ISSN 1816-0301 (Print)
ISSN 2617-6963 (Online)
УДК 004.021
https://doi.org/10.37661/1816-0301-2020-17-1-109-118

Поступила в редакцию 01.10.2019
Received 01.10.2019

Принята к публикации 04.02.2020
Accepted 04.02.2020

Алгоритм хеширования на основе SHA-3 с использованием хаотических отображений

А. В. Сидоренко✉, М. С. Шишко

Белорусский государственный университет, Минск, Беларусь
✉E-mail: sidorenkoa@yandex.ru

Аннотация. Описан алгоритм хеширования данных, основанный на методе хеширования SHA-3 (Secure Hash Algorithm-3). Для увеличения производительности при сохранении безопасности хеширования в алгоритме использованы хаотические отображения. Проведено тестирование исходного и модифицированного алгоритмов на устойчивость к коллизиям, которое показало малую вероятность коллизий. Сделан статистический анализ выходных последовательностей, а также производительности алгоритмов. Проведено тестирование алгоритма с помощью набора статистических тестов SP 800-22, которое показало, что двоичная последовательность, генерируемая предложенным алгоритмом, близка к случайной. Протестирована также производительность алгоритма: скорость хеширования модифицированного алгоритма увеличилась на 60 % по сравнению со скоростью хеширования обычного SHA-3.

Ключевые слова: хеширование, шифрование, динамический хаос, лавинный эффект, статистический криптоанализ

Для цитирования: Сидоренко, А. В. Алгоритм хеширования на основе SHA-3 с использованием хаотических отображений / А. В. Сидоренко, М. С. Шишко // Информатика. – 2020. − Т. 17, № 1. – С. 109–118. https://doi.org/10.37661/1816-0301-2020-17-1-109-118

Hashing technique based on SHA-3 using chaotic maps

Alevtina V. Sidorenko✉, Maksim S. Shishko

Belarusian State University, Minsk, Belarus
✉E-mail: sidorenkoa@yandex.ru

Abstract. New hashing technique based on SHA-3 (Secure Hash Algorithm-3) is introduced. Chaotic maps are used in this technique to enhance performance without losing security. Introduced algorithm was tested for resistance against collisions, statistical analysis of output sequences was performed, hashing performance was evaluated. The testing showed a low collision probability. The testing corresponds the standards of National Institute of Standards and Technology and showed that output sequences are close to random. Performance testing showed 60 % enhancement in comparison with plain SHA-3.

Keywords: hashing, encryption, chaos, avalanche effect, statistical cryptanalisys

For citation: Sidorenko A. V., Shishko M. S. Hashing technique based on SHA-3 using chaotic maps. *Informatics*, 2020, vol. 17, no. 1, pp. 109–118 (in Russian). http://doi.org/10.37661/1816-0301-2020-17-1-109-118

Введение. В последнее время автономные роботы находят все большее применение в решении самых разнообразных задач, таких как ликвидация последствий чрезвычайных ситуаций, охрана территории, проведение медицинских операций, разведка. Для решения поставленной задачи робот оснащается разного рода средствами наблюдения, датчиками, навигационным и иным оборудованием. Данные, полученные с помощью такого оборудования, могут обраба-

тываться бортовым компьютером робота либо передаваться для обработки в пункт управления. В любом случае у робота должен быть способ общения с пунктом управления для получения команд и передачи полезной информации. Проводные технологии передачи данных для этого подходят плохо, так как любые провода существенно уменьшают мобильность робота.

Одним из наиболее перспективных направлений современной робототехники является разработка роботов, способных работать в группе, – так называемых роевых роботов [1, 2]. Их поведение должно быть коллективным для эффективного выполнения задачи. Каждый робот должен четко определять положение в пространстве, ориентацию, вектор направления движения и подзадачу, выполняемую другими участниками роя для того, чтобы корректировать свои действия. Компрометация и подлог информации даже одного участника роя приведут к дестабилизации всей системы и невозможности выполнять поставленную задачу. Поэтому обеспечение безопасности обмена информацией между участниками роя является очень важной задачей.

Совместные методы борьбы с неизвестными средами или синхронизация между различными группами роя способствуют достижению в области проектирования и изготовления таких аппаратных средств, как материнские платы Raspberry Pi 8 или Intel Galileo 9 (URL: https://www.raspberrypi.org/documentation/hardware/computemodule/datasheets/rpi_DATA_CM3plus_1p0.pdf). Они позволяют в настоящее время роботам обеспечивать передачу информации с увеличением возможностей ее обработки в устройствах связи с малой мощностью.

