rossiyskoy-federatsii-organizatsionno-pravovye-problemy. – Дата доступа: 08.11.2021.

2. Куклина, Т. В. Теоретические основы и классификация налогового администрирования [Электронный ресурс] // Режим доступа: https://cyberleninka.ru/article/n/teoreticheskie-osnovy-i-klassifikatsiya-nalogovogo-administrirovaniya. – Дата доступа: 08.11.2021.

3. ITIL «Service Strategy» («Стратегия сервиса») [Электронный ресурс]. – Режим доступа: https://www.smlogic.ru/g-it-s/ itil/s-strat/ (дата обращения: 15.11.2021)

4. Wessel, M. Creating Value in a digital economy. 2017 [Электронный ресурс]. – Режим доступа: https://hbr.org/webinar/2017/01/ creating-value-in-a-digital-economy (дата обращения: 15.11.2021)

5. Зараменских Е.П. Цифровые сервисы: их атрибуты и взаимосвязь с архитектурой предприятия [Электронный ресурс]: Вестник университета Стратегии и инновации – 2018 г. Режим доступа: https://vestnik.guu.ru/jour/article/view/1166/589. Дата доступа: 25.11.2021.

УДК 004.056

К.С. Сакан, Д.С. Дюсенбаев, К.Т. Алгазы,
О.А. Лизунов, Хомпыш Ардабек
Институт информационных и вычислительных технологий МОН РК,
Алматы, Республика Казахстан

РАЗРАБОТКА И АНАЛИЗ АЛГОРИТМА ХЕШИРОВАНИЯ «HAS01»

Аннотация. *В данной статье представлена разработка нового алгоритма хеширования «HAS01» на основе криптографической губки. Отличие конструкции заключается, что функция f, входящая в состав функции F, вызывается более одного раза. Также проведен ряд исследований алгоритма по требованиям, предъявляемые хеш-функциям*

S.K. Sakanuly, D.S. Dyussenbayev, K.T. Algazy,
O.A. Lizunov, Khompysh Ardabek
Institute of Information and Computational MES RK
Almaty, Republic of Kazakhstan

DEVELOPMENT AND RESEARCH OF HASHING ALGORITHM «HAS01»

Abstract. *This article presents the development of a new HAS01 hashing algorithm based on a cryptographic sponge. The difference in the construction is that the function f, which is part of the function F, is called more than once. A number of studies of the algorithm on the requirements of hash functions have also been carried out.*

Введение. Алгоритм хеширования «HAS01» основан на конструкции «губки», которая преобразует открытый текст произвольной длины, состоящий из 192-битовых (или 24-байтовых) блоков в хеш-значение размером 256 или 512 битов. Функция губки представляет собой простую итеративную конструкцию с входными данными переменной длины и выходными данными произвольной длины на основе преобразования f, работающего с фиксированным числом битов b, где b называется шириной. Конструкция губки работает с данными состояния, состоящими из $b = r + c$ бит, где r – скорость, а c – ёмкость губки. Первоначально все b битов состояния устанавливаются в ноль, а входное сообщение дополняется и делится на блоки по r бит каждый. Выполнение функции губки проходит в две стадии: поглощение и выжимание. Общая схема криптографической губки представлена на рисунке 2.5. M_i – входные данные, z_i – хешированные выходные данные.

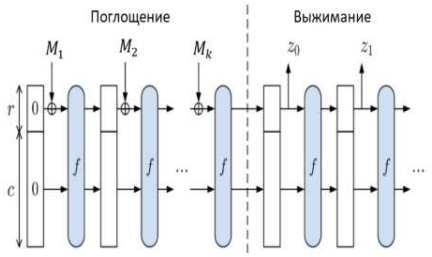

Рис. 1 – Конструкция губки для хеш-функций

Структура входных и выходных данных алгоритма «HAS01». Блок входных данных M_i имеет размер 24 байта и может быть представлен в виде 24-байтовой последовательности $m_1, m_2, \dots, m_{24}$ или как матрица размером [3×8]. Внешнее состояние хеша r_i имеет тот же размер, что и размер блока входных данных (24 байта), и может быть представлено как матрица размером [3×8]. Внутреннее состояния хеша c_i имеет размер 40 байт и может быть представлено как матрица размером [5×8]. Таким образом, полное состояние хеша y_i представляет собой объединение внешнего и внутреннего состояния, то есть $y_i = r_i \,||\, c_i$, и может быть представлено как матрица размером [8×8].

Хеш-значение $h(M)$ представляет собой байтовую последовательность $z_0, z_1, \dots, z_l$. При $l = 4$ размер хеш-значения составляет 256 бит, при $l = 8$ размер хеш-значения составляет 512 бит.

