14. *Odlyzhko A.O.* Cryptanalytic attacks on the multiplicative knapsack cryptosystem and on Shamir's fast signature scheme // IEEE Transactions on Information Theory. – Jul 1984. – Vol. IT-30, № 4. – P. 594-601.

Статью рекомендовал к опубликованию д.т.н., профессор Р.З. Камалян.

Осипян Валерий Осипович – Кубанский государственный университет; e-mail: rrwo@mail.ru; 350040, г. Краснодар, ул. Ставропольская, 149; тел.: 89184651399; кафедра информационный технологий; д.ф.-м.н.; профессор.

Жук Арсений Сергеевич – e-mail: arseniyzhuck@mail.ru; тел.: 89654620300; аспирант.

Арутюнян Ашот Хоренович – e-mail: ashotax@gmail.com; тел.: 89180319557; аспирант.

Карпенко Юрий Александрович – Адыгейский государственный университет; e-mail: rrwo@mail.ru; 385000, г. Майкоп, ул. Свободы, 233, кв. 71; тел.: 89184651399; аспирант.

Osipyan Valeriy Osipovich –Kuban State University; e-mail: rrwo@mail.ru; 149, Stavropol'skaya street, Krasnodar, 350040, Russia; phone: +79184651399; the department of information technology; dr. of phis.-math. sc.; professor.

Karpenko Yuriy Aleksandrovich – Adyghe State University; e-mail: rrwo@mail.ru; 233 / 71, Svobody street, Maikop, 385000; phone: +79184651399; postgraduate student.

Zhuck Arseniy Sergeevich – mail: arseniyzhuck@mail.ru; phone: +79654620300; postgraduate student.

Arutyunyan Ashot Horenovich – e-mail: ashotax@gmail.com; phone: +79180319557; postgraduate student.

УДК 681.03.245

Л.К. Бабенко, Е.А. Ищукова

ФИНАЛИСТЫ КОНКУРСА SHA-3 И ОСНОВНЫЕ СВЕДЕНИЯ ОБ ИХ АНАЛИЗЕ*

В последние годы в научном мире наблюдается повышенный интерес к проектированию и анализу алгоритмов хэширования. Наряду с анализом уже существующих функций хэширования, предлагаются новые, заявляемые авторами как более надежные. Кроме того, предлагаются новые методы анализа, которые, как правило, рассчитаны на довольно широкий класс алгоритмов хэширования. Подтверждением тому служит конкурс на принятие нового стандарта хэширования SHA-3, недавно завершенный Национальным институтом стандартов и технологий США. В настоящей статье рассматриваются основы построения функций хэширования, которые явились финалистами конкурса SHA-3. Рассматриваются следующие хэш-функции: BLAKE, Skein, JH, Keccak, Grostl. Для каждой функции рассматриваются основные шаги преобразования и составляющие компоненты. Так, в частности, в составе функции хэширования Skein описывается новый блочный алгоритм шифрования Threefish. Также в статье приводятся основные сведения, известные на данный момент, об основных результатах анализа рассматриваемых функций хэширования.

Криптграфия; анализ; функция хэширования; надежность; стойкость; шифр.

* Работа выполнена при поддержке грантов РФФИ №12-07-31120_мол_а, №12-07-33007_мол_а_вед, № 12-07-00037-а.

L.K. Babenko, E.A. Ischukova

COMPETITION FINALISTS OF SHA-3 AND INFORMATION ON THEIR ANALYSIS

In recent years, in the scientific world has been an increased interest in the design and analysis of hash algorithms. Along with the analysis of existing hash functions, are offered new, claimed by authors as being more reliable. Besides, the new methods of the analysis generally calculated on quite wide class of hashing algorithms are offered. Confirmation to that is competition on adoption of the new standard of hashing SHA-3 which has been recently finished by National institute of standards and technologies. The article discusses the basics of building hashing functions , which were the finalists SHA- 3. It covers the following hash functions: BLAKE, Skein, JH, Keccak, Grostl. The basic steps of transformation and its component parts are considereds for each function. Thus, in particular, a new block cipher algorithm Threefish is described in the Skein hash function. The article also provides the basic information of the known at the moment, Also its provides the main results of the analysis under consideration hashing functions .

Cryptography; analysis; hash function; reliability; strength; cipher.