Существенным препятствием для широкомасштабного применения роботов в коммерческих приложениях является невозможность обеспечить их безопасность. В некоторых исследованиях подчеркнута необходимость разработки систем, в которых члены роя могут распознавать своих коллег и доверять им [3–5]. Ошибочное или злонамеренное включение новых членов роя может представлять потенциальный риск для целей роя, а также нарушать его безопасность.

Безопасность в любой информационной среде, включая роботизированные роевые системы, принципиально связана с предоставлением основных услуг, таких как конфиденциальность и целостность данных, аутентификация объектов и источника данных. Роботизированные роевые системы испытывают недостаток практических решений этих проблем. Теме безопасности в современных исследованиях уделяется недостаточное внимание в основном из-за сложных и гетерогенных характеристик роботизированных систем. Технология блокчейна [6] может обеспечить не только надежный peer-to-peer канал связи для агентов роя, но и способ преодоления потенциальных угроз, уязвимостей и атак.

Блокчейн – это новая технология, возникшая в поле биткоинов и демонстрирующая, что с помощью объединения одноранговых сетей с криптографическими алгоритмами группа агентов может достичь соглашения по конкретному положению дел и зафиксировать это соглашение без необходимости контролирующего органа. Комбинация блокчейна с другими распределенными системами, такими как роботизированные роевые системы, может предоставить необходимые возможности для того, чтобы сделать операции внутри роботизированного роя более безопасными, автономными и гибкими [7, 8]. Блокчейн является, по сути, общедоступной хронологической базой данных транзакций, записанных сетью агентов. Отдельные транзакции содержат сведения о том, кто и кому отправил сообщение. Данные сгруппированы в наборы, называемые блоками.

Каждый блок содержит информацию об определенном количестве транзакций (рис. 1), ссылку на предыдущий блок в цепочке блоков и ответ на сложную математическую задачу, известную как «доказательство работы». Концепция доказательства работы используется для проверки данных, связанных с этим конкретным блоком, а также для того, чтобы сделать разбиение на блоки вычислительно «жестким», тем самым не позволяя злоумышленникам изменять блок-цепочку в свою пользу. Она основана на криптографических вычислениях, в частности вычислении значений хеш-функции, которые дают непредсказуемые числовые последовательности. Блокчейн инкапсулирует все транзакции внутри блока в цифровом отпечатке, которым и является хеш. Любые различия во входных данных (порядке транзакций, количестве получателей и полезной информации и т. д.) будут приводить к различиям в выходных

данных (доказательстве работы), и таким образом будет получен другой цифровой отпечаток. Следовательно, одной из ключевых частей технологии блокчейна является хеш-функция.

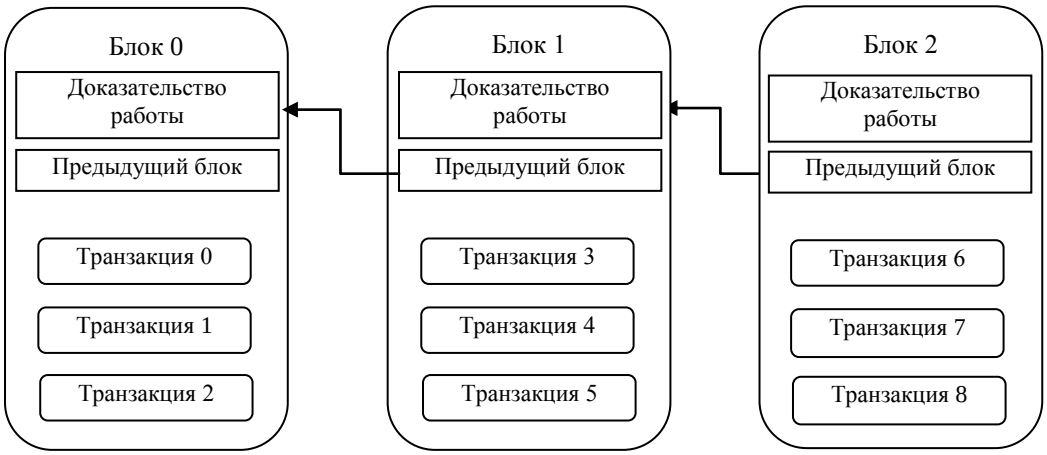

Рис. 1. Простая цепочка блоков блокчейна

Хеш-функции. Существуют два основных типа хеш-функций [9]: ориентированные на данные и ориентированные на безопасность (рис. 2). Хеш-функции, ориентированные на данные, используются в системах, работающих с большими объемами данных для ускорения их поиска, сравнения и выдачи. Они также подразделяются на не зависящие от данных и зависящие от данных хеш-функции. Не зависящие от данных функции хеширования не используют данные для вычисления хеш-суммы. Не зависящие от данных методы хеширования не хранят информацию об обработанных данных для оценки качества хеширования. Часто хеширующие функции являются предопределенными, хотя некоторые из них могут изучать распределения данных для улучшения результатов хеширования, таких как чувствительность к местоположению. Не зависящие от данных хеширующие функции можно разделить на четыре класса, основанные на следующих режимах: случайная проекция, локально-чувствительная проекция, обучающееся хеширование и структурированная проекция.

