УДК 004.056:004.424.47

Ю. В. Барышев, к. т. н.; А. О. Комаров

МЕТОДЫ РАСПАРАЛЛЕЛЕННОГО ХЕШИРОВАНИЯ, СТОЙКОГО К ОБЩИМ АТАКАМ

В работе проведен анализ современного состояния развития хеш-функций и атак на них. На основе этого анализа определено, что одними из наиболее опасных являются общие атаки на основе мультиколлизий. Для улучшения стойкости хеширования, которое будет использовать распараллеливание, предложен ряд конструкций хеширования и алгоритмов. Определены функции сжатия для методов хеширования на основе этих конструкций. Экспериментальные исследования позволили подтвердить увеличение стойкости предложенного метода хеширования к общим атакам на основе мультиколлизий.

Ключевые слова: *хеширование, распараллеленное хеширование, мультиколлизия, атака Жу, конструкция хеширования, функция сжатия.*

Введение

Много процессоров разрабатывают со встроенными двумя, четырьмя и больше ядрами. Учитывая это, программы должны максимально использовать эту возможность вычислительных платформ. Соответственно возникает необходимость в распараллеливании вычислений. На сегодняшний день все распараллеленные методы криптографического хеширования являются неустойчивыми к мультиколлизиям [1 – 4]. При этом устойчивость к мультиколлизиям не зависит от криптографических примитивов, используемых на каждой итерации во время хеширования, а зависит от используемых конструкций [3 – 5]. Соответственно возникает актуальная задача разработки методов хеширования, позволяющих использовать возможность распараллеливания вычислений с сохранением устойчивости хеш-функции к взлому.

Целью этой работы является увеличение стойкости методов хеширования, которые используют распараллеливание вычислений, к мультиколлизиям.

Для достижения цели выполнены такие задачи:

– проанализированы известные хеш-функции и атаки на них;

– разработаны стойкие к мультиколлизиям хеш-функции для распараллеленного хеширования;

– разработаны алгоритмы хеширования;

– разработаны средства, реализующие данные алгоритмы.

Анализ известных конструкций хеширования

Дамгард и Меркль независимо один от другого предложили теоремы, которые показывают, что если существует стойкая к коллизиям функция сжатия для входящих данных постоянной длины $f(\cdot)$: $\{0, 1\}^b \times \{0, 1\}^t \rightarrow \{0, 1\}^t$, то можно спроектировать стойкую к коллизиям функцию сжатия для входящих данных сменной длины $h(\cdot)$: $\{0, 1\}^* \rightarrow \{0, 1\}^t$ через итеративные вызовы функции сжатия $f(\cdot)$. Таким образом, если функция сжатия $f(\cdot)$ уязвима к некоторой атаке, то и итерированная хеш-функция $h(\cdot)$ тоже будет уязвима к атакам, но в целом противоположный результат не является верным [1 – 3].

Конструкция хеширования, предложенная Дамгардом и Мерклем, выглядит следующим образом:

$$h_i = f(h_{i-1}, m_i), \tag{1}$$

где h_i – промежуточное хеш-значение, полученное после обработки i-го блока данных; m_i – i-й блок данных.

Существует много похожих друг на друга конструкций, использующих блочный шифр как функцию сжатия. Одной из наиболее распространенных конструкций такого вида является конструкция Дэвиса – Мейера. Для определенного блочного шифра на основе ключа k $E_k(\cdot)$ конструкция хеширования выглядит так [1, 2, 6, 7]:

$$h_i = E_{m_i}(h_{i-1}) + h_{i-1} \tag{2}$$

Из конструкции (2) видно, что по сути данный класс конструкций аналогичный конструкции (1).

Люкс предложил конструкцию широкого канала и двойного канала, которая использует

две функции сжатия. Для конструкции широкого канала при $i \in [1; l]$ вычисляют:

$$h_i = f'(h_{i-1}, m_i),$$

$$h(M) = f''(h_{i-1}, h_l),$$

где исходные значения $f'(\cdot)$ имеют в два раза большую длину, чем $f''(\cdot)$ [5].

