УДК 003.26 : 004.424.47

В. А. Лужецький, д. т. н., проф.; Ю. В. Барышев

КОНСТРУКЦИИ ХЕШИРОВАНИЯ СТОЙКИЕ К МУЛЬТИКОЛЛИЗИЯМ

В данной статье представлен анализ атак, основывающихся на нахождении мультиколлизий и методов противодействия им. Предложена обобщенная конструкция многоканального хеширования. С использованием данной конструкции обобщено и усовершенствовано известные подходы увеличения стойкости хеширования к мультиколлизиям. Предложен новый подход к построению параллельного хеширования стойкого к мультиколлизиям, который формализован в виде конструкций. Приведены оценки времени хеширования с помощью предложенных в статье конструкций.

Ключевые слова: *конструкции хеширования, мультиколлизии, атака Жукса, стойкие конструкции многоканального хеширования.*

Введение

В последние годы отрасль криптографии, связанная с хешированием, столкнулась с существенной проблемой – обеспечением стойкости к мультиколлизям, которые используются атакой Жукса. Зарождение этой проблемы началось с атаки "Дня рождения", ответом на которую стало увеличение в два раза длины результирующего хеш-значения. Такой выход оказался неадекватным имеющимся вычислительным ресурсам общества и ограничениям во времени выполнения хеширования. Для уменьшения длительности хеширования предложено каскадирование, описанное в работе [1], то есть распараллеливание вычислений хеш-значений малой разрядности, которые вычисляются, используя разные хеш-функции, и их конкатенация после завершения последней итерации. Каскадирование считалось эффективным средством противодействия атаке "Дня рождения" до 2004 года, когда Жукс опубликовал работу [2]. Находя, так называемые, мультиколлизии в одной из хеш-функций, вычисляющихся параллельно, Жукс показал, что стойкость хеш-значения, полученного путем каскадирования этих функций, не намного больше стойкости одной из них (при условии одинаковой стойкости функций). Таким образом, проблема, связанная со стойкостью и продолжительностью хеширования, вернулась.

Подходы, предложенные Жуксом в его атаке, были обобщены для любого количества хеш-функций и усовершенствованы для более быстрого нахождения мультиколлизий в работах [3 – 6]. Совершены ряд попыток предложить математические модели хеш-функций, стойких к таким атакам, которые согласно традиции, сложившейся в западной литературе, принято называть конструкциями. Наиболее известными являются конструкция HAIFA [7] и конструкция удвоенного канала Люкса [8]. Первая использует ряд специфических аргументов, но теоретически не доказывает увеличения стойкости данной конструкции к атаке Жукса, а вторая решает задачу путем двукратного увеличения вычислений и длин промежуточных хеш-значений.

Целью данного исследования является разработка конструкций хеширования, которые обеспечивают распараллеливание вычислений хеш-значений и не позволяют построить мультиколлизии.

Для достижения цели решаются следующие задачи:

– анализ известных методов построения мультиколлизий;

– анализ известных конструкций, которые предлагались для увеличения стойкости мультиколлизиям;

– разработка обобщенной конструкции хеширования;

– в обобщенной конструкции хеширования определение свойств, не позволяющих построить мультиколлизии криптоаналитику.

Анализ методов построения мультиколлизий

Впервые атака с использованием мультиколлизии предложена в работе [2] Жуксом. Данная атака была предложена для хеш-функций, использующих каскадирование [1], и основанных на усиленной конструкции Меркля-Дамгаарда. Согласно этой конструкции, информационное сообщение **M** дополняется до длины, кратной длине блока данных, разбивается на l частей и дополняется блоком m_{l+1}, который содержит длину оригинального сообщения **M**, а хеширование происходит согласно формуле:

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

где h_i – промежуточное хеш-значение, полученное на i-ом шаге;

$f(\cdot)$ – функция сжатия, обеспечивающая фиксированную длину результата.