В последние годы в научном мире наблюдается повышенный интерес к проектированию и анализу алгоритмов хэширования. Об этом свидетельствует большое количество статей, посвященных хэш-функциям, представляемым на мировых конференциях по криптографии CRYPTO и EUROCRYPT. Авторами этих статей часто являются люди, стоящие у истоков современной криптографии, такие как Эли Бихам (Eli Biham), Ади Шамир (Adi Shamir), Барт Пренил (Bart Preneel), Ларс Кнудсен (Lars R. Knudsen), Рональд Ривест (Ronald L. Rivest), Алекс Бирюков (Alex Biryukov), Орр Дункелман (Orr Dunkelman), Винсент Рижмен (Vincent Rijmen) и др.

Наряду с анализом уже существующих функций хэширования, предлагаются новые, заявляемые авторами как более надежные. Кроме того, предлагаются новые методы анализа, которые, как правило, рассчитаны на довольно широкий класс алгоритмов хэширования. Подтверждением тому служит конкурс на принятие нового стандарта хэширования SHA-3, недавно завершенный Национальным институтом стандартов и технологий США (НИСТ – National Institute of Standards and Technology (NIST)). В 2002 году в США был принят стандарт Federal Information Processing Standard 180-2 (FIPS 180-2), определявший 5 основных функций хэширования SHA-1, SHA-224, SHA-256, SHA-384 и SHA-512. Появление серии работ японских ученых, направленных на анализ алгоритмов семейства SHA [1–3], позволило усомниться в стойкости данного стандарта. Так, в работе [3] заявлено, что для алгоритма SHA-1 возможно выполнить поиск коллизий. Несмотря на то, что работы [1–3] не содержат полных сведений о методах, предложенных для анализа, и на заявление сотрудника НИСТ Вильяма Бюрра (William E. Burr) о том, что метод поиска коллизий, предложенный в работе [3] до сих пор никем не подтвержден, в ноябре 2007 года стартовал проект, направленный на поиск и принятие стандарта нового поколения SHA-3. При этом на сайте НИСТ были опубликованы сведения о том, что после 2010 года алгоритм SHA-1 не должен быть использован для ЭЦП и любых других приложений, требующих устойчивости к поиску коллизий (здесь и далее данные взяты с сайта Национального института стандартов и технологий США http://csrc.nist.gov).

На конкурс SHA-3 был представлен 51 алгоритм, о чем было объявлено в декабре 2008 г. В результате первого раунда конкурса, который завершился в июле 2009 г. было отобрано 14 претендентов: BLAKE (автор Jean-Philippe Aumasson), Blue Midnight Wish (автор Svein Johan Knapskog), CubeHash (автор D.J. Bernstein), ECHO (автор Henri Gilbert), Fugue (автор Charanjit S. Jutla), Grøstl (автор Lars Ramkilde Knudsen), Hamsi (автор Ozgul Kucuk), JH (автор Hongjun Wu), Keccak

(автор Joan Daemen), Luffa (автор Dai Watanabe), Shabal (автор Jean-Francois Misarsky), SHAvite-3 (автор Orr Dunkelman), SIMD (автор Gaetan Leurent), Skein (автор Bruce Schneier).

9 декабря 2010 г. NIST объявил о начале третьего этапа конкурса. В течение года институтом принимались дополнительные исследования всех заинтересованных сторон, посвященные исключительно пяти финалистам: Blake, Grostl, JH, Keccak и Skein. Создателям позволили вносить незначительные изменения в их алгоритмы вплоть до 16 января 2011 г. С этого времени алгоритмы считались полностью спроектированными и NIST выделил еще один год для их публичного рассмотрения. Победитель был объявлен в начале 2013 г. Им оказалась функция хэширования Keccak (автор Joan Daemen). Несмотря на то, что в качестве стандарта был выбран всего один алгоритм, все пять финалистов заслуживают особого внимания. Во-первых потому, что в них предложены новые для криптографии подходы к построению современных функций хэширования. Во-вторых потому, что все пять финалистов являются "сильными" функциями и предположительно не уступают друг другу в стойкости. В настоящей статье мы предлагаем ознакомиться с основными принципами, лежащими в основе построения новых функций хэширования – финалистов конкурса SHA-3. А также рассмотреть основные подходы, которые к настоящему времени применялись для их анализа, и полученные с их помощью результаты.

Функция хэширования Skein. Skein представляет собой новое семейство функций хэширования, которые могут оперировать блоками данных различной длины: 256, 512 и 1024 битов. Любой из вариантов функции Skein может быть применим для получения любой длины хэш-значения. Изюминкой Skein является идея построения функции хэширования на основе нового класса блочных шифров, так называемых Настраиваемых Блочных шифров (Tweakable Block Ciphers (TBC)). Говоря более точно Skein основывается на трех новых компонентах. На настраиваемом блочном шифре Threefish, на уникальной блочной итерации (Unique Block Iteration (UBI)), а также на выборочной системе параметров.