Математическое описание алгоритма «HAS01».

В начале алгоритм хеширования разделяет сообщение M с открытым текстом на 192-битные (24-байтовые) блоки $M_1, M_2, \dots, M_k$. Если длина сообщения M не кратна длине блока, то выполняется дополнение последнего блока единицей с последующими нулями. В случае, когда длина *M* кратна *192* битам, в конец *M* добавляется еще один *192*-битный блок, состоящий из нулевых битов, за исключением первого бита, который равен единице: $M = M_1||M_2||M_3|| \dots ||M_{k-1}||M_k$. Начальное состояние хеша y_0 содержит только нулевые значения. На стадии поглощения выполняются следующие преобразования:

$$F(M_1, 0) = f\left(f\left(f\left(f(((M_1 \oplus 0)||0)\right)\right)\right) = y_1 = r_1||c_1$$

$$F(M_2, y_1) = f\left(f\left(f\left(f((M_2 \oplus r_1)||c_1)\right)\right)\right) = y_2 = r_2||c_2$$

$$F(M_k, y_{k-1}) = f\left(f\left(f\left(f((M_k \oplus r_{k-1})||c_{k-1})\right)\right)\right) = y_k = r_k||c_k$$

На стадии выжимания выполняются следующие преобразования:

$$z_1 \leftarrow f(y_k) = x_1, z_2 \leftarrow f(x_1) = x_2, \quad \dots \quad z_l \leftarrow f(y_{l-1}) = x_l$$

$$h(M) = z_1||z_2|| \dots ||z_l$$

где $f(X)$ – функция преобразования.

Описание функции f. Функция f: [8×8]→[8×8] преобразует 64-байтовую матрицу A размером [8×8], содержащую текущее состояние хеша, в 64-байтовую матрицу A' размером [8 × 8], которая содержит новое состояние хеша. Функция f представляет собой композицию функций f_1, f_2 и f_3: $f(A) = f_1 \circ f_3 \circ f_2 \circ f_1 \circ f_3 \circ f_1 \circ f_2 \circ f_1(A)$.

Описание функции f_1. Функция f_1: [8×8] → [8×8] выполняет нелинейное преобразование матрицы A размером [8×8]. Преобразование элементов матрицы A осуществляется по строкам слева направо и сверху вниз, используя операций XOR, логический сдвиг и таблицы замены S-блока.

$$a'_{(i+1)\ mod\ 8,j} = \mathrm{S}\left(\left(\left(a_{i,j} \oplus a_{(i+1)\ mod\ 8,j}\right) \ll 3\right) \oplus \left(\left(a_{i,j+1} \oplus a_{(i+1)\ mod\ 8,j+1}\right) \gg 5\right)\right),$$

$$i = 1,2, \dots,7, j = 1,2, \dots,6.$$

$$a'_{(i+1)\ mod\ 8,7} = S\left(\left(\left(a_{i,7} \oplus a_{(i+1)\ mod\ 8,7}\right) \ll 3\right) \oplus \left(\left(a_{i,0} \oplus a_{(i+1)\ mod\ 8,0}\right) \gg 5\right) \oplus 7\right),$$

$i = 1,2,\dots,7.$

Описание функции f_2. Функция f_2 выполняет транспонирование матрицы A размером [8×8], состоящей из элементов $a_{i,j}$ множества $GF(2^8)$, принимающих значения от 0 до 255.

Описание функции f_3. Функция f_3 преобразует матрицу A размером [8×8] в матрицу B размером [8×8] по строкам по формуле:

$$b_{i,k} = \oplus_{j=0}^{7}\left(\left(\left((a_{i,j}[x] \times x^{8-k}) mod(x^8+1)\right)\&1\right) \times x^{7-j}\right).$$

Общая схема алгоритма хеширования «HAS01».

На рис. 2 представлена общая схема алгоритма хеширования «HAS01». M_i – входные данные, z_i – хешированные выходные данные. На стадии поглощения функция f, входящая в состав функции F, вызывается более одного раза. Кроме того, при формировании хеш-значения $h(M) = z_0, z_1, \dots, z_l$ каждый элемент последовательности z_i получается путём копирования определённого столбца матрицы состояния, текущего состояние хеша, сформированное на данный момент. Длина блока исходного сообщения и длина внешнего состояния хеша r_i равны 24 байтам. После этого функция $F(y_i)$ выполняет описанные выше преобразования текущего состояния y_i хеша, включающего внутреннее состояние c_i и внешнее состояние r_i. Стадия поглощения завершается после преобразования последнего блока M_k исходного сообщения M. На стадии выжимания формируются элементы z_i хеш-значения $i = \overline{1,l}$. При $l = 4$ размер хеш-значения составляет 256 бит, при $l = 8$ размер хеш-значения составляет 512 бит.