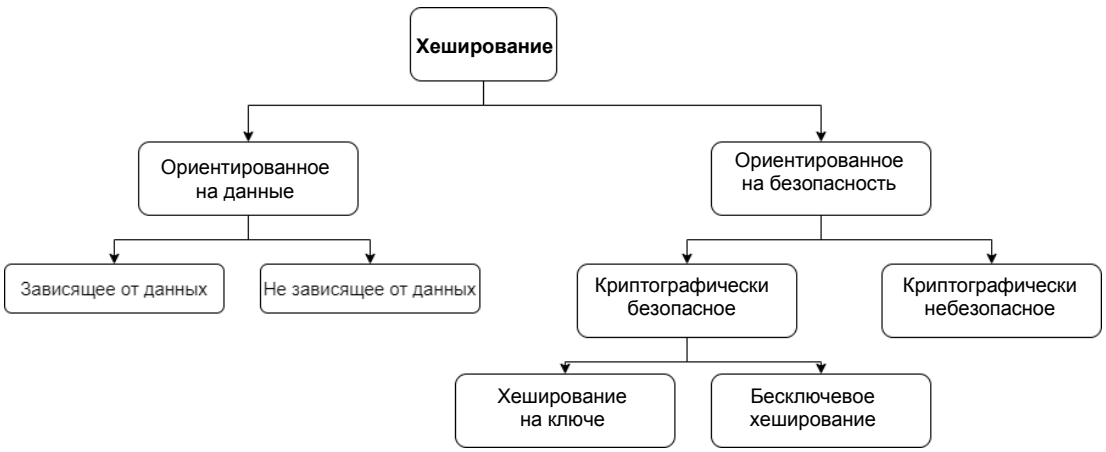

Рис. 2. Классификация хеш-функций

Зависящее от данных хеширование изучает хеширующие функции на основе набора данных, заданного для обучения, поэтому функции хеширования могут найти лучшие компактные коды для всех типов данных. Поскольку зависящие от данных методы хеширования очень чувствительны к базовым данным, они отличаются более быстрым временем запроса и меньшим по-

треблением памяти. Чтобы сохранить информацию о локальности и добиться лучшей избирательности, зависящее от данных хеширование должно точно соответствовать распределению данных в пространстве функций, однозначно определяя семейство хеширующих функций для этого набора обучающих данных. Кроме того, зависящее от данных хеширование обычно рассматривает сходство с особенностями обучающих данных.

Хеширование, ориентированное на безопасность, относится к методам, которые применяются для проверки целостности или аутентификации данных. Поскольку коды хеширования, ориентированного на безопасность, часто вычисляются намного дольше, чем коды хеширования, ориентированного на данные, хеш-таблица, как правило, не требуется или не может поддерживаться. Методы данной категории в первую очередь ориентированы на проблемы, поэтому они часто являются дорогостоящими и менее эффективными по сравнению с методами хеширования, ориентированного на данные.

Криптографически безопасное хеширование (криптографическое хеширование) относится к методам, хеширующая функция которых является односторонней, т. е. по значению хеш-суммы невозможно восстановить данные. При использовании таких методов длина ввода (называемая также «сообщение») является произвольной, а размер вывода (называемый также «дайджест сообщения») фиксирован. Хеш-результаты фиксированного размера применяются в качестве подписи в целях представления исходного сообщения для проверки. В связи с такой чувствительностью к безопасности криптографическое хеширование должно иметь строгий лавинный эффект, который заключается в значительном изменении хеш-выхода (примерно половины выходных битов), если есть даже незначительное изменение на входе (например, одного бита). Для использования в блокчейне важны именно эти особенности для подтверждения подлинности данных.

Хотя криптографически безопасное хеширование обладает хорошими свойствами безопасности, оно часто проводится неэффективно. Для приложений без серьезных проблем безопасности есть более простой механизм хеширования, называемый криптографически небезопасным хешированием или некриптографическим хешированием, который является более практичным. Для некриптографического хеширования, такого как хеш-функция Fowler-Noll-Vo (FNV), основная цель по-прежнему заключается в создании хеш-вывода для проверки, но процесс хеширования не должен учитывать криптографию. В результате становится возможной более быстрая обработка, более низкая вероятность коллизий, более высокая вероятность обнаружения небольших ошибок и более легкое обнаружение коллизий по сравнению с криптографически безопасным хешированием. Этот метод хеширования особенно популярен в приложениях, требующих быстрого поиска или обработки данных.