Рассмотрим две функции сжатия $f^{(1)}(\cdot)$ и $f^{(2)}(\cdot)$, которые реализуют хеширование согласно формуле (1). Результирующее хеш-значение формируется путем конкатенации хеш-значений малой разрядности $h_{l+1}^{(1)}$ и $h_{l+1}^{(2)}$, определенных при помощи $f^{(1)}(\cdot)$ и $f^{(2)}(\cdot)$ соответственно. Такой способ хеширования позволяет выполнять параллельное вычисление функций $f^{(1)}(\cdot)$ и $f^{(2)}(\cdot)$, а потому сократить время хеширования. Но, как доказал Жукс [2], это уменьшает стойкость результирующего хеш-значения в сравнении с хеш-значением, определенным при помощи одной функции, имеющей вдвое большую разрядность хеш-значения, чем $f^{(1)}(\cdot)$ и $f^{(2)}(\cdot)$.

Свою атаку Жукс построил на нахождении мультиколлизии для одной функции. Потом среди образованных коллизий находят такие, которые приведут к образованию коллизии и в другой функции. Мультиколлизию Жукс получил, находя для каждого i-го блока данных m_i другой блок данных m_i^* такой, в котором выполняется следующее условие:

$$f^{(1)}(m_i, h_{i-1}) = f^{(1)}(m_i^*, h_{i-1}). \tag{2}$$

На рис. 1 схематически изображено, как находит мультиколлизии Жукс в своей работе [2] для хеш-функций, которые используют усиленную конструкцию Меркля-Дамгаарда или аналогичную ей.

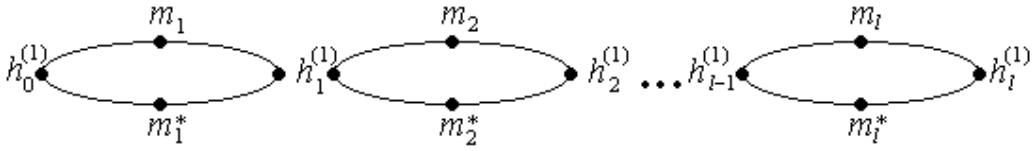

Рис. 1. Вид мультиколлизии Жукса

Используя l-мультиколлизию, которая изображена на рис. 1, можно построить 2^l разных сообщения, которые вызовут коллизию в первой функции $f^{(1)}(\cdot)$, а потом среди них со значительной вероятностью можно найти хотя бы одно такое сообщение, которое приведет к коллизии в другой функции $f^{(2)}(\cdot)$. Для усиленной конструкции Меркля-Дамгаарда последний блок данных m_{l+1} не является объектом поиска коллизии, поскольку он содержит длину оригинального сообщения, а потому его подмена сразу будет замечена. Очевидно, что чем больше количество блоков данных l, тем большая вероятность нахождения коллизии. Таким образом, сложность поиска коллизии для n-розрядных функций $f^{(1)}(\cdot)$ и $f^{(2)}(\cdot)$ будет оцениваться в $O(1 \cdot 2^{n/2})$ операций вместо ожидаемого значения $O(2^{2n/2})$.

В работе [2] Жукс предложил нахождение мультиколлизий только для случая, когда параллельно вычисляются две хеш-функции и блоки данных обрабатываются только во

время одной итерации. Работа [3] направлена на поиск мультиколлизий в случае, когда блоки данных обрабатываются при помощи функции сжатия два раза. В работе [4] Хоч и Шамир развили и обобщили этот результат. Они рассматривали случай, когда хеш-функции являются "расширенными" (expanded), то есть входные блоки данных для функций $f^{(1)}(\cdot)$ и $f^{(2)}(\cdot)$ могут подаваться на вычислительные каналы, по терминологии Люкса [8], в разной последовательности. Например, при помощи функции $f^{(1)}(\cdot)$ вычисляется хеш-значение сначала четных блоков данных, а потом нечетных, а при помощи функции $f^{(2)}(\cdot)$ – наоборот, или используется любой другой способ перестановки блоков данных. Кроме того, в работе [4] к понятию "расширенные" хеш-функции относятся те, которые обрабатывают каждый блок данных несколько раз, например, двигаясь первый раз от m_1 к m_l, а потом в инверсном направлении. Эти идеи были развиты в статье [5], в которой авторы с позиции теории автоматов описывают хеширование и поиск мультиколлизий.