Данная конструкция имеет недостаток, связанный с тем, что для вычисления исходных значений функции сжатия $f'(\cdot)$ необходимо в два раза больше ресурсов по сравнению с $f(\cdot)$ в конструкции Меркля – Дамгарда.

Атаки на хеш-функции могут быть разделены на две категории: атаки методом прямого перебора и криптоаналитические атаки [1, 2, 4]. К атакам, не зависящим от алгоритма (прямой перебор, атака методом «дней рождений») уязвимы все алгоритмы. Единственная возможность им противодействовать – увеличить длину хеш-значения и ключа.

В соответствии с работой [2], атаки на хеш-функции классифицируют согласно схеме на рис. 1.

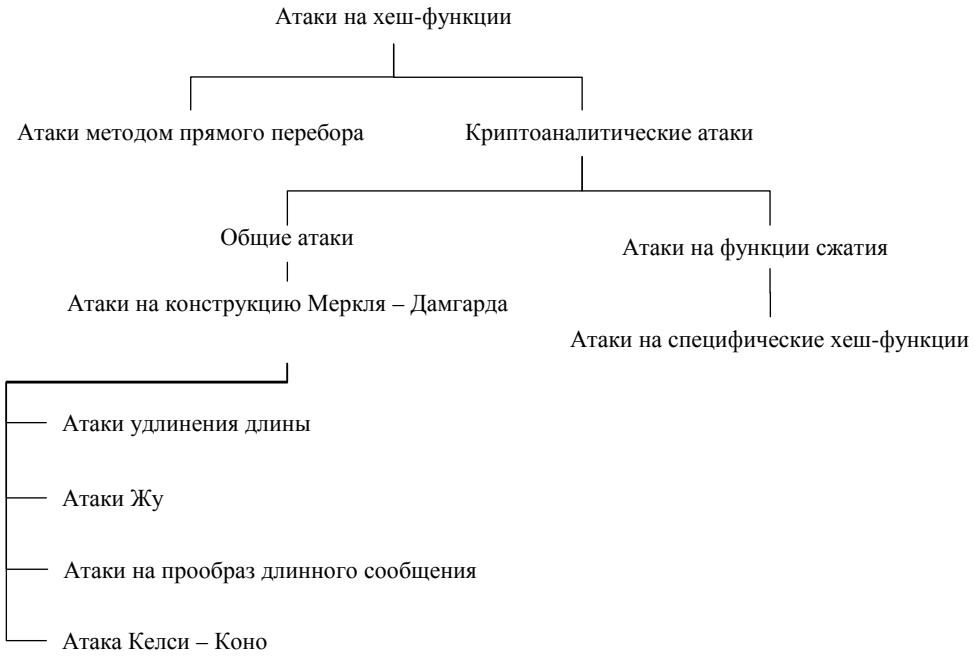

Рис. 1. Классификация атак по Гауравараму

Среди атак данных классов наиболее опасными являются общие атаки, поскольку, в отличие от атак на функции сжатия, они угрожают не одной хеш-функции, а всему классу хеш-функций, которые используют определенную конструкцию, несмотря на большее количество ресурсов, необходимых для их реализации. Таким образом, после появления атаки Жу подавляющее большинство хеш-функций, построенных на основе конструкции Меркля – Дамгарда, перестало считаться устойчивыми. Особенно это коснулось хеш-функций, где использовали идею Пренила [6] относительно «каскадирования» хеш-значений – параллельного вычисления хеш-функций, обеспечивающих исходные значения небольшой разрядности (16, 32, 64 бита – в зависимости от вычислительных платформ), и конкатенации их исходных значений для получения хеш-значения. В то же время, в отличие от атак полного перебора, общие атаки для своей реализации требуют существенно меньшего количества ресурсов, и в определенных случаях это количество ресурсов может быть доступным для злоумышленников, в отличие от количества ресурсов, необходимых для реализации атак полного перебора для хеш-функций, имеющих современную длину хеш-значений (256, 512, 1024, 2048 бит).

