УДК 004.056: 004.424.47

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

МЕТОДЫ И СРЕДСТВА ПАРАЛЛЕЛЬНОГО УПРАВЛЯЕМОГО ХЕШИРОВАНИЯ

В данной статье представлен анализ подходов к разработке управляемых криптографических примитивов. Разработаны конструкции управляемого многоканального хеширования. Определены криптографические примитивы для реализации функций сжатия в хешировании. Предложены подходы к синтезу этих функций и методы хеширования, которые используют данные подходы. Приведены оценки скорости программной реализации этих методов.

Ключевые слова: *хеширование, криптографические примитивы, управляемость, многоканальность, функции сжатия.*

Вступление

Появление новых атак на функции хеширования [1] вызвало необходимость в пересмотре подходов не только к проектированию функций сжатия, но и к переосмыслению сути хеширования. Это обусловило актуальность разработки новых концепций хеширования. Одним из направлений совершенствования хеширования является концепция управляемого хеширования [2], которая предусматривает изменение параметров преобразований в функциях сжатия от итерации к итерации. Поскольку известные методы криптоанализа как объект исследования используют функции сжатия, параметры которых не меняются от итерации к итерации, то от реализации управляемого хеширования ожидается повышение стойкости.

Для криптографических методов и методов хеширования, в частности, актуальным является обеспечение высокой скорости обработки данных, что достигается путем распараллеливания вычислений [3]. Концепция управляемого хеширования позволяет распараллелить процесс хеширования, а значит уменьшить его продолжительность [4]. Именно поэтому исследования в направлении реализации концепции управляемого хеширования актуальны.

Целью данного исследования является повышение скорости хеширования за счет реализации концепции управляемого хеширования.

Для достижения цели необходимо решить следующие задачи:

– анализ известных методов управляемого хеширования;

– разработка конструкций управляемого хеширования;

– обоснование выбора криптографических примитивов и синтез функций сжатия;

– разработка методов хеширования и их программная реализация.

Анализ известных методов управляемого хеширования

В функциях хеширования Dynamic SHA и Dynamic SHA-2, предложенных для участия в конкурсе на новый стандарт хеширования SHA-3 [5, 6], реализован один из подходов к построению управляемого хеширования, который основывается на использовании управляемых сдвигов. Для создания множества функций сжатия в работе [5] предлагается использовать логические функции аналогичные, использующимся в стандарте хеширования SHA-2 [7]. Как управляемый параметр преобразования используется количество битов, на которое происходит сдвиг. Причем этот параметр предлагалось определять в зависимости от нескольких модифицированных входных блоков данных. Именно это стало причиной недостаточной стойкости Dynamic SHA и Dynamic SHA-2. Авторы работы [6] предложили атаку на эти хэш-функции, которая использует способность криптоаналитика влиять на

управление, навязывая определенные блоки данных.

Известен подход к построению управляемых функций хеширования на основе подстановочно-перестановочных сетей (ППС), предложенный в работе [8] и развитый в работах [9, 10]. В частности в работе [8] приводятся классы криптографических примитивов, примеры шифров и хэш-функций. Пример примитивов класса $F_{2/1}$ (два входных / выходных бита данных, один управляющий) в алгебраической нормальной форме приведен в таблице 1 [8].