В работе [6] предложен подход к построению мультиколлизий, основывающийся на нахождении "расширяющихся сообщений" (expandable messages). Эти сообщения используются аналогично m_i^* из формулы (2), но в отличие от похода, предложенного Жуксом, "расширяющиеся сообщения" могут состоять из разного количества блоков данных. Так находится несколько сообщений разной длинны, которые вызывают коллизию и имеют одинаковое промежуточное хеш-значение в качестве начального. Кроме того, в статье [6] авторы показывают как можно увеличить эффективность данной атаки, если функция сжатия, лежащая в основе хеширования имеет фиксированные точки, что характерно для конструкции Девиса-Мэйера. Для длинных сообщений такой подход позволил достигнуть лучших результатов, чем в работе [2].

Анализ известных конструкций хеширования

С целью усложнения атак, использующих поиск мультиколлизий, авторами статьи [7] была предложена конструкция HAIFA (HAsh Iterative FrAmework). Эта конструкция предусматривала расширение аргументов итеративной функции за счет увеличения их количества в конструкции (1):

$$h_i = f\left(m_i, h_{i-1}, \#bits_i, r\right), \tag{3}$$

где $\#bits_i$ – количество уже захешированных бит сообщения; r – псевдослучайное число.

Псевдослучайное число r в работе [7] предлагаем использовать в качестве идентификатора сессии обмена данными или как некоторое секретное число при использовании хеш-функции для цифровой подписи. Введение дополнительных аргументов функции сжатия позволило усложнить криптоаналитику задачу предварительно подготовленной атаки, но в то же время, конструкция (3) не обеспечивает нерешаемость этой задачи. Использование счетчика уже захешированных данных заставляет криптоаналитика искать коллизии с учетом относительного места блока данных в сообщении, но никак не противодействует атаке Жукса. Основным преимуществом конструкции (3) является то, что до тех пор, пока злоумышленнику неизвестно псевдослучайное число r и хотя бы часть хешируемого сообщения, он не может начинать атаку.

Наиболее стойкой к мультиколлизиям является конструкция "удвоенного канала" (double pipe) Люкса [8], поэтому она имеет наибольший потенциал для распараллеливания в сравнении с другими конструкциями. Конструкция, предложенная Люксом, возникла из очевидного ответа на атаку Жукса. Если при распараллеливании стойкость n-разрядного хеш-значения $O\left(2^{n/2}\right)$, то необходимо увеличить в два раза разрядность промежуточных хеш-значений [8]. Так появилась конструкция "широкого канала" (wide pipe) [8], но такое увеличение разрядности очевидно будет негативно влиять на скорость и вычислительные

ресурсы. Во избежание этого Люксом и была предложена конструкция "удвоенного канала", которая предусматривает наличие двух вычислителей, каналов при помощи которых определяется промежуточное хеш-значение разрядности n, при этом информационные данные разбиваются на блоки по k бит ($k \geq n$) [8]:

$$\begin{cases} h_i^{(1)} = f\left(h_{i-1}^{(1)}, h_{i-1}^{(2)} \,||\, m_i\right) \\ h_i^{(2)} = f\left(h_{i-1}^{(2)}, h_{i-1}^{(1)} \,||\, m_i\right) \end{cases}, \tag{4}$$

где $h_i^{(1)}$ и $h_i^{(2)}$ – промежуточные хеш-значения, полученные на первом и втором каналах соответственно.

После последней итерации предусматривается раунд хеширования, который сжимает полученные значения до n разрядов [8]:

$$h = f\left(h^{\circ}, h_l^{(1)} \,||\, h_l^{(2)} \,||\, 0^{k-n}\right), \tag{5}$$

где h° – начальное заполнение, аналогичное $h_0^{(1)}$ и $h_0^{(2)}$; 0^{k-n} – дополнение до полного $(k+n)$-разрядного блока.