На данный момент известно множество различных методов хеширования. Стандарты этих методов разрабатываются научным сообществом и выбираются после однораноговых исследований Национальным институтом стандартов и технологий (англ. The National Institute of Standards and Technology, NIST), США. Одной из наиболее широко применяемых хеш-функций является SHA-1, которая используется в большом количестве приложений и протоколов безопасности сети Интернет.

Между тем в 2004 г. хеш-функции MD и SHA-0 были взломаны. Последующая атака на SHA-1 потребовала всего 2^{69} операций (CRYPTO-200), т. е. оказалась в 2000 раз быстрее, чем атака brute force (потребовала 2^{80} операций). Даже если на обычных компьютерах все еще сложно реализовать 2^{69} операций, такой результат, основанный на предыдущей атаке на SHA-0, является очень важным, поскольку варианты SHA-2 алгоритмически близки к SHA-1 и в конечном итоге производят дайджесты сообщений на принципах, аналогичных алгоритмам дайджестов сообщений MD4 и MD5.

В настоящее время необходим новый стандарт хеша, основанный на оригинальных подходах, должны быть найдены новые хеш-функции или улучшены уже существующие.

Предлагаемый алгоритм хеширования на основе SHA-3. В настоящей работе выполняется модификация хеш-функции SHA-3 (URL: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf), когда к существующей хеш-функции для укрепления ее свойств и увеличения производительности добавляется компонент хаоса.

Алгоритм SHA-3 использует функцию Keccak [10] для перестановки бит внутреннего состояния. Функция Keccak описывается как набор операций перестановки, выполненных над трехмерным массивом бит A, вид и внутренняя индексация которого показаны на рис. 3. Параметры функции обозначаются буквами греческого алфавита θ, ρ, π, χ, ι.

Функция $A' = \theta(A)$ суммирует по модулю два каждый бит внутреннего состояния с каждым битом из двух смежных столбцов (рис. 4):

1. Для всех пар (x, z), $0 \le x < 5$, $0 \le z < w$, $C[x, z] = A[x, 0, z]$ XOR $A[x, 1, z]$ XOR $A[x, 2, z]$ XOR $A[x, 3, z]$ XOR $A[x, 4, z]$.
2. Для всех пар (x, z), $0 \le x < 5$, $0 \le z < w$, $D[x, z] = C[(x - 1) \bmod 5, z]$ XOR $C[(x + 1) \bmod 5, (z - 1) \bmod w]$.
3. Для всех троек (x, y, z), $0 \le x < 5$, $0 \le y < 5$, $0 \le z$, $A'[x, y, z] = A[x, y, z]$ XOR $D[x, z]$.

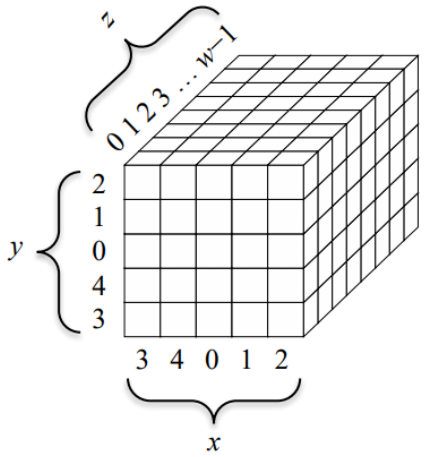

Рис. 3. Общий вид и индексация внутреннего состояния Keccak

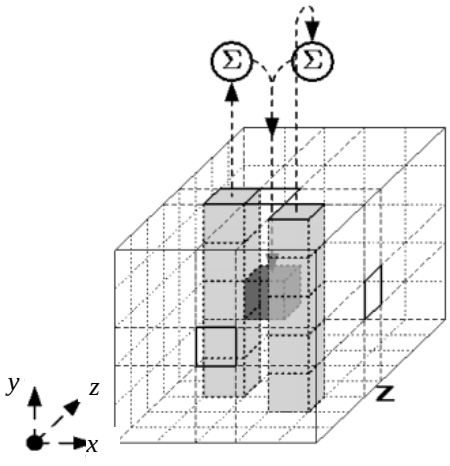

Рис. 4. Графическое представление функции θ(A)

Функция $A' = \rho(A)$ циклически сдвигает строки внутреннего состояния на различное количество бит в зависимости от номера строки (рис. 5).