Атака «прямым перебором» [1, 7] может быть выполнена для нахождения прообраза по заданному хеш-значению или для нахождения прообраза, дающего заданное хеш-значение. Суть атаки заключается в последовательном или случайном переборе входящих сообщений и сравнении результата вычисления хеш-функции для этих сообщений с заданным. Сложность такой атаки оценивают 2^{n-1} операций вычисления функции сжатия, где n – длина значений в битах.

Жу описал общую атаку з использованием мультиколлизий на хеш-функцию Меркля – Дамгарда, где показал, что на построение 2^l коллизий нужно потратить меньше времени, чем на 2^l реализаций атак «дня рождения» [1, 3, 7]. Такой результат был достигнут благодаря оригинальному подходу к построению коллизий. На рис. 2 изображена схема коллизий на хеш-функцию Меркля – Дамгарда.

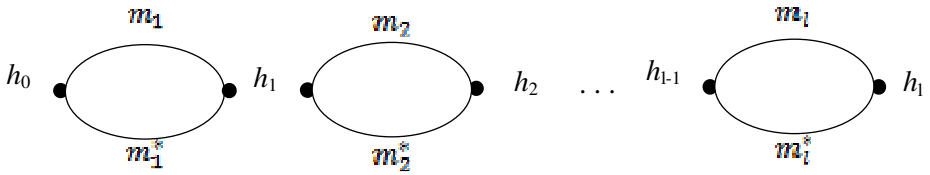

Рис. 2. Граф процесса хеширования сообщений при атаке Жу на конструкцию Меркля – Дамгарда

Из рис. 2 видно, что атака предусматривает l реализаций атак «дня рождения», то есть имеет сложность $O(l \cdot 2^{0.5 \cdot n})$ вычислений функции сжатия f(·) для конструкции Меркля – Дамгарда. Характерной особенностью этого метода является то, что все сообщения, образующие коллизии, имеют одинаковую длину [3].

Конструкции повышенной стойкости

Поскольку общие атаки могут быть применены для различных хеш-функций, использующих одну конструкцию, то есть уязвимость к этим атакам заключена в конструкциях, то методы противодействия необходимо внедрять именно на уровне конструкций.

Для улучшения устойчивости к мультиколлизиям предлагаем такую конструкцию:

$$\begin{cases} h_i = f(h_{i-1}, m_i, m_{i-r_i}) \\ r_i = rand(m_i) \end{cases};$$

где $rand(\cdot)$ – функция, обеспечивающая равномерное распределение выходных значений r_i на

каждой итерации (если $i - r_i < 0$, то выбирают блок $i - r_i + l$).

Пусть для данной конструкции происходит атака Жу. Тогда злоумышленник, в соответствии с парадоксом «дня рождения» за $2^{0.5 \cdot n}$ вичислений, находит коллизию для блока данных m_i:

$$\begin{cases} h_i = f(h_{i-1}, m_i, m_{i-r_i}) = f(h_{i-1}, m_i^*, m_{i-r_i^*}) \\ r_i = rand(m_i) \\ r_i^* = rand(m_i^*) \end{cases};$$

где $r_i \in [1; l - 1]$, $r_i \in N$.

Тогда граф процесса хеширования приобретает следующий вид, как на рис. 3.