Основными недостатками конструкции "удвоенного канала" являются удвоенные аппаратные затраты для реализации хеширование в сравнении с другими конструкциями; большое количество векторов инициализации – $3n$, что усложняет разработку ключевого варианта хеширования на основе такой конструкции. Следует отметить, что последний шаг негативно влияет на стойкость всей конструкции в целом и нарушает однородность процесса хеширования. Кроме того, структура Люкса остается уязвимой к атакам, которые используют предварительную подготовку криптоаналитика.

Обобщим известные конструкции хеширования для определения методов противодействия построению мультиколлизий.

Обобщенная конструкция хеширования

Рассмотрим обобщенную конструкцию итеративного хеширования. Определим основные аргументы, которые могут использоваться в хеш-функции. Все известные конструкции итеративного хеширования в качестве аргументов имеют блок хешируемого сообщения и промежуточное хеш-значение, полученное на предыдущей итерации, что представляет собой конструкцию (1). Расширим это понятие, считая, что промежуточное хеш-значение, полученное на i-ом шаге, может зависеть от: всех блоков информационных данных; i предыдущих промежуточных хеш-значений (включая вектор инициализации h_0); дополнительных ключевых данных. Поскольку в большинстве случаев в качестве ключевых данных используют вектор инициализации h_0, а использование дополнительных ключевых данных обуславливается спецификой конкретных задач и не влияет на поиск мультиколлизий, то не будем их отдельно рассматривать в обобщенной конструкции для уменьшения ее громоздкости.

Таким образом, промежуточное хеш-значение на i-ой итерации может вычисляться по функции такого вида:

$$\mathrm{h_i} = \mathrm{f}\left(\mathrm{h_0}, \mathrm{h_1}, ..., \mathrm{h_{i-1}}, \mathrm{m_1}, \mathrm{m_2}, ..., \mathrm{m_1}, \mathrm{m_{l+1}}\right).$$

В общем случае можно выполнять усиление конструкции, учитывая количество уже захешированных информационных бит вместо дописывания битовой длинны сообщения к его концу, как это сделано в конструкции (3):

$$\mathrm{h_i} = \mathrm{f}\left(\mathrm{h_0}, \mathrm{h_1}, ..., \mathrm{h_{i-1}}, \mathrm{m_1}, \mathrm{m_2}, ..., \mathrm{m_l}, \mathrm{length_i}\right), \tag{6}$$

где $length_i$ – битовая длина уже захешированной части сообщения или ее эквивалент.

В хешировании могут использовать и аргументы, изменяющиеся псевдослучайно. Если

рассматривать общий случай, то числа псевдослучайной последовательности могут использоваться несколько раз на разных итерациях. Кроме того, на одной итерации может быть использовано несколько раз на разных итерациях. Более того, на одной итерации может быть использовано несколько псевдослучайных чисел. Учитывая это, формула (6) примет такой вид:

$$\mathrm{h_i} = \mathrm{f}\left(\mathrm{h_0}, \mathrm{h_1}, ..., \mathrm{h_{i-1}}, \mathrm{m_1}, \mathrm{m_2}, ..., \mathrm{m_1}, \mathrm{r_1}, \mathrm{r_{2,}} ..., \mathrm{r_z}, \mathrm{length_i}\right), \quad (7)$$

где z – количество псевдослучайных чисел, использующихся во время одной итерации; $r_1, r_{2,}, ..., r_z$ – псевдослучайные числа.

Псевдослучайные числа могут быть опубликованы открыто, определятся только перед началом обмена данными или использоваться как некоторый секретный параметр для цифровой подписи, как было предложено в работе [7].