Для всех z, $0 \le z < w$, $A'[0, 0, z] = A[0, 0, z]$. Пусть $(x, y) = (1, 0)$. Тогда для всех t от 0 до 23 выполняется соотношение: для всех z, $0 \le z < w$, $A'[x, y, z] = A[x, y, (z - (t + 1)(t + 2) / 2) \bmod w]$, $(x, y) = (y, (2x + 3y) \bmod 5)$.

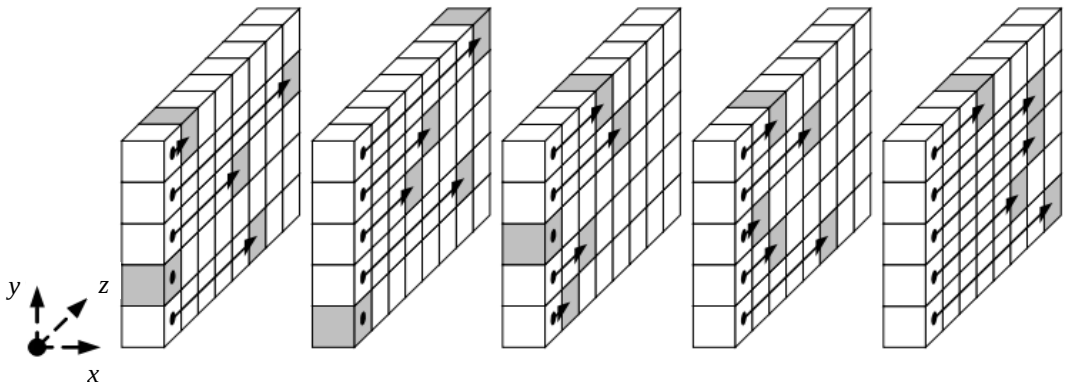

Рис. 5. Графическое представление функции ρ(A)

Функция $A' = \pi(A)$ производит перестановку бит в пределах битовой плоскости по следующему правилу: для каждой тройки (x, y, z), $0 \le x < 5$, $0 \le y < 5$, $0 \le z < w$, выполняется соотношение

$$A'[x, y, z] = A[(x + 3y) \bmod 5, x, z].$$

Функция $A' = \chi(A)$ суммирует по модулю два каждый бит с нелинейной функцией двух других бит той же строки:

$$A'[x, y, z] = A[x, y, z] \text{ XOR } ((A[(x + 1) \bmod 5, y, z] \text{ XOR } 1) * A[(x + 2) \bmod 5, y, z]).$$

Функция $A' = \iota(A)$ модифицирует некоторые биты первой строки в зависимости от текущего раунда, остальные строки при этом остаются неизменными.

В конечном итоге один раунд i_r функции Keccak представляет собой последовательное применение всех пяти функций:

$$\text{Rnd}(A, i\_r) = \iota(\chi(\pi(\rho(\theta(A))))), i_r).$$

В настоящей работе предлагается улучшить алгоритм перестановок, используемый в SHA-3, путем добавления хаотичности в функцию перестановок. Для этого следует модифицировать функцию перестановки Keccak для увеличения лавинного эффекта с помощью хаотического отображения. В оригинальной функции Keccak на шагах ρ и π перестановка элементов промежуточного состояния является предопределенной. Авторы предлагают выбирать индекс элемента для перестановки с помощью хаотического отображения на основании входных данных.

В качестве хаотического отображения используется логистическое отображение λ, которое имеет вид

$$\lambda(x_n) = Mx_n(1 - x_n),$$

где x_n принимает значения от 0 до 1. Параметр M означает скорость размножения (роста популяции). При значениях параметра от 3,57 до 4 система проявляет хаотическое поведение. Применение данного отображения позволяет уменьшить количество раундов перестановки до 12 без потери качества хеширования, что увеличивает производительность алгоритма.

Функция $\rho(A)$ принимает следующий вид: для всех z, $0 \le z < w$, $A'[0, 0, z] = A[0, 0, z]$, $(x, y) = (1, 0)$. Тогда для всех t от 0 до 23 выполняется соотношение: для всех z, $0 \le z < w$, $A'[x, y, z] = A[x, y, [w * \lambda(A[x, y, z] / 2^w)]]$, $(x, y) = (y, (2x + 3y) \bmod 5)$.

Функция $\pi(A)$ принимает следующий вид: для каждой тройки (x, y, z), $0 \le x < 5$, $0 \le y < 5$, $0 \le z < w$, выполняется соотношение

$$A'[x, y, z] = A[\lfloor 5 * \lambda(A[a, y, z] / 2^w) \rfloor, x, z].$$

Тестирование алгоритмов. Для оценки качества модифицированной хеш-функции были проведены следующие тесты: на лавинный эффект, на устойчивость к атаке «дней рождения», тестирование свойств выходной последовательности по методике NIST.

