УДК 004.021+681.3.05

А.В. Антонов

Харьковский университет Воздушных Сил им. И. Кожедуба, Харьков

АНАЛИЗ УЯЗВИМОСТЕЙ РЕАЛИЗАЦИИ В ЦИФРОВЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМАХ ХЕШ-ФУНКЦИИ НА ОСНОВЕ ХАОТИЧЕСКИХ ОТОБРАЖЕНИЙ С ПЕРЕМЕННЫМИ ПАРАМЕТРАМИ И ПАРАЛЛЕЛЬНОЙ ОРГАНИЗАЦИИ ВЫЧИСЛЕНИЙ

Исследуется хеш-функция, построенная на основе хаотических отображений с переменными параметрами и параллельной организации вычислений. Показана ее уязвимость к атакам, использующих особенности реализации хаотических систем на конечном множестве состояний в цифровых вычислительных системах. Доказано, что рассматриваемая функция не является устойчивой к возникновению коллизий первого и второго рода.

Ключевые слова: *хеш-функция, хаотическое отображение, распараллеливание, коллизия.*

Введение

В последнее время в качестве одного из альтернативных подходов к конструированию сжимающих функций хеширующих алгоритмов все чаще предлагается использовать достижения теории динамического хаоса. Из работ, выполненных в данном направлении, следует выделить метод построения хеш-функции на основе хаотических отображений с переменными параметрами и параллельной организации вычислений, предложенный в публикации [1]. Большой практический интерес к этому методу обусловлен его способностью в значительной мере компенсировать низкую эффективность вычислений результатов хаотических преобразований. Однако детальное исследование, представленное в работе [2], показало структурные уязвимости данного метода и его уязвимость к поиску и обнаружению близких коллизий, а так же коллизий первого и второго рода. В данной работе будет продолжено изучение этого метода, в частности рассмотрены уязвимости, вызванные особенностями реализации хаотических динамических систем на конечном множестве состояний. Вкратце дадим описание метода построения хеш-функции, предложенного в работе [1].

Основной раздел

Метод построения хеш-функции на основе хаотических отображений с переменными параметрами и параллельной организации вычислений

Рассматриваемый метод основан на совокупности следующих приемов и правил при построении хеш-функции:

– использование итеративной параллельно-последовательной схемы;

– использование нескольких хаотических отображений в качестве базовых преобразований;

– использование переменных параметров отображений на различных этапах обработки сообщения и его отдельных блоков.

Хеш-функции, построенные в рамках указанного метода, имеют три основных этапа: инициализации, параллельных вычислений, финализации. Рассмотрим вариант хеш-функции на основе данного метода, предложенный его авторами.

Базовые преобразования

В качестве базовых преобразований в анализируемой хеш-функции используются представители класса кусочно-линейных хаотических отображений: палаточное и кусочно-линейное отображение второго порядка, которое в работе [1] называют просто кусочно-линейным. Рекуррентное уравнение для палаточного отображения задается в виде:

$$x_{i+1} = \begin{cases} \dfrac{x_i}{a}, & 0 \le x_i < a \\ \dfrac{1-x_i}{1-a}, & a \le x_i \le 1 \end{cases}, \qquad (1)$$

где $x_i \in (0,1)$ – точки траектории; $a \in (0,1)$ – управляющий параметр отображения (часто называемый ключевым), а рекуррентное уравнение для кусочно-линейного отображения второго порядка задается следующим выражением:

$$x_{i+1} = \begin{cases} x_i/\beta & 0 \le x_i < \beta \\ (x_i-\beta)/(0.5-\beta) & \beta \le x_i < 0.5 \\ (1-x_i-\beta)/(0.5-\beta) & 0.5 \le x_i < 1-\beta \\ (1-x_i)/\beta & 1-\beta \le x_i < 1 \end{cases}, \qquad (2)$$

где $x_i \in (0,1)$ – точки траектории; $\beta \in (0, 0.5)$ – управляющий параметр отображения (ключевой).