С целью распараллеливания вычислений при хешировании необходимо использовать q каналов вычисления. Конструкция (7) может быть обобщена следующим образом:

$$\begin{cases} h_i^{(1)} = f\left(h_0^{(1)}, h_1^{(1)}, ..., h_{i-1}^{(1)}, h_0^{(2)}, h_1^{(2)}, ..., h_{i-1}^{(q)}, m_1, m_2, ..., m_l, r_1^{(1)}, r_2^{(1)}, ..., r_{z_1}^{(1)}, length_i\right) \\ h_i^{(2)} = f\left(h_0^{(1)}, h_1^{(1)}, ..., h_{i-1}^{(1)}, h_0^{(2)}, h_1^{(2)}, ..., h_{i-1}^{(q)}, m_1, m_2, ..., m_l, r_1^{(2)}, r_2^{(2)}, ..., r_{z_2}^{(2)}, length_i\right) \\ ... \\ h_i^{(q)} = f\left(h_0^{(1)}, h_1^{(1)}, ..., h_{i-1}^{(1)}, h_0^{(2)}, h_1^{(2)}, ..., h_{i-1}^{(q)}, m_1, m_2, ..., m_l, r_1^{(q)}, r_2^{(q)}, ..., r_{z_q}^{(q)}, length_i\right) \end{cases}, \quad (8)$$

где z_j – количество псевдослучайных чисел, использующихся в j-ом канале ($\sum_{j=1}^{q} z_j = z$).

Более того, хеширование может происходить с использованием разных функций сжатия на каждой итерации в каждом канале. И поэтому формула (8) изменится таким образом:

$$\begin{cases} h_i^{(1)} = f_i^{(1)}\left(h_0^{(1)}, h_1^{(1)}, ..., h_{i-1}^{(1)}, h_0^{(2)}, h_1^{(2)}, ..., h_{i-1}^{(q)}, m_1, m_2, ..., m_l, r_1^{(1)}, r_2^{(1)}, ..., r_{z_1}^{(1)}, length_i\right) \\ h_i^{(2)} = f_i^{(2)}\left(h_0^{(1)}, h_1^{(1)}, ..., h_{i-1}^{(1)}, h_0^{(2)}, h_1^{(2)}, ..., h_{i-1}^{(q)}, m_1, m_2, ..., m_l, r_1^{(2)}, r_2^{(2)}, ..., r_{z_2}^{(2)}, length_i\right) \\ ... \\ h_i^{(q)} = f_i^{(q)}\left(h_0^{(1)}, h_1^{(1)}, ..., h_{i-1}^{(1)}, h_0^{(2)}, h_1^{(2)}, ..., h_{i-1}^{(q)}, m_1, m_2, ..., m_l, r_1^{(q)}, r_2^{(q)}, ..., r_{z_q}^{(q)}, length_i\right) \end{cases}. \quad (9)$$

Введем такое обозначение для многоканального хеширования (Multi-Pipe Hashing) – $\mathrm{MPHq}\left(\mathrm{k}, \mathrm{d}, \mathrm{r}, \mathrm{z}\right)$, где q – количество каналов, k – количество каналов, от промежуточных хеш-значений которых зависит следующее хеш-значение j-го $\left(\mathrm{j} = 1, 2, ..., \mathrm{q}\right)$ канала, d – количество блоков данных, которые берут участие в формировании хеш-значения в j-ом канале, g – порядок псевдослучайности, характеризующий режим использования псевдослучайных чисел ($g = 0$ – псевдослучайные числа не используются, $g = 1$ – используются как дополнительные данные, $g = 2$ – используются как индексы блоков данных), z – количество псевдослучайных чисел, использующихся во время одной итерации.

Таким образом получено обобщенную конструкцию хеширования. Рассмотрим ее свойства, которые смогут противостоять построению мультиколлизий.

Определение методов усложнения построения мультиколлизий

Наиболее очевидным методом усложнения построения мультиколлизий является многократная обработка блока информационных данных. Однако такой метод может усложнить только классическую атаку Жукса, а не более поздние атаки, производные от нее. К таким принадлежит, например, атака, предложенная в статье [4].

Более интересным методом усложнения построения мультиколлизий является одновременное использование нескольких блоков данных на каждой итерации, например, такое:

$$h_i = f\left(h_{i-1}, m_{i-a}, m_{i-b}\right), \tag{10}$$

где a, b – некоторые константы.