Лавинный эффект – одно из важнейших свойств, которыми должна обладать криптографическая хеш-функция. Он проявляется в том, что изменение значения малого количества битов во входном тексте ведет к «лавинному» изменению значений выходных битов хеш-суммы. Если криптографический алгоритм не обладает лавинным эффектом в достаточной степени, криптоаналитик может сделать предположение о входной информации, основываясь на выходной. Таким образом, достижение лавинного эффекта является важной целью при разработке криптографического алгоритма.

Для тестирования лавинного эффекта описанной выше хеш-функции был составлен словарь из 450 000 уникальных сообщений различной длины. Для каждого сообщения была вычислена хеш-сумма, затем в исходном сообщении изменен один или несколько случайных бит и уже для измененного сообщения рассчитана хеш-сумма. Далее хеш-суммы исходного и измененного

сообщений сравнивались при помощи расстояния Хемминга. В табл. 1 показано распределение количества измененных бит хеша при изменении одного бита сообщения для алгоритмов SHA-2 и SHA-3. Для наглядности эти данные представлены в виде гистограммы (рис. 6), на которой видно, что предлагаемый алгоритм имеет распределение, схожее с SHA-2 и SHA-3.

Таблица 1

Численные значения изменения количества бит
при тестировании лавинного эффекта для трех алгоритмов

Количество бит	Алгоритм хеширования		
	SHA-2	SHA-3 Keccak	SHA-3 Chaotic Keccak
<80	0,0	0,0	0,0
88	1,0	3,0	1,0
96	62,0	52,0	59,0
104	962,0	987,0	1009,0
112	6506,0	6586,0	6614,0
120	17 630,0	17 506,0	17 463,0
128	18 529,0	18 537,0	18 527,0
136	7831,0	7870,0	7799,0
144	1342,0	1309,0	1383,0
152	71,0	82,0	79,0
>160	1,0	3,0	1,0

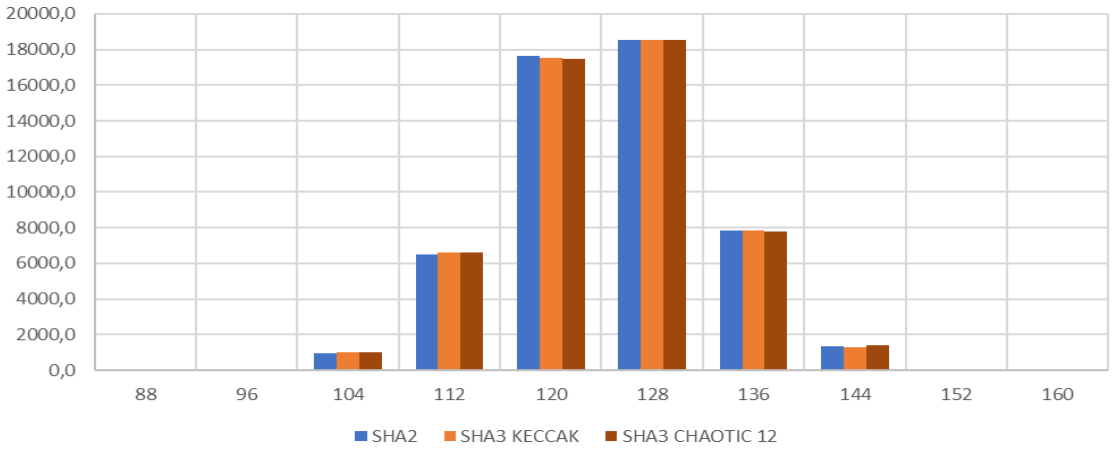

Рис. 6. Гистограмма изменения количества бит при тестировании лавинного эффекта

Затем алгоритм был протестирован на стойкость к атаке «дней рождения». Данная атака основана на парадоксе дней рождения, который заключается в том, что в группе, состоящей из 23 или более человек, вероятность совпадения дней рождения хотя бы у двух человек превышает 50 %. Суть метода состоит в значительном уменьшении количества передаваемых хеш-функции аргументов, необходимых для обнаружения коллизии. Это связано с тем, что если хеш-функция генерирует n-битное значение, то число случайных аргументов хеш-функции, для которого с большой вероятностью будет обнаружена хотя бы одна коллизия хеш-функции (т. е. найдется хотя бы одна пара равных хеш-кодов, полученных на разных аргументах), равно не 2^n, а только около $2^{n/2}$.