Алгоритм Threefish является настраиваемым блочным шифром, в котором помимо открытого текста и секретного ключа используется еще один дополнительный вход (tweak-значение), играющий роль вектора инициализации. Алгоритм Threefish может быть применим к блокам данных различной длины: 256, 512 и 1024 битов. Секретный ключ имеет такую же размерность, как и шифруемый блок данных, tweak-значение имеет размер 128 битов для любого шифруемого блока данных.

При разработке алгоритма Threefish авторы руководствовались тем принципом, что большое число простых раундов обеспечивает шифру большую надежность, чем малое число сложных раундов. Поэтому в основе шифра Threefish лежит всего три простых математических операции: операция сложения по модулю два (XOR), целочисленное сложение по модулю и циклический сдвиг. Все эти операции очень быстро выполняются на современных 64-битных процессорах.

На рис. 1 представлена основная составляющая часть алгоритма Threefish, которая заключается в простой нелинейной функции перемешивания MIX-функции, оперирующей двумя 64-битными блоками данных. Каждая MIX-функция состоит из целочисленного сложения, циклического сдвига и операции XOR.

Уникальная блочная итерация (Unique Block Iteration (UBI)) – это режим сцепления блоков, который объединяет входные связующие данные G с входными последовательностями произвольной длины M и вырабатывает выходное сообщение фиксированного размера. На рис. 2 показана работа режима UBI для функции Skein-512, обрабатывающей входную последовательность длиной 166 байтов. Соответственно входная последовательность разбита на три блока и таким образом обращение к алгоритму Threefish выполняется трижды.

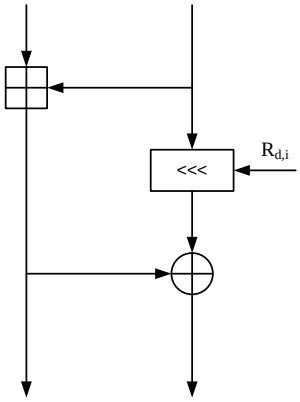

Рис. 1. MIX-функция алгоритма Threefish

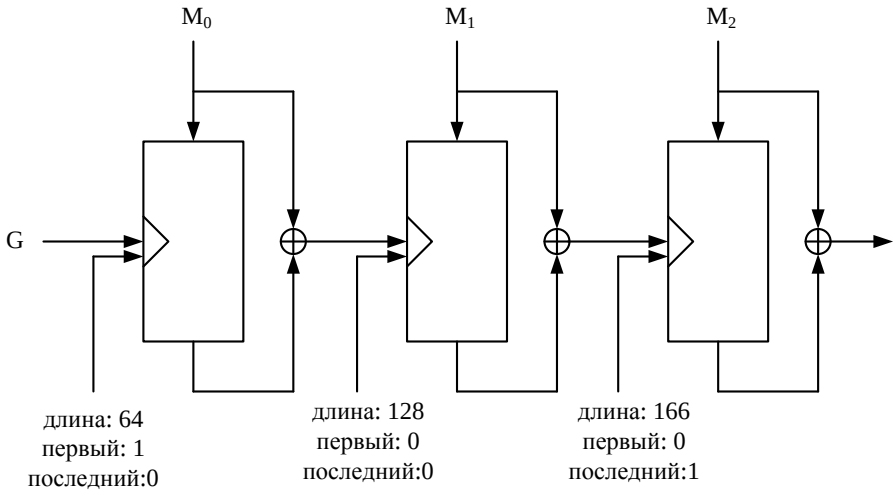

Рис. 2. Хэширование трехблочного сообщения с использованием режима UBI

Блоки сообщения M_0 и M_1 содержат по 64 байта каждый, последний блок M_3 содержит всего 38 байтов хэшируемого сообщения. Tweak-значение для каждого обрабатываемого блока содержит в себе данные о числе обработанных байтов, а также сведения о том является ли обрабатываемый блок первым или последним в обрабатываемой цепочке. Кроме того tweak-значение содержит еще поле «тип» не показанное на рис. 2, которое используется для того, чтобы режим UBI каждый раз работал по-новому.

Таким образом, tweak-значение является основой работы UBI режима. Использование настраиваемого блочного алгоритма шифрования в составе UBI режима гарантирует, что каждый блок будет обработан с использованием уникального варианта сжимающей функции.

Выборочная система параметров позволяет семейству Skein поддерживать различные комбинации исходных параметров без использования дополнительных средств и приложений.