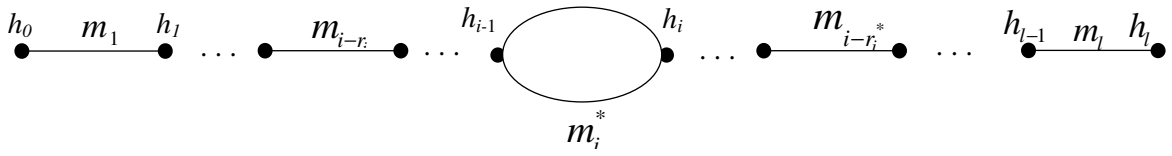

Рис. 3. Граф процесса хеширования при реализации атаки Жу на предложенную конструкцию

Если значения функции $rand(\cdot)$ распределены согласно равномерному закону и $l > 2$, то такая атака будет маловероятной, поскольку необходимо, чтобы выполнялось следующее условие: $\forall j \in N, j \in [1; l]$ и при этом $j - r_i \neq i$.

Для улучшения стойкости такого метода с последовательным вычислением предлагаем определять номер второго блока данных в зависимости от большего количества аргументов функции $rand(\cdot)$:

$$\begin{cases} h_i = f(h_{i-1}, m_i, m_{i-r_i}) \\ r_i = rand(m_i, m_{i+1}) \end{cases};$$

Поскольку данная конструкция не позволяет построение мультиколлизии, то её можно использовать для построения хеш-функций, предусматривающих распараллеливание:

$$\begin{cases} h_i^{(1)} = f^{(1)}(h_{i-1}^{(1)}, m_i, m_{i-r_i}); \\ h_i^{(2)} = f^{(2)}(h_{i-1}^{(2)}, m_i, m_{i-r_i}); \\ \dots \\ h_i^{(q)} = f^{(q)}(h_{i-1}^{(q)}, m_i, m_{i-r_i}); \\ r_i = rand(m_i). \end{cases}$$

При использовании функции сжатия, результат которой вычисляют в зависимости от значения блока данных m_{i-r_i}, где r_i зависит от большего количества блоков (m_i та m_i+1), предлагаем такую конструкцию хеширования:

$$\begin{cases} h_i^{(1)} = f^{(1)}(h_{i-1}^{(1)}, m_i, m_{i-r_i}); \\ h_i^{(2)} = f^{(2)}(h_{i-1}^{(2)}, m_i, m_{i-r_i}); \\ \dots \\ h_i^{(q)} = f^{(q)}(h_{i-1}^{(q)}, m_i, m_{i-r_i}); \\ r_i = rand(m_i, m_{i+1}). \end{cases}$$

Максимальным количеством аргументов функции *rand*(·) может быть l – 1 блок данных, где l – количество блоков данных, на которые разбивают входящее сообщение M. Такая конструкция имеет следующий вид для алгоритма с последовательной обработкой:

$$\begin{cases} h_i = f(h_{i-1}, m_i, m_{i-r_i}); \\ r_i = rand(m_1, m_2, ..., m_{i+1}, m_{i+2}, m_{i+3} ..., m_l). \end{cases}$$

В данном случае не используют m_i блок данных. Для параллельного вычисления конструкция будет выглядеть следующим образом:

$$\begin{cases} h_i^{(1)} = f^{(1)}(h_{i-1}^{(1)}, m_i, m_{i-r_i}); \\ h_i^{(2)} = f^{(2)}(h_{i-1}^{(2)}, m_i, m_{i-r_i}); \\ \ldots \\ h_i^{(q)} = f^{(q)}(h_{i-1}^{(q)}, m_i, m_{i-r_i}); \\ r_i = rand(m_1, m_2, ..., m_{i+1}, m_{i+2}, m_{i+3} ..., m_l). \end{cases}$$

Таким образом, были разработаны конструкции с использованием третьего аргумента в функции сжатия. Использование предложенных конструкций позволит противодействовать мультиколлизиям и ускорить вычисления за счет распараллеливания выполняемых операций при условии, что будет выбрана соответствующая безопасная функция сжатия.