Таблица 1

Пример управляемых криптографических примитивов класса $F_{2/1}$

№	Примитив	№	Примитив	№	Примитив
1	$x_1v \oplus x_2v \oplus x_1$	9	$x_1v \oplus x_2 \oplus v$	17	$x_1v \oplus x_2v \oplus x_1 \oplus v \oplus 1$
2	$x_2v \oplus x_1$	10	$x_1v \oplus x_2 \oplus x_1 \oplus v$	18	$x_2v \oplus x_1 \oplus v \oplus 1$
3	$x_1v \oplus x_2v \oplus x_2$	11	$x_1v \oplus x_2v \oplus x_2 \oplus v$	19	$x_1v \oplus x_2 \oplus x_1 \oplus 1$
4	$x_2v \oplus x_1 \oplus x_2$	12	$x_2v \oplus x_2 \oplus x_1 \oplus v$	20	$x_1v \oplus x_2 \oplus 1$
5	$x_1v \oplus x_2$	13	$x_2v \oplus x_2 \oplus x_1 \oplus v \oplus 1$	21	$x_2v \oplus x_1 \oplus x_2 \oplus 1$
6	$x_1v \oplus x_2 \oplus x_1$	14	$x_1v \oplus x_2v \oplus x_2 \oplus v \oplus 1$	22	$x_1v \oplus x_2v \oplus x_2 \oplus 1$
7	$x_2v \oplus x_1 \oplus v$	15	$x_1v \oplus x_2 \oplus x_1 \oplus v \oplus 1$	23	$x_2v \oplus x_1 \oplus 1$
8	$x_1v \oplus x_2v \oplus x_1 \oplus v$	16	$x_1v \oplus x_2 \oplus v \oplus 1$	24	$x_1v \oplus x_2v \oplus x_1 \oplus 1$

Попарно комбинируя такие примитивы, авторы работы [8] определяют лучшие комбинации, используя критерии сбалансированности исходных значений и нелинейности преобразований, получая ППС класса $F_{2/1}$. Аналогично в работе [8] получают ППС классов $F_{2/2}$ и $F_{3/1}$. В работе [9] приведены управляемые функции, имеющие два, три и четыре управляющих бита для двух входных / выходных битов данных, а в работе [10] – обратные преобразования для таких примитивов.

Недостатком подхода ППС является то, что они первоначально были ориентированы на разработку шифров, а потому криптографические примитивы не учитывают особенности хеширования. Например, для хеширования не обязательно наличие криптографического примитива, который осуществляет обратное преобразование, в отличие от шифрования, где это требование является обязательным. Кроме того, функции, предложенные в работах [8 – 10], в первую очередь ориентированы на аппаратную реализацию, где перестановка двух соседних битов блока данных выполняется путем разветвления проводников. В то же время программная реализация этих примитивов осложняется тем, что универсальные микропроцессоры не приспособлены к такого рода преобразованиям. Именно поэтому их необходимо реализовывать, используя резервное копирование переменной, ее сдвиг, наложение маски, что уменьшает скорость вычисления логических функций.

Таким образом, известные функций сжатия для методов управляемого хеширования имеют недостатки, связанные с недостаточными стойкостью или скоростью программной реализации. Именно поэтому необходимо разработать новые функции сжатия, для чего сначала необходимо разработать модели итераций хеширования, которые будут учитывать распараллеливание вычислений.

Разработка конструкций управляемого хеширования

Поскольку процесс хеширования должен обеспечивать хеширование данных произвольной длины, он должен быть итеративным [8]. Итеративность обусловливает обработку данных блоками и их сцепление с промежуточным хеш-значением, полученным

на предыдущей итерации. Для распараллеливания предлагается использовать несколько каналов хеширования (вычислителей, реализующих функцию сжатия). Для обеспечения устойчивого завязывания каналов друг с другом необходимо, чтобы вычисления в каждом из каналов на каждой итерации учитывали промежуточные хеш-значения, полученные в других каналах [4]. Однако управляемое хеширование позволяет завязывать каналы друг с другом как путем использования канальных промежуточных хеш-значений в качестве аргументов, так и при помощи вектора управления:

$$\begin{cases} h_i^{(1)} = f_{v_i^{(1)}}\left(h_{i-1}^{(1)}, m_i\right) \\ h_i^{(2)} = f_{v_i^{(2)}}\left(h_{i-1}^{(2)}, m_i\right) \\ \ldots \\ h_i^{(q)} = f_{v_i^{(q)}}\left(h_{i-1}^{(q)}, m_i\right) \\ v_i^{(1)} = g\left(h_{i-1}^{(2)}, h_{i-1}^{(3)}, \ldots, h_{i-1}^{(q)}\right) \\ v_i^{(2)} = g\left(h_{i-1}^{(1)}, h_{i-1}^{(3)}, \ldots, h_{i-1}^{(q)}\right) \\ \ldots \\ v_i^{(q)} = g\left(h_{i-1}^{(1)}, h_{i-1}^{(2)}, \ldots, h_{i-1}^{(q-1)}\right) \end{cases}, \quad (1)$$