Функция хэширования Skein построена на многократном обращении к режиму UBI. На рис. 3 показана схема обычной функции хэширования Skein. В качестве начальных связующих данных используется значение 0. В процессе выработки хэш-значения режим UBI используется трижды. Первый раз для обработки конфигурационного блока данных, второй раз для обработки сообщения, для которого должно быть получено хэш-значение (максимальная длина обрабатываемого сообщения не должна превышать 2^{96} -1 байтов). В третий раз (выходное преобразование) режим UBI обрабатывает нулевое входное сообщение, что обеспечивает дополнительное перемешивание информации. В результате трехкратного применения режима UBI на выходе будет сформировано хэш-значение фиксированной длины.

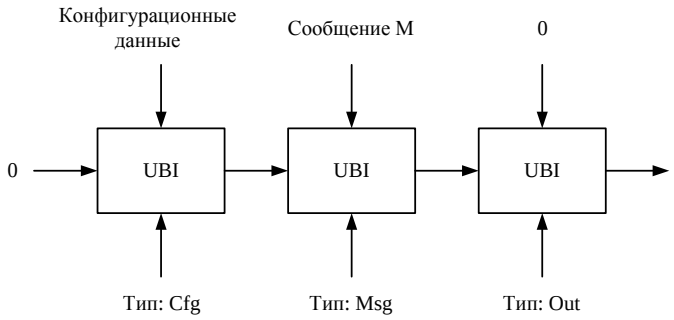

Рис. 3. Обычная функция хэширования Skein

Функция хэширования BLAKE. Авторами функции BLAKE являются Жан-Филипп Омассон (Jean-Philippe Aumasson), Лука Хензен (Luca Henzen), Уилли Мейер (Willi Meier) и Рафаэль Фан (Raphael Phan).

В основе функции лежат ранее представленные авторами алгоритмы, известные как LAKE и ChaCha. Функция BLAKE построена по усиленной схеме Меркля-Дамгаарда.

Допустимые размеры выходных значений: 224, 256, 384 и 512 бит. Алгоритм разбивает данные на блоки по 512 бит (для 224- и 256-битных хэш-значений) или по 1024 бит (для 384- и 512-битных хэш-значений).

Допустимые размеры выходных значений могут быть 224, 256, 384 и 512 бит. В табл. 1 приведены основные параметры хэш-функции BLAKE.

Таблица 1

Основные параметры хэш-функций BLAKE (размеры в битах)

Алгоритм	Слово	Сообщение	Блок	Обзор	Синхропосылка
BLAKE - 224	32	$< 2^{64}$	512	224	128
BLAKE - 256	32	$< 2^{64}$	512	256	128
BLAKE - 384	64	$< 2^{128}$	1024	384	256
BLAKE - 512	64	$< 2^{128}$	1024	512	256

Рассмотрим основные принципы работы функции хэширования BLAKE на примере ее разновидности BLAKE-256. Функция BLAKE-256 работает с 32-битными словами, а возвращает 32-байтное хэш-значение.

BLAKE-256 начинает хэширование с использования инициализирующих значений, так же как это было в SHA-256. Используются 8 векторов инициализации IV_0 ... IV_7. Так же функция BLAKE-256 использует 16 констант, определенных ее разработчиками c_0 ... c_{15}. Во всех функциях BLAKE используются десять перестановок, которые заданы специальными таблицами.

Функция сжатия BLAKE-256 использует 4 входных значения:

- начальное значение h=h0,…,h7;
- блок данных m=m0,…,m15;
- синхропосылка s=s0,…,s3;
- счетчик t=t0,t1.

Эти четыре входных значения представляют собой 30 общих слов (120 байт=960 бит). На выходе функции получается новые значения h'=h'0,…,h'7 из восьми слов (32 байта=256 бит). Шестнадцать слов v0,…,v15 инициализируются, следующим образом:

$$\begin{pmatrix} v_0 & v_1 & v_2 & v_3 \\ v_4 & v_5 & v_6 & v_7 \\ v_8 & v_9 & v_{10} & v_{11} \\ v_{12} & v_{13} & v_{14} & v_{15} \end{pmatrix} \leftarrow \begin{pmatrix} h_0 & h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 & h_7 \\ s_0 \oplus c_0 & s_1 \oplus c_1 & s_2 \oplus c_2 & s_3 \oplus c_3 \\ t_0 \oplus c_4 & t_0 \oplus c_5 & t_1 \oplus c_6 & t_1 \oplus c_7 \end{pmatrix}$$