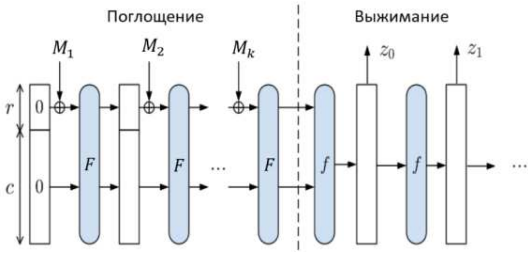

Рис. 2 – Конструкция губки, используемая алгоритмом хеширования «HAS01»

Оценка исследования «лавинного эффекта» и криптостойкости алгоритма хеширования HAS01

Основными методами криптоанализа хеш-функций являются методы полного перебора и поиска коллизии. Ниже приведён анализ F-функции хеш-алгоритма HAS01 с использованием этих методов.

Исследование лавинного эффекта. Разброс битов хеш-алгоритма HAS01 для F-функции показан на рис. 3. В качестве входов F-функции выбираются 512-битные блоки, каждый из которых отличается от первого только одним битом (при этом все биты разные). Вероятность совпадения соответствующих битов зашифрованного текста при преобразовании (шифровании) этих блоков с помощью F-функции находится между 0,4277 и 0,5722. Следовательно, разброс битов F-функции указывает на то, что эта функция удовлетворяет критерию лавинного эффекта.

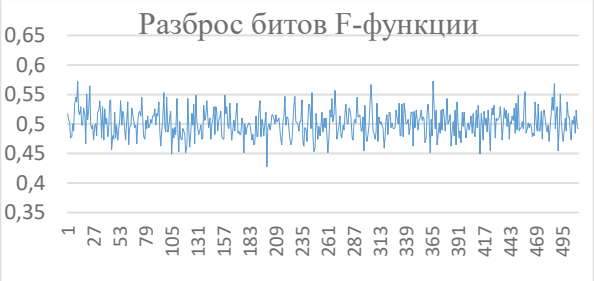

Рис. 3 – Диаграмма лавинного эффекта F-функции

Исследование криптографической стойкости алгоритма

Метод полного перебора. Метод полного перебора оценивает возникновение по крайней мере одной коллизии при атаке с использованием парадокса «дней рождения» с вероятностью 0,5, при переборе $3{,}4 \cdot 10^{38}$ для 256-битного хеш-значения и $1{,}15 \cdot 10^{77}$ для 512-битного хеш-значения.

Метод поиска коллизий. При поиске коллизии нужно попробовать найти слабые места в алгоритме, которые позволят избежать полного перебора. Одним из таких мест в рассматриваемом нами алгоритме является функция f_1. Функции f_2 и f_3 выполняют перестановку байтов и битов, следовательно, применение атаки методом поиска коллизий против функций f_2, f_3 не представляет угрозы. Единственным слабым местом F-функции является нелинейная функция f_1, поскольку входом S-блока является сумма значений по модулю 2. Если значения разные, но сумма их одинакова, то на выходе S-блока получается одно и то же значение. Следовательно, должны быть выполнены проверки всех входов F-функции, которые дают одинаковое значение суммы в функции f_1, то есть необходимо выполнить проверку только тех элементов, которые расположены друг от друга на расстоянии, равном длине строки матрицы. Проверим хеш-значения двух текстов в шестнадцатеричном формате.

Первый текст:

$$80\ \underbrace{00\ 00\ \ldots\ 00\ 00}_{63\text{ раза}}\ 40\ \underbrace{00\ 00\ \ldots 00\ 00}_{63\text{ раза}}\ c0\ \underbrace{00\ 00\ \ldots 00\ 00}_{63\text{ раза}}\ 40\ \underbrace{00\ \ldots\ 00}_{5\text{ раз}}\ 01\ \underbrace{00\ 00\ \ldots 00\ 00}_{56\text{ раз}}\ 20$$

Второй текст:

$\underbrace{00\ 00\ \ldots\ 00\ 00}_{8\text{ раз}}\ 80\ \underbrace{00\ 00\ \ldots 00\ 00}_{63\text{ раза}}\ 40\ \underbrace{00\ 00\ \ldots 00\ 00}_{63\text{ раза}}\ c0\ \underbrace{00\ \ldots\ 00}_{61\text{ раз}}\ 01\ 00\ 40\ \underbrace{00\ 00\ \ldots 00\ 00}_{54\text{ раза}}\ 20$