где $h_i^{(j)}$ – промежуточное хеш-значение, полученное в j-ом канале ($j = \overline{1, q}$) на i-ой итерации ($i = \overline{1, l}$); m_i – i-ый блок данных; $f_{v_i^{(j)}}(\cdot)$ – функция сжатия, обеспечивающая постоянную длину выходного значения; $v_i^{(j)}$ – вектор управления, который определяет параметры преобразования функции сжатия $f_{v_i^{(j)}}(\cdot)$ в j-ом канале на i-ой итерации ($i = \overline{1, l}$); $g(\cdot)$ – функция формирования вектора управления.

Обобщим конструкцию (1) для k промежуточных хеш-значений, использующихся как аргумент функции $f_{v_i^{(j)}}(\cdot)$, и ϕ промежуточных хеш-значений – как аргумент функции $g(\cdot)$:

$$\begin{cases} h_i^{(1)} = f_{v_i^{(1)}}\left(h_{i-1}^{(1)}, h_{i-1}^{(2)}, \ldots, h_{i-1}^{(k)}, m_i\right) \\ h_i^{(2)} = f_{v_i^{(2)}}\left(h_{i-1}^{(2)}, h_{i-1}^{(3)}, \ldots, h_{i-1}^{(k+1)}, m_i\right) \\ \ldots \\ h_i^{(q)} = f_{v_i^{(q)}}\left(h_{i-1}^{(q)}, h_{i-1}^{(1)}, \ldots, h_{i-1}^{(k-1)}, m_i\right) \\ v_i^{(1)} = g\left(h_{i-1}^{(q)}, h_{i-1}^{(q-1)}, \ldots, h_{i-1}^{(q-\phi+1)}\right) \\ v_i^{(2)} = g\left(h_{i-1}^{(1)}, h_{i-1}^{(q)}, h_{i-1}^{(q-1)}, \ldots, h_{i-1}^{(q-\phi+2)}\right) \\ \ldots \\ v_i^{(q)} = g\left(h_{i-1}^{(q-1)}, h_{i-1}^{(q-2)}, \ldots, h_{i-1}^{(q-\phi)}\right) \end{cases}, \quad (2)$$

где $k < (q - \phi)$.

Для реализации конструкции многоканального управляемого хеширования (2) определим криптографические примитивы, с помощью которых синтезируем множество функций сжатия.

Множество функций сжатия для управляемого хеширования

Для того, чтобы методы управляемого хеширования позволяли реализовать быстрое выполнение этого процесса, необходимо, чтобы операции, использующиеся как

криптографические примитивы, были "удобными" для универсальных микропроцессоров. К таким операциям принадлежат: сложение по модулю 2^n (+); исключающее или ($\oplus$); инвертирование (~); логическое произведение ($\wedge$); логическое сложение ($\vee$); циклический сдвиг на u бит вправо ($>>> u$); простой сдвиг на u бит вправо/влево ($>> u$/ $<< u$). В то же время криптографические операции должны обеспечивать одинаковое влияние каждого бита входных данных на результат преобразования, а операция простого сдвига ($>> u$/ $<< u$) не позволяет выполнить это условие. Именно поэтому операции $>> u$ и $<< u$ не могут быть использованы для построения функций сжатия в управляемом хешировании. Параметром управляемых преобразований предлагается выбрать количество бит u, на которое циклично сдвигается переменная вправо, поскольку его изменение является простым и быстро выполняется при программной реализации. Кроме того, каждый управляемый параметр предлагается использовать как часть вектора управления.

Как основу для разработки функций сжатия предлагается использовать логические функции, использованные в стандарте SHA-2 [7], и модифицировать их для учета именно управляемого хеширования. Так каждый канал будет реализовывать преобразование ($k = 2$):

$$\begin{aligned} h_i^{(j)} = &\left(m_i >>> u_i^{(j)(m1)} \wedge h_{i-1}^{(j)} >>> u_i^{(j)(h1)}\right)\oplus \\ \oplus &\left(\sim m_i >>> u_i^{(j)(m1)} \wedge h_{i-1}^{(j+1)} >>> u_i^{(j)(h2)}\right) \end{aligned}, \tag{3}$$

где $u_i^{(j)(x\kappa)}$ – количество бит, на которое сдвигается переменная x в к -ой позиции в функции сжатия на i-ой итерации в j-ом канале хеширования.

