логический флаг для каждой ячейки, помечающий удален элемент в ней или нет, но при этом необходимо модифицировать процедуру поиска существующего элемента, таким образом, чтобы она посчитала ее занятым. Полностью избавиться от коллизий невозможно, но снизить вероятность возникновения и разрешения проблем с коллизиями возможно. Поэтому любые исследования в этой области являются актуальными на сегодняшний день.

Заключения /выводы. Таким образом, учитывая растущие вычислительные мощности современных компьютерных систем, обеспечение защиты данных в информационных системах является одним из наиболее актуальных задач. Поиск эффективных алгоритмов и практических решений должны опережать вероятности нахождения коллизий, тем самым обеспечивая сохранность и целостность данных.

ЛИТЕРАТУРА

1. Бабаш А. В., Шанкин Г. П. Криптография / под ред. В. П. Шерстюка, Э. А. Применко. М.: СОЛОН-ПРЕСС, 2007.

2. Баричев С. Г, Гончаров В. В., Серов Р. Е. Основы современной криптографии: учебный курс. М.: Горячая линия-Телеком, 2002.

3. Болелов Э. А. Криптографические методы защиты информации. Часть 1. Симметричные криптосистемы. М.: МГТУ ГА, 2011.

4. Варфоломеев А. А., Жуков А. Е., Пудовкина М. А. Поточные криптосистемы. Основные свойства и методы анализа стойкости. М.: ПАИМС, 2000.

5. Дональд Кнут. Искусство программирования, том 3. Сортировка и поиск. М.: «Вильямс», 2007.

6. Методы сортировки и поиска [Электронный ресурс]. URL: http:// www.citforum.ru/programming/theory/sorting/sorting2.shtml

7. Молдовян Н. А. Теоретический минимум и алгоритмы цифровой подписи. СПб.: БХВ-Петербург, 2010.

8. Рябко Б. Я., Фионов А. Н. Основы современной криптографии и стеганографии: учебное пособие для вузов. М.: Горячая линия-Телеком, 2010.

9. Харин Ю. С. и др. Математические и компьютерные основы криптологии: учеб. пособие. Мн.: Новое знание, 2003.

10. Ян Сонг Й. Криптоанализ RSA. М.-Ижевск: НИЦ «Регулярная и хаотическая динамика», Ижевский институт компьютерных исследований, 2011.

11. ГОСТ Р 34.10 2012 – Процессы формирования и проверки электронной цифровой подписи.

REFERENCES

1. Babash A. V., Shankin G. P. Kriptografiya / pod redaktsiey V. P. Sherstyuka, Eh. A. Primenko. M.: SOLON-PRESS, 2007.

2. Barichev S. G., Goncharov V. V., Serov R. E. Osnovy sovremennoy kriptografii: Uchebnyy kurs. M.: Goryachaya liniya-Telekom, 2002.

3. Bolelov Eh. A. Kriptograficheskie metody zashchity informatsii. Chast' 1. Simmetrichnye kriptosistemy. M.: MGTU GA, 2011.

4. Varfolomeev A. A., Zhukov A. E., Pudovkina M. A. Potochnye kriptosistemy. Osnovnye svoystva i metody analiza stoykosti. M.: PAIMS, 2000.

5. Donal'd Knut. Iskusstvo programmirovaniya, tom 3. Sortirovka i poisk. M.: «Vil'yams», 2007.

6. Metody sortirovki i poiska [Ehlektronnyy resurs]. URL: http:// www.citforum.ru/programming/theory/sorting/sorting2.shtml

7. Moldovyan N. A. Teoreticheskiy minimum i algoritmy tsifrovoy podpisi. SPb.: BKHV-Peterburg, 2010.

8. Ryabko B. Ya., Fionov A. N. Osnovy sovremennoy kriptografii i steganografii: Uchebnoe posobie dlya vuzov. M.: Goryachaya liniya-Telekom, 2010.

9. Kharin Yu. S. i dr. Matematicheskie i komp'yuternye osnovy kriptologii: Ucheb. posobie. Mn.: Novoe znanie, 2003.

10. Yan Song Y. Kriptoanaliz RSA. M.-Izhevsk: NITS «Regulyarnaya i khaoticheskaya dinamika», Izhevskiy institut komp'yuternykh issledovaniy, 2011.

11. GOST R 34.10 2012 – Protsessy formirovaniya i proverki ehlektronnoy tsifrovoy podpisi.

ОБ АВТОРЕ

Антонов Владимир Феохарович, кандидат технических наук, доцент, доцент кафедры систем управления и информационных технологий, Института сервиса, туризма и дизайна (филиал) СКФУ в г. Пятигорске, e-mail: antonovpgtu@mail.ru

Antonov Vladimir Feokharovich, Candidate of Technical Sciences, Associate Professor, Associate Professor of the Department of management systems and information technologies, Institute of service, tourism and design (branch) of NCFU in Pyatigorsk, e-mail: antonovpgtu@mail.ru

Дата поступления в редакцию: 20.03.2019
После рецензирования: 1.08.2019
Дата принятия к публикаци: 15.08.2019