Этап инициализации

На этапе инициализации исходное сообщение представляется набором векторов (блоков данных) по 127 элементов размером 1 байт (т.е. по 1016 бит) в каждом векторе. Последний блок сообщения дополняется до длины, кратной 1016 бит: сначала добавляется 64 бита, в которые записывается длина исходного сообщения, а потом необходимое количество бит в виде последовательности $(1010\ldots10)_2$. Дополнение делается даже если длина исходного сообщения кратна 1016 битам. В результате получаем набор векторов $\vec{m}'_i$, где $i = 1,2\ldots n'$, а n' – число векторов (блоков) на которые были разбито исходное сообщение.

Далее формируется инициализирующий вектор $\vec{x}$ размерностью $n'+128$, элементы которого соответствуют точкам траектории отображения (1) при $x_0 = 0,7654$ и $a = 0,6$. С использованием вектора $\vec{x}$ и набора векторов $\vec{m}'_i$ формируются вектор-строка $\vec{c}$ и вектор-столбец $\vec{r}$, элементы которых задаются значениями:

$$c_j = \bigoplus_{i=1}^{n'} \left(m_{i,j} \oplus x_i \right) + x_{n'+j}; \qquad (3)$$

$$r_i = \begin{cases} \sum_{j=1}^{127} \left(\left(m_{i,j} + x_{n'+j} \right) \oplus x_i \right), i = 1,2,\ldots,n' \\ \sum_{j=1}^{127} \left(\left(c_j + x_{n'+j} \right) \oplus x_i \right), i = n'+1 = n \end{cases}, \qquad (4)$$

где $m_{i,j}$ – элементы векторов $\vec{m}'_i$, $j = 1,2,\ldots,127$, $i = 1,2,\ldots,n$, $\oplus$ – битовая операция «исключающее или», «+» и «Σ» – операции сложения по модулю 2^8. На основе векторов $\vec{m}'_i$, векторов $\vec{c}$ и $\vec{r}$ формируется матрица M размерностью $n \times 128$:

$$M = \begin{bmatrix} m_{1,1} & m_{1,2} & \ldots & m_{1,127} & r_2 \\ m_{2,1} & m_{2,2} & \ldots & m_{2,127} & r_3 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ m_{n-1,1} & m_{n-1,2} & \ldots & m_{n-1,127} & r_n \\ c_{127} & c_{126} & \ldots & c_1 & r_1 \end{bmatrix} = \begin{bmatrix} \vec{m}_1 \\ \vec{m}_2 \\ \ldots \\ \vec{m}_{n-1} \\ \vec{m}_n \end{bmatrix}.$$

Этап параллельных вычислений

Далее вычисления хеш-значения функции строятся по итеративной параллельно-последовательной схеме:

$$M = \begin{bmatrix} \vec{m}_1 \\ \vec{m}_2 \\ \ldots \\ \vec{m}_n \end{bmatrix} \Rightarrow \begin{matrix} h_1 = f_1(\vec{m}_1) \\ h_2 = f_2(\vec{m}_2) \\ \ldots \\ h_n = f_n(\vec{m}_n) \end{matrix} \Rightarrow \bigoplus_{i=1}^{n} h_i \Rightarrow H(M),$$

где $f_i(\ )$ – параллельная функция сжатия, которая последовательными итерациями вычисляет по элементам векторов $\vec{m}_i$ значения их промежуточных хеш-значений h_i. Размер хеш-значений h_i в битах соответствует размерности вектора $\vec{m}_i$, т.е. составляет $L = 128$ бит.