Раунд функции. Положение v инициализируется, после чего сжатие функции повторяется последовательно 14 раундов. Раунд преобразует положение v с использованием функции Gi:

$$G_0(v_0, v_4, v_8, v_{12}); \quad G_1(v_1, v_5, v_9, v_{13}); \quad G_2(v_2, v_6, v_{10}, v_{14});$$
$$G_4(v_0, v_5, v_{10}, v_{15}); \quad G_5(v_1, v_6, v_{11}, v_{12}); \quad G_6(v_2, v_7, v_8, v_{13});$$
$$G_3(v_3, v_7, v_{11}, v_{15});$$
$$G_7(v_3, v_4, v_9, v_{14}).$$

В течение раунда r для функции Gi (a, b, c, d) проходят следующие преобразования:

$$a \leftarrow a + b + \left(m_{\sigma_r(2i)} \oplus c_{\sigma_r(2i+1)}\right)$$
$$d \leftarrow (d \oplus a) >>> 16$$
$$c \leftarrow c + d$$
$$b \leftarrow (b \oplus c) >>> 12$$
$$a \leftarrow a + b + \left(m_{\sigma_r(2i+1)} \oplus c_{\sigma_r(2i)}\right)$$
$$d \leftarrow (d \oplus a) >>> 8$$
$$c \leftarrow c + d$$
$$b \leftarrow (b \oplus c) >>> 7$$

Значения G0,…,G3 могут быть рассчитаны параллельно, потому что каждый из них модернизирует отдельный столбец матрицы.

Назовем процедуру вычисления G0,…,G3 шагом столбца. Так же, последние четыре значения G4,…,G7 модернизируют отдельные диагонали, которые назовем диагональным шагом. В раундах, где r > 9 используется перестановка $\sigma_{r \bmod 10}$.

Рис. 4 и 5 илюстрируют работу функции Gi, шаг столбца и диагональный шаг.

После прохождения всех раундов преобразования, новые полученные значения h'0,…,h'7 выглядят следующим образом:

$$h'_0 \leftarrow h_0 \oplus s_0 \oplus v_0 \oplus v_8$$
$$h'_1 \leftarrow h_1 \oplus s_1 \oplus v_1 \oplus v_9$$
$$h'_2 \leftarrow h_2 \oplus s_2 \oplus v_2 \oplus v_{10}$$
$$h'_3 \leftarrow h_3 \oplus s_3 \oplus v_3 \oplus v_{11}$$
$$h'_4 \leftarrow h_4 \oplus s_0 \oplus v_4 \oplus v_{12}$$
$$h'_5 \leftarrow h_5 \oplus s_1 \oplus v_5 \oplus v_{13}$$
$$h'_6 \leftarrow h_6 \oplus s_2 \oplus v_6 \oplus v_{14}$$
$$h'_7 \leftarrow h_7 \oplus s_3 \oplus v_7 \oplus v_{15}$$