Их хеш-значения, следующие:

256-битное хеш-значение первого текста: 4b d2 de fd 01 1f 52 a9 8e 46 16 83 4c 07 2b 4c 28 43 f5 27 df 34 bc a9 da 58 45 0b 63 88 eb ad

256-битное хеш-значение второго текста: 46 df 0f 4d a0 3d cf c4 ea d2 59 d9 46 ae 7b bc 77 3e 8e 67 f6 5a d4 d5 f1 60 f9 4a f0 c1 00 65

512-битное хеш-значение первого текста: 3f e0 ef 8a a9 6f 80 bf d0 49 3f 7a 7d f1 84 fb 98 b4 87 42 e4 65 7b a6 2e 7c 71 7 19 78 fc 6 25 bf 5c ae 8c 63 49 b2 82 7f 1c 38 b9 e6 80 38 6e 59 6d 7a f3 44 68 d6 e0 23 99 ce 42 e3 df 74

512-битное хеш-значение второго текста: f6 a8 51 d9 80 be 62 66 3c 0a ae 10 b8 c3 7e 3e 42 35 ae 66 2f 26 16 81 90 10 4a fd 38 66 a9 3c 31 c6 7c d4 4b be 88 eb bf c0 61 cf 89 67 80 83 c0 49 e7 e7 e5 ef db e8 00 cd 05 13 1b 63 d7 55

Необходимо проверить все такие случаи на предмет возникновения коллизий. Если эта проверка не даёт результата, то вероятность возникновения коллизий соответствует вероятности полного перебора, значение которого было приведено выше.

Прежде всего мы рассмотрели все 512 позиций со смещением на одну позицию на входе функции f_1 по битовому рассеянию (лавинному эффекту). В итоге обнаружили, что они дают разные значения, поэтому вход функции f_1 не приводит к коллизиям. На следующем этапе выполним полную проверку элементов, расположенных в соседних позициях матрицы, участвующих в преобразовании функции f_1.

Заключение. Разработан алгоритм хеширования «HAS01» на основе конструкции криптографической «губки». Изучены надежности данного алгоритма по оценкам анализа «лавинного эффекта» и F-функция алгоритма HAS01 исследована методами полного перебора и поиска коллизии. В настоящее время проводятся другие исследования.

Благодарность. Исследовательская работа выполнена в рамках проекта программно-целевого финансирования OR11465439 «Разработка и исследование алгоритмов хеширования произвольной длины для цифровых подписей и оценка их стойкости» Министерства образования и науки Республики Казахстан.

Список использованных источников:

1 А. В. Сидоренко, М. С. Шишко Алгоритм хеширования на основе SHA-3 с использованием хаотических отображений // Информатика. -

2020. - Т. 17, № 1. - С. 109–118 // https://doi.org/10.37661/1816-0301-2020-17-1-109-118.

2 G. Bertoni, J. Daemen, M. Peeters, G. Van Assche, The Keccak reference, Version 3.0. - 2011 // http://keccak.noekeon.org/, Keccak-reference-3.0.pdf: 11.10.2021.

3 Мулярчик К. С. // Лавинный эффект в алгоритмах шифрования на основе дискретных хаотических отображений // Доклады БГУИР. - 2013. - № 6 (76). - С. 86-91.

4 Vergili I., Yücel M. D. Avalanche and Bit Independence Properties for the Ensembles of Randomly Cho-sen× S-Boxes // Turk J Elec Engin. - 2001. - Т. 9, №. 2. - С. 137-145.

УДК 004.056.53

Ю.Д. Сандихаев, Д.Е. Сидорчик
Белорусский государственный технологический университет,
Минск, Беларусь

ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ В АСУТП

Аннотация. *В работе рассматриваются проблемы широкого внедрения автоматизированных систем во все сферы жизни общества, что требует повышенного внимания к защите применяемых для автоматизации управления информационных технологий и непосредственно информации.*

Yu.D. Sandihaev, D.E. Sidorchyk
Belarusian State Technological University,
Minsk, Belarus

INFORMATION SECURITY IN THE AUTOMATED CONTROL SYSTEM OF TECHNOLOGICAL PROCESSES

Abstract. *The paper deals with the problems of widespread introduction of automated systems in all spheres of society, which requires increased attention to the protection of information technologies and information directly used for automation of management.*

Широкое внедрение автоматизированных систем во все сферы жизни общества требует повышенного внимания к защите применяемых для автоматизации управления информационных технологий и непосредственно информации. Любые нарушения и неполадки в работе