Для оценки стойкости хеш-функции к атакам «дней рождения» был использован тот же словарь, что и для проверки лавинного эффекта. Однако теперь был проведен поиск коллизий хеш-сумм для сообщений из словаря, который коллизий не выявил.

Пакет тестов NIST STS был разработан отделом безопасности компьютерных технологий NIST. Полное описание пакета тестов, рекомендации по выбору методики исследований

и интерпретации результатов приведены в публикации (URL: https://nvlpubs.nist.gov/ nistpubs/ legacy/sp/nistspecialpublication800-22r1a.pdf). Последняя на момент исследований версия 2.2.1 пакета есть в открытом доступе. Применительно к исследуемой хеш-функции данный пакет тестов позволяет определить, насколько она статистически безопасна и устойчива к возникновению коллизий.

Всего пакет NIST содержит 15 тестов, однако некоторые из них являются сложными и фактически выборки с последовательностями проходят 188 тестов. Общая структура отдельно взятого теста для каждой последовательности имеет следующий вид:

1. Выдвигаются нулевая гипотеза H_0 (предположение о том, что данная двоичная последовательность S_q случайная) и альтернативная гипотеза H_a (последовательность неслучайная).

2. По последовательности S_q и соответствующим тесту алгоритмам рассчитывается значение статистики теста S.

3. С использованием специальной функции и значения статистики теста рассчитывается значение вероятности $P \in [0, 1]$, которая суммирует силу доказательств против нулевой гипотезы.

4. Значение вероятности P сравнивается с уровнем значимости, который выбирается, как правило, в интервале $\alpha \in [0{,}001;\ 0{,}01]$ и отражает вероятность ошибок первого рода (неправильного отвергания «хорошей» (прошедшей тест) последовательности). Если $P < \alpha$, гипотеза H_0 отвергается и принимается альтернативная гипотеза H_a. В противном случае принимается гипотеза H_0 (последовательность случайная).

В соответствии с рекомендациями NIST тестирование генераторов последовательностей (в рассматриваемом случае – алгоритмов хеширования) с применением пакета тестов в общем случае должно включать ряд этапов:

1. Формирование наборов последовательностей для тестирования. Для исходной версии метода были сформированы шесть выборок из 100 последовательностей длиной по 104 857 600 бит в каждой, т. е. каждая выборка содержала 1 048 576 100 бит. Выборки формировались путем вычисления хеш-значений сообщений длиной по 256 байт для псевдослучайных наборов сообщений, генерируемых линейным конгруэнтным генератором из библиотеки на языке C++.

2. Тестирование выборок пакетом. Для тестирования применялись все 15 (188) тестов пакета с предлагаемыми по умолчанию параметрами. Порядок тестирования последовательностей из выборок был описан выше.

3. Оценку значений вероятности P. Анализ сформированных значений P позволяет вскрыть конкретные дефекты тестируемых последовательностей и хеш-функции в целом.

4. Оценку прохождения теста(ов) последовательностями в выборках путем сравнения значений вероятности P с выбранным уровнем значимости. На этом этапе также вычисляется процент прошедших тест последовательностей в выборке.

5. Интерпретацию полученных результатов. NIST предлагает два основных подхода (которые, однако, не являются исчерпывающими) к оценке качества последовательностей и прохождению теста всей выборкой. Первый подход основан на оценке равномерности распределения значений вероятности P для последовательностей из выборки, второй – на оценке процента прошедших тест последовательностей из выборки. При представлении результатов исследований примем второй подход в качестве основного. Для успешного прохождения каждого отдельного теста всей тестируемой выборкой в соответствии с выбранными по умолчанию параметрами тестирования необходимо, чтобы процент хороших последовательностей в выборке был не менее 96.

В целом при оценке последовательностей и интерпретации результатов могут быть сделаны следующие выводы:

– тестирование не показало отклонений от случайности;

– тестирование четко указывает на отклонение от случайности;

– тестирование безрезультатно.

В табл. 2 приведены результаты тестирования. Модифицированный SHA-3 прошел все тесты, кроме универсального теста Маурера, поэтому последовательность, выдаваемую данным алгоритмом, можно считать случайной.