Реализация конструкции (10) возможна несколькими способами: последовательным вычислением двух блоков данных аналогично конструкции (1) – фактически это две последовательные итерации; "объединением" m_{i-a} и m_{i-b} при помощи определенной операции, которая быстро выполняется. В первом случае получаем хеш-функцию, уязвимость которой к мультиколлизиям доказана еще в работе [3]. Второй случай более интересен. Пусть операция, преобразующая два операнда в один, обозначается знаком "$\circ$", тогда конструкцию (10) можно переписать так:

$$\begin{cases} m'_i = m_{i-a} \circ m_{i-b} \\ h_i = f\left(h_{i-1}, m'_i\right) \end{cases}. \tag{11}$$

Полученная конструкция более стойкая к мультиколлизиям, чем (1), но очевидно, что ее можно взломать аналогичными действиями. Так, используя атаку Жукса, можно найти такие $m_i'^* \neq m'_i$, которые будут соответствовать равенству аналогичному (2), потом найти мультиколлизию. Основным отличием между конструкциями (1) и (11) является то, что при использовании конструкции (11), злоумышленник будет вынужден решать систему уравнений для нахождения m^*_{i-a} и m^*_{i-b}. Это не должно вызвать значительные проблемы для криптоаналитика.

Последнее не свидетельствует о том, что подход, использованный в конструкции (10), является неперспективным, но он нуждается в усовершенствовании. Для этого изменим порядок псевдослучайности и будем использовать псевдослучайные числа как индексы хешируемых блоков данных. Действительно, введение псевдослучайности в первое уравнение системы (11) не позволит злоумышленнику в последствии построить систему уравнений, о которой шла речь выше. Предлагаем следующее введение псевдослучайности:

$$\begin{cases} r_i = rand\left(m_i\right) \\ m'_i = m_i \circ m_{i-r_i} \\ h_i = f\left(h_{i-1}, m'_i\right) \end{cases}, \tag{12}$$

где $rand(\cdot)$ – некоторая функция, значения которой имеют равномерный закон распределения.

Например, функция $rand(\cdot)$ может быть реализована в виде генератора псевдослучайных чисел, начальным состоянием которого будет блок данных m_i. Конструкция (12) стойкая к мультиколлизиям, поскольку злоумышленник уже не может последовательно заменять блоки данных для получения мультиколлизии. Действительно, если одна коллизия m_i^* найдена, то уже блок m_{i-r_i} и блок $m_{i-r_i^*}$ $\left(r_i^* = rand\left(m_i^*\right)\right)$ не могут быть заменены для построения мультиколлизии. Соответственно, если продолжать поиск коллизий, то l коллизий не возможно будет преобразовать в l-мультиколлизию. Многоканальный вариант конструкции (12) $\mathrm{MPHq}(1,2,2,1)$ будет иметь вид:

$$\begin{cases} r_i = rand(m_i) \\ h_i^{(1)} = f_i^{(1)}\left(h_{i-1}^{(1)}, m_i \circ m_{i-r_i}\right) \\ h_i^{(2)} = f_i^{(2)}\left(h_{i-1}^{(2)}, m_i \circ m_{i-r_i}\right). \\ \ldots \\ h_i^{(q)} = f_i^{(q)}\left(h_{i-1}^{(q)}, m_i \circ m_{i-r_i}\right) \end{cases} \quad (13)$$

Для усложнения задачи криптоанализа предлагаем использовать перемешивание блоков данных перед хешированием для разных каналов. Пусть $\left\{m_1^{(j)}, m_2^{(j)}, ..., m_1^{(j)}\right\}$ – некоторая перестановка блоков сообщения $\mathbf{M} = \{m_1, m_2, ..., m_1\}$, тогда конструкция $MPHq(1,2,2,1)$, которая описывается формулой (13), может быть усовершенствована до $MPHq(1,2,2,q)$:

$$\begin{cases} r_i^{(1)} = rand\left(m_i^{(1)}\right) \\ r_i^{(2)} = rand\left(m_i^{(2)}\right) \\ \ldots \\ r_i^{(q)} = rand\left(m_i^{(q)}\right) \\ h_i^{(1)} = f_i^{(1)}\left(h_{i-1}^{(1)}, m_i^{(1)} \circ m_{i-r_i^{(1)}}^{(1)}\right) \\ h_i^{(2)} = f_i^{(2)}\left(h_{i-1}^{(2)}, m_i^{(2)} \circ m_{i-r_i^{(2)}}^{(2)}\right) \\ \ldots \\ h_i^{(q)} = f_i^{(q)}\left(h_{i-1}^{(q)}, m_i^{(q)} \circ m_{i-r_i^{(q)}}^{(q)}\right) \end{cases}. \quad (14)$$

Кроме подхода, предложенного для увеличения стойкости, можно использовать подход, предложенный Люксом в статье [8]. Конструкция $MPH2(2,1,0,0)$, описанная формулами (4) и (5), может быть обобщена конструкцией $MPHq(q,1,0,0)$:

$$\begin{cases} h_i^{(1)} = f_i^{(1)}\left(h_{i-1}^{(1)}, h_{i-1}^{(2)}, ..., h_{i-1}^{(q)}, m_i\right) \\ h_i^{(2)} = f_i^{(2)}\left(h_{i-1}^{(1)}, h_{i-1}^{(2)}, ..., h_{i-1}^{(q)}, m_i\right) \\ \ldots \\ h_i^{(q)} = f_i^{(q)}\left(h_{i-1}^{(1)}, h_{i-1}^{(2)}, ..., h_{i-1}^{(q)}, m_i\right) \end{cases}. \quad (15)$$

Вместе с сохранением эффекта связи каналов хеширования после каждой итерации, возможен вариант упрощения конструкции $MPHq(q,1,0,0)$, которая описывается формулой (15), до конструкции $MPHq(2,1,0,0)$ с сохранением аналогичной стойкости при хешировании сообщений, состоящих из количества блоков не меньшим, чем $(q/2+1)$:

$$\begin{cases} h_i^{(1)} = f_i^{(1)}\left(h_{i-1}^{(q)}, h_{i-1}^{(1)}, m_i\right) \\ h_i^{(2)} = f_i^{(2)}\left(h_{i-1}^{(1)}, h_{i-1}^{(2)}, m_i\right) \\ \ldots \\ h_i^{(q)} = f_i^{(q)}\left(h_{i-1}^{(q-1)}, h_{i-1}^{(q)}, m_i\right) \end{cases}. \quad (16)$$

Преимуществом конструкции $MPHq(1,2,2,q)$, которая описывается формулой (14), перед конструкций $MPHq(2,1,0,0)$, которая описывается формулой (16), является то, что вычисления в каждом канале не зависят от вычислений в других каналах. Это дает

возможность выполнить независимое вычисление хеш-значения. То есть время вычисления хеш-значений при помощи конструкции $\mathrm{MPHq}(2,1,0,0)$ составляет:

$$t_{MPHq(2,1,0,0)} \approx \sum_{i=1}^{l} \max\left(t_i^{(1)}, t_i^{(2)}, ..., t_i^{(q)}\right), \tag{17}$$

где $t_i^{(j)}$ – время выполнения i-ой итерации в j-ом канале.

Оценка продолжительности вычисления хеш-значения при помощи конструкции $\mathrm{MPHq}(1,2,2,\mathrm{q})$ составляет:

$$\mathrm{t}_{\mathrm{MPHq}(1,2,2,\mathrm{q})} \approx \max\left(\sum_{\mathrm{i}=1}^{\mathrm{l}}\left(\mathrm{t}_{\mathrm{i}}^{(1)}\right), \sum_{\mathrm{i}=1}^{\mathrm{l}}\left(\mathrm{t}_{\mathrm{i}}^{(2)}\right), ... \sum_{\mathrm{i}=1}^{\mathrm{l}}\left(\mathrm{t}_{\mathrm{i}}^{(\mathrm{q})}\right)\right). \tag{18}$$

Очевидно, что всегда $t_{MPH(q,2,1,0)} \geq t_{MPH(q,1,2,2)}$.