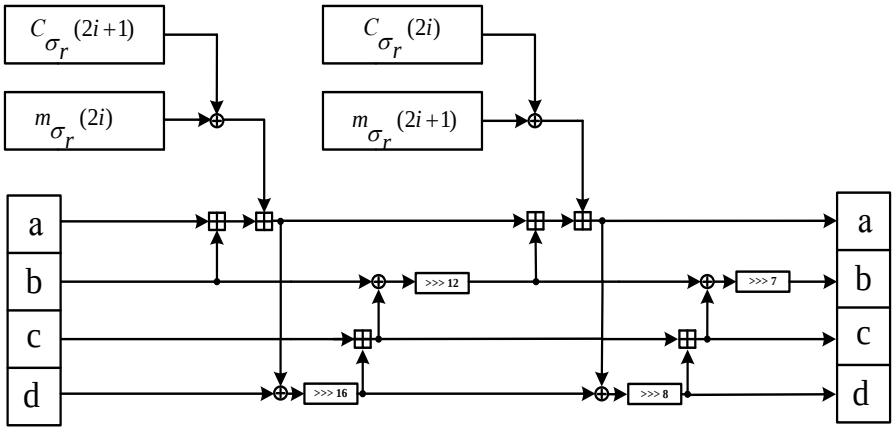

Рис. 4. Функция Gi

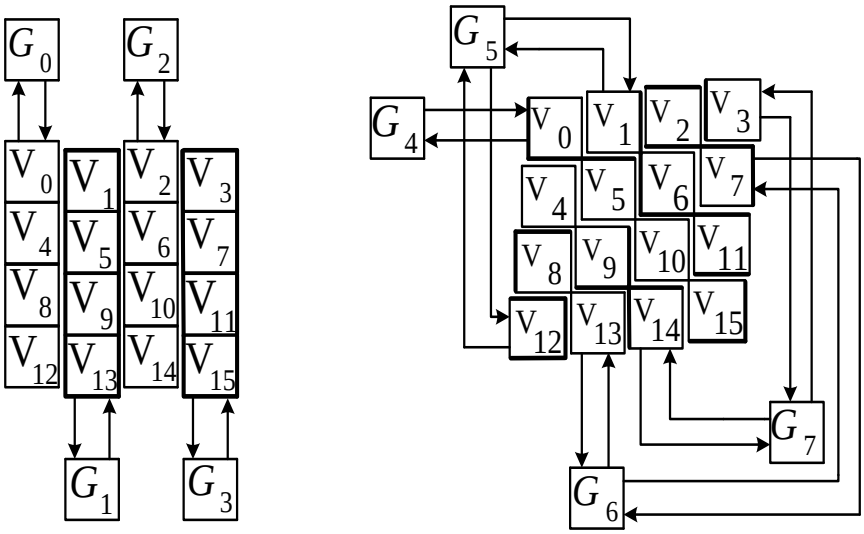

Рис. 5. Шаг столбца и диагональный шаг

Обычно, для повторного хэширования, первое сообщение дополняется по следующему принципу. Первоначально сообщение расширяется таким образом, чтобы его длина была сравнима со значением 447 mod 512. Для этого в начале и в конце дополняемого блока ставится бит, равный 1, а все промежуточные биты принимают значение 0:

$$m \leftarrow m \parallel 1000...0001\langle l \rangle_{64}.$$

Последние 64 бита расширенного блока содержат значение l_{64}, указывающее длину хэшируемого сообщения m.

Если исходное сообщение содержит несколько блоков (в том числе и последний дополненный блок), то хэширование осуществляется в несколько итераций с последовательной обработкой блока. Дополненное сообщение разбивается на 16 слов $m^0,\ldots,m^{N-1}$. Допустим, что l^i номер битового сообщения, а $m^0,\ldots,m^i$ это исключающие биты дополнения. Например, начальное (без дополнения) сообщение длиной 600 бит, тогда сообщение с дополнением имеет 2 блока $l^0 = 512, l^1 = 600.$ Основное правило: если счетчик содержит 0, то последний блок не содержит в себе биты от начального сообщения, это гарантирует что $i \neq j$, когда $l_i \neq l_j$.

Синхропосылка s выбирается пользователем и равна нулю, если она не требуется ($s_0 = s_1 = s_2 = s_3 = 0$). Если сообщение m было дополнено, то хэш-значение рассчитывается следующим образом:

$$h^0 \leftarrow IV$$
$$for \quad i = 0,\ldots,N-1$$
$$h^{i+1} \leftarrow compress\left(h^i, m^i, s, l^i\right)$$
$$return \quad h^N$$

Функция хэширования Keccak. Keccak (читается как "кечак") представляет собой функцию хеширования переменной разрядности. Авторами функции является большая группа разработчиков во главе с Джоном Даеменом, который является соавтором ныне действующего стандарта шифрования данных AES. Размер вырабатываемого функцией хэш-значения может быть равен 224, 256, 384 или 512 бит. При этом используется одно и тоже число раундов – 24. Согласно проведенным исследованиям на этапе отбора, данная функция хэширования является достаточно быстрой (12,5 циклов на байт на системах с CPU Intel Core 2) и эффективно реализуется без больших затрат ресурсов, что позволяет использовать ее без особых проблем в различных по своим параметрам системах. Аппаратная реализация Keccak оказалась наиболее быстрой из всех представленных на конкурс работ [4].

Функция Keccak построена по новому криптографическому принципу, который получил название "губка". и состоит из двух этапов:

Первый этап – «впитывание», заключается в том, что исходное сообщение проходит через многораундовую перестановку f.

Второй этап – «выжимание», заключается в выработке хэш-значения.

Конструкция губки имеет внутреннее состояние с данными фиксированного размера b (бит). Фиксированное состояние делится на 2 части: b=r+c, где r – размер битовой скорости, а c – размер мощности.

В начале работы функции хэшируемое сообщение N разбивается на блоки кратные размеру r. В фазе «впитывания» задается исходное состояние из нулевого вектора размером до 1600 бит. Затем блок N1 складывается по модулю 2 с фраг-

ментом исходного состояния r, вторая часть состояния c остается незатронутой. Полученный результат после сложения проходит через функцию f(), данный алгоритм проделывается для остальных блоков Ni.