Таблица 2

Результаты тестирования набором статистических тестов NIST

Тест	Алгоритм хеширования		
	SHA-2	SHA-3	Chaotic SHA-3
Frequency	0,971 699	0,455 937	0,816 537
Block Frequency	0,080 519	0,455 937	0,971 699
Runs	0,935 716	0,946 308	0,213 309
Longest Run	0,334 538	0,924 076	0,262 249
Rank	0,090 936	0,699 313	0,616 305
DFT	0,001 895	0,514 124	0,191 687
Non-Owerlaping Template	0,678 686	0,474 986	0,224 821
Owerlaping Template	0,090 936	0,437 274	0,262 249
Universal	0	0	0
Linear Complexity	0,911 413	0,834 308	0,534 146
Serial	0,000 082	0,000 003	0,494 392
Entropy	0,275 709	0,007 566	0,911 413
Cumulative Sums	0,145 326	0,115 387	0,494 392
Всего пройдено	*11*	*10*	*12*

В работе также была проведена оценка производительности трех алгоритмов. Тестирование проводилось на персональном компьютере под управлением Windows 10 с процессором Intel core i-5 2330M. Самую высокую производительность показал SHA-2 – 77,1 Мб/с, SHA-3 показал наихудшую производительность – 16 Мб/с. Производительность же модифицированного SHA-3 оказалась 26 Мб/с, что на 60 % больше, чем производительность обычного SHA-3.

Заключение. Разработан алгоритм хеширования на основе SHA-3 с использованием хаотических отображений. С целью увеличения производительности в качестве хаотического отображения в функции перестановки используется логистическое отображение.

Для оценки качества модифицированной хеш-функции были проведены тест на лавинный эффект, тест на устойчивость к атаке «дней рождения» и тестирование свойств выходной последовательности по методике NIST. Результаты тестирования лавинного эффекта очень схожи с результатами тестирования SHA-2 и SHA-3 и показывают, что примерно половина битов хеша меняется при изменении одного бита в хешируемых данных. Атака «дней рождения» коллизий не выявила. По результатам тестирования пакетом статистических тестов NIST алгоритм SHA-3 с модифицированной функцией перестановок прошел все тесты, кроме универсального теста Маурера, поэтому последовательность, выдаваемую данным алгоритмом, можно считать случайной.

References

1. Bayindir L. A review of swarm robotics tasks. *Neurocomputing*, 2016, vol. 172, pp. 292–321.

2. Navarro I., Matia F. An introduction to swarm robotics. *ISRN Robotics*, 2013, vol. 2013, pp. 1–10.

3. Higgins F., Tomlinson A., Martin K. M. Survey on security challenges for swarm robotics. *Fifth International Conference on Autonomic and Autonomous Systems, 20–25 April 2009, Valencia, Spain*. Valencia, 2009, pp. 307–312.

4. Priyadarshini I. *Cyber Security Risks in Robotics*, 2017. Available at: https://www.researchgate.net/publication/319354229_Cyber_security_risks_in_Robotics (accessed 21.07.2019).

5. Shah R. *Security Landscape for Robotics*, 2019. Available at: https://arxiv.org/abs/1904.03033v1 (accessed 21.07.2019).

6. Nakamoto S. *Bitcoin: a Peer-to-Peer Electronic Cash System*, 2008, Available at: https://bitcoin.org/bitcoin.pdf (accessed 21.07.2019).

7. Lopes V., Alexandre L. A. *An Overview of Blockchain Integration with Robotics and Artificial Intelligence*, 2018. Available at: https://arxiv.org/abs/1810.00329v1 (accessed 21.07.2019).

8. Ferrer E. C. *The Blockchain: a New Framework for Robotic Swarm Systems*, 2017. Available at: https://arxiv.org/abs/1608.00695v4 (accessed 21.07.2019).

9. Chi L., Zhu X. Hashing techniques: a survey and taxonomy. *ACM Computing Surveys*, 2017, vol. 50, no. 1, pp. 1–36. https://doi.org/10.1145/3047307

10. Bertoni G., Daemen J., Peeters M., Assche van G. *The Keccak Reference*, 2011. Available at: https://keccak.team/files/Keccak-reference-3.0.pdf (accessed 21.07.2019).

Информация об авторах

Сидоренко Алевтина Васильевна, доктор технических наук, профессор кафедры физики и аэрокосмических технологий, факультет радиофизики и компьютерных технологий, Белорусский государственный университет, Минск, Беларусь.
E-mail: sidorenkoa@yandex.ru

Шишко Максим Сергеевич, аспирант кафедры физики и аэрокосмических технологий, факультет радиофизики и компьютерных технологий, Белорусский государственный университет, Минск, Беларусь.
E-mail: maxshishko@yandex.ru

Information about the authors

Alevtina V. Sidorenko, Dr. Sci. (Eng.), Professor of Department of Physics and Aerospace Technology, Faculty of Radiophisics and Computer Technology, Belarusian State University, Minsk, Belarus.
E-mail: sidorenkoa@yandex.ru

Maksim S. Shishko, Postgraduate Student of Department of Physics and Aerospace Technology, Faculty of Radiophisics and Computer Technology, Belarusian State University, Minsk, Belarus.
E-mail: maxshishko@yandex.ru