Выводы

Появление атаки "Дня рождения" заставило увеличить в два раза разрядность хеш-значений, что существенно увеличило время их вычисления. Это, в свою очередь, сделало невозможным использование хеширования для ряда задач. Поэтому распараллеливание хеширования необходимо для того, чтобы время вычисления опять соответствовало требованиям его продолжительности, но на пути распараллеливания стали мультиколлизии, которые используется в атаке Жукса в ее последующих обобщениях и усовершенствованиях, объектом которых является хеш-значения, полученные при помощи распараллеленных хеш-функций.

Проанализировав известные методы распараллеливания, можно обобщить, что в известных конструкциях стойкость к мультколлизиям достигается двукратным увеличением ресурсоемкости вычислений без уменьшения времени вычислений. Известный подход Люкса был усовершенствован и обобщен путем уменьшения его ресурсоемкости и возможности вычисления при помощи q параллельных каналов хеширования, вместо двух, рассмотренных в статье [8]. Это позволяет уменьшить время вычислений. Эти подходы формализовано конструкциями $\mathrm{MPHq}(\mathrm{q},1,0,0)$ и $\mathrm{MPHq}(2,1,0,0)$.

Предложен новый подход к построению хеширования стойкого в мультиколлизиям, который был формализован конструкцией $\mathrm{MPHq}(1,2,2,\mathrm{q})$. Полученные оценки времени вычисления хеш-значений свидетельствуют о том, что предложенный подход, представленный конструкцией $\mathrm{MPHq}(1,2,2,\mathrm{q})$, будет быстрее, чем усовершенствованный, представленный конструкцией $\mathrm{MPHq}(2,1,0,0)$, при условии одинакового качества их реализации.

СПИСОК ЛИТЕРАТУРЫ

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

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

3. Nandi M. Multicollision Attacks on Some Generalized Sequential Hash Functions / M. Nandi, D.R. Stinton // Scientific Literature Digital Library. – Режим доступу до ресурсу : http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.104.5370&rep=rep1&type=pdf

4. Hoch J.J. Breaking the ICE – Finding Multicollisions in Iterative Concatenated and Expanded (ICE) Hash Functions / J.J. Hoch, A. Shamir. – 2006. – Режим доступу до ресурсу : http://www.wisdom.weizmann.ac.il/~yaakovh/papers/hashpaper_submission.pdf

5. Halunen K. An Automata-Theoretic Interpretation of Iterated Hash Functions – Application to Multicollisions / Kimmo Halunen, Juha Kortelainen, Tuomas Kortelainen // Cryptology ePrint Archive. – 2009. – 13 с. – Режим доступу до ресурсу : http://eprint.iacr.org/2009/456.pdf

6. Kelsey J. Second preimages on *n*-bit Hash Function for Less than 2^n Work. / J. Kelsey, B. Schneier // Cryptology ePrint Archive. – 2004. – 15 с. – Режим доступу до ресурсу : http://eprint.iacr.org/2004/304.pdf

7. Biham E. A Framework for Iterative Hash Functions: HAIFA. / Eli Biham, Orr Dunkelman // Second cryptographic hash workshop. – 2006. – 9 с. – Режим доступу до ресурсу : http://csrc.nist.gov/pki/HashWorkshop/2006/Papers/DUNKELMAN_NIST3.pdf

Lucks S. Design Principles for Iterated Hash Functions / S. Lucks // Cryptology ePrint Archive. – 2004. – 22 с. Режим доступу до ресурсу : http://eprint.iacr.org/2004/253.pdf

Лужецький Владимир Андреевич – д. т. н., профессор, заведующий кафедрой защиты информации.

Барышев Юрий Владимирович – магистр по информационной безопасности, аспирант кафедры защиты информации. E-mail: yuriy.baryshev@gmail.com.

Винницкий национальный технический университет.