В функции сжатия (3) используется три управляемых параметра, соответственно вектор управления является конкатенацией этих параметров $v_i^{(j)} = u_i^{(j)(m1)} \| u_i^{(j)(h1)} \| u_i^{(j)(h2)}$.

Для обеспечения большего влияния аргументов функции сжатия на выходное значение предлагается модифицировать еще одну функцию, использующуюся в стандарте [7]:

$$\begin{aligned} h_i^{(j)} = &\left(m_i >>> u_i^{(j)(m1)} \wedge h_{i-1}^{(j)} >>> u_i^{(j)(h1)}\right)\oplus \\ \oplus &\left(m_i >>> u_i^{(j)(m1)} \wedge h_{i-1}^{(j+1)} >>> u_i^{(j)(h2)}\right)\oplus \\ \oplus &\left(h_{i-1}^{(j)} >>> u_i^{(j)(h1)} \wedge h_{i-1}^{(j+1)} >>> u_i^{(j)(h2)}\right) \end{aligned}. \tag{4}$$

Для функции (4) вектор управления не изменится по сравнению с функцией (3). При увеличении количества аргументов в функциях сжатия кроме операции $\wedge$ предлагается использовать $\vee$. Например, для случая $k = 4$ функция сжатия (3) приобретет вид:

$$\begin{aligned} h_i^{(j)} = &\left(m_i >>> u_i^{(j)(m1)} \wedge h_{i-1}^{(j)} >>> u_i^{(j)(h1)}\right)\oplus \\ \oplus &\left(\sim m_i >>> u_i^{(j)(m1)} \wedge h_i^{(j+1)} >>> u_i^{(j)(h2)}\right)\oplus \\ \oplus &\left(m_i >>> u_i^{(j)(m2)} \vee h_{i-1}^{(j+2)} >>> u_i^{(j)(h3)}\right)\oplus \\ \oplus &\left(\sim m_i >>> u_i^{(j)(m2)} \vee h_i^{(j+3)} >>> u_i^{(j)(h4)}\right) \end{aligned}. \tag{5}$$

Функция сжатия (4) при значении параметра $k = 4$ изменится так:

$$\begin{aligned} h_i^{(j)} = &\left(m_i >>> u_i^{(j)(m1)} \wedge h_{i-1}^{(j)} >>> u_i^{(j)(h1)}\right)\oplus \\ \oplus &\left(m_i >>> u_i^{(j)(m1)} \wedge h_i^{(j+1)} >>> u_i^{(j)(h2)}\right)\oplus \\ \oplus &\left(h_{i-1}^{(j)} >>> u_i^{(j)(h1)} \wedge h_i^{(j+1)} >>> u_i^{(j)(h2)}\right)\oplus \\ \oplus &\left(m_i >>> u_i^{(j)(m2)} \vee h_{i-1}^{(j+2)} >>> u_i^{(j)(h3)}\right)\oplus \\ \oplus &\left(m_i >>> u_i^{(j)(m2)} \vee h_i^{(j+3)} >>> u_i^{(j)(h4)}\right)\oplus \\ \oplus &\left(h_{i-1}^{(j+2)} >>> u_i^{(j)(h3)} \vee h_i^{(j+3)} >>> u_i^{(j)(h4)}\right) \end{aligned}. \tag{6}$$

При увеличении параметра k предлагается использовать подход, аналогичный используемому при построении функций сжатия (5) и (6) из функций (3) и (4) соответственно.