Функции сжатия

Для построения хеш-функций на основе конструкций хеширования, приведенных выше, необходимо реализовать функции сжатия. Сейчас известно большое количество способов их реализации, однако с точки зрения параметров устойчивость/скорость одними из самых распространенных являются такие [1, 7 – 10]:

– на основе возведения в степень по модулю простого числа (дискретное логарифмирование) [1, 7]:

$$f(a,b,c) = g^{a+b+c} \bmod p.$$

– на основе эллиптических кривых:
– выбирают точку на эллиптической кривой x и для нее рассчитывают координату y. Кривую задают формулой, например $y^2 = x^3+ax^2+bx+c$.
– на основе нелинейных логических функций (SHA2, SEAL) [10, 11]:

$$f(a,b,c) = (a \wedge b) \vee (a \wedge c);$$

$$f(a,b,c) = a \oplus b \oplus c;$$

$$f(a,b,c) = (a \wedge b) \vee (a \wedge c) \vee (b \wedge c);$$

$$f(a,b,c) = (a \wedge b) \oplus (a \wedge c) \oplus (b \wedge c);$$

$$f(a,b,c) = (a \wedge b) \oplus (\bar{a} \wedge c);$$

Функция сжатия на основе возведения в степень по модулю простого числа и на основе эллиптических кривых является теоретически устойчивой к атакам [7]. Хотя оценки ее сложности меньше оценки реализации атак полного перебора, они достаточно высоки для обеспечения хеширования, взлом которого с практической точки зрения невозможен.

Последние пять преобразований позволяют быстро объединять входные значения за счет скорости вычисления логических операций на микропроцессорах, однако их стойкость

теоретически не доказана, и нельзя гарантировать, что они не будут сломаны в дальнейшем. Несмотря на это, они были детально исследованы в работах различных ученых (вследствие их использования в международных стандартах хеширования, которые действовали около 20 лет), и в этих работах не показана их уязвимость [10, 11].

Для тестирования было сгенерировано пять псевдослучайных последовательностей длиной 6400 байт. С добавлением размера сообщения и разбиванием по 256 бит получено 1000 блоков данных. В табл. 1 показана частота выбора определенного блока на каждой итерации хеширования для каждой из пяти последовательностей. Использовали генерацию номера блока данных относительно одного блока данных (m_i).

Таблица 1

Частота выбора блока в качестве аргумента функции сжатия

	Эксперимент №1	Эксперимент №2	Эксперимент №3	Эксперимент №4	Эксперимент №5
Блоки не были завязаны	351/1000	40/1000	36/1000	33/1000	40/1000
Блоки были завязаны на 1 итерации	400/1000	359/1000	402/1000	389/1000	310/1000
Блоки были завязаны на 2 итерациях	170/1000	150/1000	170/1000	250/1000	210/1000
Блоки были завязаны на 3 и более итерациях	81/1000	110/1000	82/1000	40/1000	91/1000

Из табл. 1 видно, что около трети блоков не участвуют в завязывании блоков, около 40% используются на определенной итерации 1 раз, и менее трети блоков используются в функции сжатия более одного раза. При изменении метода генерирования псевдослучайных чисел полученные результаты экспериментального исследования изменятся.

Данный эксперимент был проведен для реализации, в которой сгенерированный номер блока данных зависит от значения двух блоков на каждой итерации (m_i и m_{i+1}). В данном случае каждый блок данных стал аргументом функции сжатия как минимум один раз (не считая того, что он является аргументом m_i на i-ой итерации) (табл. 2).

Таблица 2

Частота выбора блока в качестве аргумента функции сжатия

	Эксперимент №1	Эксперимент №2	Эксперимент №3	Эксперимент №4	Эксперимент №5
Блоки не были завязаны	-	-	-	-	-
Блоки были завязаны на 1 итерации	1000/1000	1000/1000	1000/1000	1000/1000	1000/1000
Блоки были завязаны на 2 итерациях	361/1000	432/1000	371/1000	389/1000	380/1000
Блоки были завязаны на 3 и более итерациях	271/1000	240/1000	200/1000	271/1000	262/1000

Сравнивая данные в таблицах, можно сделать вывод, что количество связанных блоков данных увеличилась от 60% до 100%, а связанность блоков в целом (количество использований блока на других итерациях) увеличилась больше, чем в два раза.