Keccak реализован в виде трехмерного массива 5x5 (рис. 2), где одномерная часть содержит: содержит строку “row” из 5 битов констант y и z координат, столбец “column” из 5 битов констант x и z координат, а также полосу “lane” для хранения слова w, x и y координат. Двумерная часть содержит следующие элементы: лист “sheet” из 5w битов с постоянной координатой x, плоскость “plane” из 5w битов с постоянной координатой y и часть “slice” набор из 25 битов с постоянной координатой z.

Рассмотрим подробнее функцию f(). Функция f() состоит из следующих 5 операций: тета-функция (Theta-Function), чи-функция (Chi-Function), пи-функция (Pi-Function), ро-функция (Rho-Function) и йота-функция (Iota-Function).

Функция хэширования Grostl. Функция хэширования Grostl способна возвращать хэш-значение произвольной длины от 1 до 64 байт, то есть от 8 до 512 бит, при этом само хэш-значение должно быть кратно байту. Вариант функции, возвращающий n бит принято называть Grostl-n. Функция работает следующим образом. Исходное сообщение M дополняется и разбивается на блоки m_1 ... m_t по l бит каждый. Дальше эти блоки последовательно обрабатываются. Начальное l-битное значение h_0 приравнивается к значению вектора инициализации IV, после чего все блоки обрабатываются следующим образом:

$$h_i \leftarrow f(h_{i-1}, m_i) \text{ для всех } i=1, ..., t$$

Заметим, что функция f преобразует два входных сообщения по l бит каждое в одно выходное значение длиной l бит. Первый вход называется связующим входом, второе сообщение является обрабатываемым сообщением M. Для варианта функции Grostl, возвращающей 256-битовое хэш-значение, параметр l равен 512, для больших длин хэш-значений, параметр l равен 1024.

После того, как будет обработан последний блок сообщения M, выходное хэш-значение вычисляется как

$$H(M) = \Omega(h_t).$$

В общем виде схему выработки хэш-начения с помощью функции Grostl можно представить так, как это сделано на рис. 6.

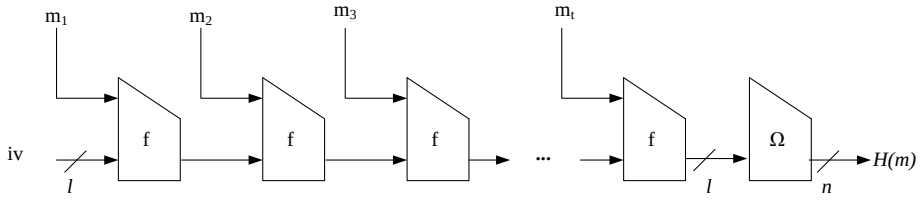

Рис. 6. Общий фид преобразования с помощью функции Grostl

Сжимающая функция базируется на двух l-битовых перестановках P и Q. В общем виде преобразование можно описать с помощью следующей формулы

$$f(h,m) = P(h \oplus m) \oplus Q(m) \oplus h.$$

Схематично данное преобразование представлено на рис. 7.

Выходное преобразование Ω можно описать с помощью формулы:

$$\Omega(h) = trunc_n(P(x) \oplus x),$$

где $trunc_n[x]$ – операция, которая отбрасывает все, кроме последних n битов сообщения x. Схематично преобразование Ω можно представить в виде рис. 8.

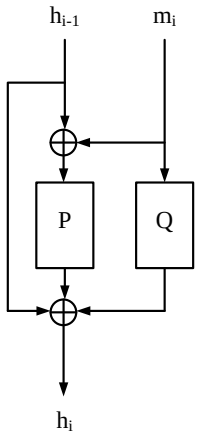

Рис. 7. Сжимающее преобразование

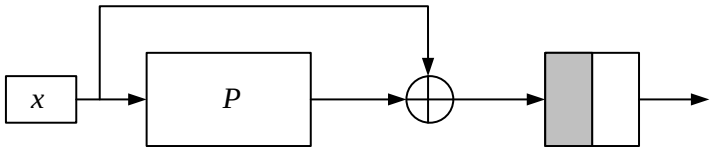

Рис. 8. Заключительная операция сжатия

Функция хэшированя JH. Автором функции хэширования JH является Хонгджун Ву (Hongjun Wu) из Института исследований в области телекоммуникаций, Сингапур. Алгоритм построен по схеме Меркля-Дамгарда.

Сообщение, для которого необходимо выработать хэш-функцию разбивается на блоки по 512 бит. Получаемое на выходе хэш-значение может иметь размер 224, 256, 384 и 512 бит. При этом в функции хэширования JH используется один и тот же алгоритм сжатия, формирующий после обработки последнего блока сообщения 1024-битное значение. Данное 1024-битное значение в последствии усекается до требуемого размера хэш-значения.