Функции сжатия (3 – 6) были использованы для программной реализации управляемого хеширования на языке С и скомпилированы при помощи среды программирования Visual Studio 2005. Полученные программы предусматривали использование одинаковых процедур формирования вектора управления и инициализации. Тестирование программ происходило на компьютере с процессором Intel Pentium 4 с частотой 3 ГГц объемом оперативной памяти 2 Гб. Для оценивания скорости хеширования использована стандартная библиотека time.h. Оценки продолжительности хеширования данных этими методами управляемого хеширования для 16 каналов, оперирующих данными длиной 16 бит (выходное хеш-значение – 256 бит), приведено в таблице 2.

Таблица 2

Оценки продолжительности хеширования

Функция сжатия	Продолжительность хеширования, cycles			
	10 Кб	100 Кб	1 Мб	10 Мб
(3)	78	781	8219	82328
(4)	141	1516	15297	151813
(5)	94	812	8547	84718
(6)	171	1719	17406	171719

Разработанные программы предусматривают обработку данных любой длины и формирование для них выходного хеш-значения, длина которого кратна разрядности канала.

Вывод

Потребность в достижении большей скорости хеширования и одновременно необходимость обеспечения стойкости этого процесса к атакам порождают необходимость поиска новых концепций хеширования, к которым относится концепция управляемого хеширования. Построение методов управляемого хеширования с помощью криптографических примитивов, выполнение которых естественно для универсальных микропроцессоров, позволило достичь высоких показателей скорости, которые увеличиваются линейно при увеличении объема хешируемых данных, в то время как известные методы достигают наилучших показателей скорости при хешировании только больших массивов входных данных. Именно поэтому наиболее уместным использованием предложенных в данной статье методов является хеширования малых объемов данных, в частности, в протоколах аутентификации.

СПИСОК ЛИТЕРАТУРЫ

1. Second Preimage Attacks on Dithered Hash Functions [Електронний ресурс] / E. Andreeva, C. Bouillaguet, P.-A. Fouque and others. – 19 с. – Режим доступу: http://homes.esat.kuleuven.be/~eandreev/eurocrypt08.pdf.

2. Баришев Ю. В. Підхід до хешування, що стійке до аналізу зловмисника / Ю. В. Баришев // Системи обробки інформації. – № 3. – 2010. – С. 99 – 100.

3. Корченко А. Г. Построение систем защиты информации на нечетких множествах. Теория и практические решения / А. Г. Корченко. – К.: "МК-Пресс", 2006. – 320 с.

4. Баришев Ю. В. Методи побудови швидкого хешування / Ю. В. Баришев // Методи та засоби кодування, захисту й ущільнення інформації. Тези другої Міжнародної науково-практичної конференції, м. Вінниця, 22 – 24 квітня 2009 року. – Вінниця: ВНТУ, 2009. – С. 138 – 139.

5. Zijie Xu. Dynamic SHA [Електронний ресурс] / Zijie Xu // Cryptology ePrint Archive. – 2007. – 34 с. – Режим доступу: http://eprint.iacr.org/2007/476.pdf.

6. Aumasson J.-P. Cryptanalysis of Dynamic SHA(2) [Електронний ресурс] / J.-P. Aumasson, O. Dunkelman, S. Indesteege and B. Preneel // COSIC publications. – 2009. – 18 с. – Режим доступу: https://www.cosic.esat.kuleuven.be/publications/article-1277.pdf.

7. Secure Hash Standard: Federal Information Processing Publication Standard Publication 180-3 [Електронний

ресурс] – Gaithersburg, 2008. – 27 с. – Режим доступу: http://csrc.nist.gov/publications/fips/fips180-3/fips180-3_final.pdf.

8. Молдовян Н. А. Криптография: от примитивов к синтезу алгоритмов. / Н. А. Молдовян, А. А. Молдовян, М. А. Еремеев. – СПб.: БХВ-Петербург, 2004. – 448 с.

9. Рудницький В. М. Модель уніфікованого пристрою криптографічного перетворення інформації / В. М. Рудницький, В. Г. Бабенко // Системи обробки інформації. – № 3. – 2009. – С. 91 – 95.

10. Рудницький В. М. Синтез математичних моделей пристроїв декодування інформації для криптографічних систем / В. М. Рудницький, В. Г. Бабенко // Системи обробки інформації. – № 2. – 2009. – С. 124 – 128.

Лужецкий 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.