Выводы

Из проведенного анализа атак, использующих мультиколлизии, было определено, что основным методом противодействия им является нарушение итеративности процесса хеширования. Для этого был предложен ряд конструкций хеширования сообщения, предусматривающих использование аргумента в функции сжатия, определяющегося по определенному псевдослучайному закону. Анализ этого подхода позволил формально обосновать повышение устойчивости, в частности к атакам Жу и Келси – Коно. По результатам экспериментального исследования было обнаружено, что при использовании определенного типа завязывания блоков данных получаемое повышение устойчивости варьируется в зависимости от выбора генератора псевдослучайных чисел. Поэтому именно этой особенности будут посвящены дальнейшие исследования.

СПИСОК ЛИТЕРАТУРЫ

1. Баришев Ю. В. Методи та засоби швидкого багатоканального гешування даних в комп'ютерних системах : монографія / Ю. В. Баришев, В. А. Лужецький; за заг. ред. В. А. Лужецького. – Вінниця : ВНТУ, 2016. – 144 с.

2. Gauravaram P. Cryptographic Hash Functions: Cryptanalysis, Design and Applications [Електронний ресурс] / Praveen Gauravaram. – 2009. – 298 с. – Режим доступу до ресурсу: http://eprints.qut.edu.au/16372/1/Praveen_Gauravaram_Thesis.pdf. – Назва з екрану.

3. Joux A. Multicollisions in Iterated Hash Functions. Application to Cascaded Constructions / Antoine Joux // Lecture Notes in Computer Science. – 2004. – № 3152. – С. 306 – 316

4. Kelsey J. Second preimages on n-bit hash functions for much less than 2n work / J. Kelsey, B. Schneier // EUROCRYPT. – 2005. – P. 474 – 490.

5. Lucks S. Design Principles for Iterated Hash Functions [Електронний ресурс] / S. Lucks // Cryptology ePrint Archive. – 2004. – 22 с. – Режим доступу до ресурсу: http://eprint.iacr.org/2004/253.pdf. – Назва з екрану.

6. Preneel B. Analysis and Design of Cryptographic Hash Functions: PhD thesis [Електронний ресурс] / Bart Preneel. – Leuven: Katholieke Universiteit Leuven, 1993. – 323 с. – Режим доступу до ресурсу: http://homes.esat.kuleuven.be/~preneel/phd_preneel_feb1993.pdf. – Назва з екрану.

7. Schneier B. Applied Cryptography, Second edition: Protocols, Algorithms, and Source Code in C / Bruce Schneier. – New-York: Wiley Computer Publishing, John Wiley & Sons, Inc, 1996. – 1027 с.

8. Schneier B. The Skein Hash Function Family [Електронний ресурс] / [B. Schneier, N. Ferguson, S. Lucks and others]. – Режим доступу: URL https://www.schneier.com/skein1.3.pdf. – Назва з екрану.

9. Bertoni G. The Keccak sponge function family [Електронний ресурс] / G. Bertoni, J. Daemen, M. Peeters, G. V. Assche. – Режим доступу: URL http://keccak.noekeon.org/specs_summary.html. – Назва з екрану.

10. Secure Hash Signature Standard (SHS): FIPS PUB 180-2. – [Чинний від 2002-08-01]. Processing Standards Publication. – Gaithersburg 2002. – 76 с. (NIST).

11. Implementation of SHA-256 in C [Електронний ресурс] / Режим доступу: URL http://bradconte.com/sha256_c. – Назва з екрану.

Барышев Юрий Владимирович – к. т. н., доцент кафедры защиты информации, e-mail: yuriy.baryshev@gmail.com.

Комаров Андрей Олегович – магистрант кафедры защиты информации, e-mail: komand9@gmail.com.

Винницкий национальный технический университет.