На вход функции сжатия поступают 512-битные блоки сообщения, а также 1024-битное выходное значение функции сжатия после обработки предыдущего блока H_{i-1}. Для обработки первого блока вместо H_{i-1} используется значение вектора инициализации IV. Функция сжатия выполняет следующие действия. Обрабатываемый блок сообщения M_i складывается по модулю два с левой 512-битной половиной значения H_{i-1}. Результат предыдущей операции обрабатывается функцией преобразования E. Сообщение M_i складывается по модулю два с правой половиной 1024-битного выходного значения функции E. В результате получается значение H_i.

Выходное 1024-битное значение функции E представляется в виде восьмимерного массива, каждое измерение которого содержит 2 слова по 4 бита. В основе E функции лежит блочный шифр, который представляет собой SPN-сеть (сеть на основе подстановок и перестановок) и состоит из 35 раундов. В каждом раунде используются: замена с помощью двух S-блоков S0 или S1; линейное преобразо-

вание, поочередно обрабатывающее по два слова состояния с помощью операций XOR над определенными битами входных слов; перестановка слов состояния по определенному закону.

Результаты анализа функций хэширования. В соответствии с документацией по конкурсу SHA-3 все пять финалистов подвергались тщательному анализу с различных точек зрения. Основные результаты анализа приведены в таблицах 2 - 3 [5].

Таблица 2

Лучшие известные атаки по поиску коллизий для финалистов конкурса SHA-3

Алгоритм	Тип атаки	Цель	Число раундов	Процент взлома, %
BLAKE	Semi-free-start near collision	Сжимающая функция	4/14	29
Grostl	Semi-free-start near collision	Сжимающая функция	6/10	60
JH	Near collision	Функция хэширования	26/42	62
Keccac	Semi-free-start near collision	Сжимающая функция	5/24	21
Skein	Collision	Функция хэширования	32/72	44

Таблица 3

Лучшие известные атаки с использованием дифференциальных свойств для финалистов конкурса SHA-3

Алгоритм	Цель	Число раундов	Процент взлома, %
BLAKE	Блочный шифр	7/14	50
Grostl	Перестановка	9/10	90
JH	Сжимающая функция	42/42	100
Keccac	Перестановка	14/24	58
Skein	Сжимающая функция	37/72	52

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Xiaoyun Wang, Hongbo Yu., Yiqun Lisa Yin*, Efficient Collision Search Attacks on SHA-0 [Электронный ресурс]. – http://citeseerx.ist.psu.edu, свободный.
2. *XiaoyunWang, Hongbo Yu.,* How to BreakMD5 and Other Hash Functions [Электронный ресурс]. – http://citeseerx.ist.psu.edu, свободный.
3. *XiaoyunWang, Yiqun Lisa Yin, Hongbo Yu.,* Finding Collisions in the Full SHA-1 [Электронный ресурс]. – http://people.csail.mit.edu/yiqun/pub.htm, свободный.
4. *Guido Bertoni, Joan Daemen, Michaël Peeters, Gilles Van Assche*. The Keccak reference, Version 3.0, January 14, 2011. – С. 1-69.
5. *Shu-jen Chang, Ray Perlner, William E. Burr* et all Third-Round Report of the SHA-3 Cryptographic Hash Algorithm Competition [Электронный ресурс]. – http://nvlpubs.nist.gov/nistpubs/ir/2012/NIST.IR.7896.pdf, свободный.

Статью рекомендовал к опубликованию д.т.н., профессор Я.Е. Ромм.

Бабенко Людмила Климентьевна – Федеральное государственное автономное образовательное учреждение высшего профессионального образования «Южный федеральный университет»; e-mail: blk@fib.tsure.ru; 347928, г. Таганрог, ул. Чехова, 2, корпус "И"; тел.: 88634312018; кафедра безопасности информационных технологий; профессор.

Ищукова Евгения Александровна – e-mail: jekky82@mail.ru; тел.: 88634371905; кафедра безопасности информационных технологий; доцент.

Babenko Lyudmila Klimentevna – Federal State-Owned Autonomy Educational Establishment of Higher Vocational Education “Southern Federal University”; e-mail: blk@fib.tsure.ru; Block “I”, 2, Chehov street, Taganrog, 347928, Russia; phone: +78634312018; the department of security of information technologies; professor.

Ischukova Evgeniya Aleksandrovna – e-mail: jekky82@mail.ru; phone: +78634371905; the department of security of information technologies; associate professor.