Функция сжатия $f(\ )$ строится по итеративной последовательной схеме. Ее ядром является преобразование вида:

$$h_{i,j} = f_{plm}\left(f_{atm}\left(h_{i,j-1}, m_{i,j} \right), m_{i,j} \right),$$

которое для каждого параллельного потока $i = 1,2,\ldots,n$ итеративно осуществляет преобразование элементов $m_{i,j}$ ($j = 1,2,\ldots,L$, $L = 128$ – размер хеша в битах) функциями $f_{atm}(\ )$ и $f_{plm}(\ )$.

Значение функции $f_{atm}(\ )$ определяется координатой точки траектории отображения (1), $z = \lfloor (j/L) \cdot m_{i,j} \rfloor$ – порядковый номер точки, т.е. $f_{atm}\left(h_{i,j}, m_{i,j} \right) = x_z$. Значение управляющего параметра отображения на каждой итерации функции сжатия определяется как $a_{i,j} = (i/n + j/L)/2$, а начальная точка траектории как $x_0 = h_{i,j-1}$, где $h_{i,j-1}$ – выход предыдущей итерации $f(\ )$ или $h_{i,0} = m_{i,128}/256$ для ее первой итерации.

Значение функции $f_{plm}(\ )$ определяется как координатой точки траектории отображения (2), $z = \lfloor (1 - j/L) \cdot m_{i,j} \rfloor$ – порядковый номер точки, т.е. $f_{plm}\left(h_{i,j}, m_{i,j} \right) = x_z$. Значение управляющего параметра отображения на каждой итерации функции сжатия определяется как $\beta_{i,j} = a_{i,j}/2$, а начальное значение отображения – как выход функции $f_{atm}\left(h_{i,j}, m_{i,j} \right)$, т.е последняя точка траектории отображения (1).

Промежуточные хеши h_i строятся путем округления промежуточных значений $h_{i,j}$ функции сжатия $f(\ )$ после каждой итерации. При этом из полученных последовательностей (цифр 0 и 1) формируется битовый хеш соответствующей длины.

Этап финализации

На завершающем этапе функция финализации (финального сжатия) объединяет все промежуточные хеши h_i операцией «исключающее или» в хеш-значение всего исходного сообщения $H(M) = \bigoplus_{i=1}^{n} h_i$. Более полное описание метода можно найти в [1] и [2].

Общая характеристика предложенного метода

Как видно из описания хеш-функции на основе хаотических отображений с переменными параметрами, ее стойкость опирается на свойства хаотических отображений, а сама функция описывается достаточно простой математической моделью. При этом метод построения хеш-функции позволяет распараллелить вычисления.

Тем не менее, рассмотренная хеш-функция не является безопасной, и проведенное ниже исследование обнаружило в ней ряд уязвимостей, которые вызваны как несовершенством самой структуры функции, так и спецификой ее реализации на цифровых вычислительных средствах. Рассмотрим особенности реализации хаотических динамических систем на конечном множестве состояний.

Особенности реализации хаотических динамических систем на конечном множестве состояний

Как правило, областью определения хаотических отображений (систем) является ограниченное некоторым интервалом множество вещественных чисел, являющееся бесконечным. Например, для отображения (1) областью определения является множество вещественных чисел на интервале $X \in (0,1)$. Именно в пределах своей области определения хаотические системы, в частности, задаваемые рекуррентными уравнениями, демонстрируют хаотическое поведение и проявляют присущие им свойства, такие как: бесконечность, неповторяемость, непредсказуемость и некоррелированность значений (точек) траекторий, высокая чувствительность к начальным параметрам и т.д. В тоже время современные цифровые вычислительные средства при обработке данных ограничены разрядностью своих процессоров и объемами памяти, т.е. работают только с конечными подмножествами рациональных чисел, а точность представления данных зависит от характеристик вычислительных систем.

Таким образом, при реализации хаотических динамических систем на конечном множестве состояний в современных вычислительных средствах их траектории становятся цикличными (конечными), а длина цикла k зависит от точности представления данных и не может быть выше $k \le 2^L$, где L - точность представления данных в двоичной системе. Исследованию поведения и свойств хаотических динамических систем, задаваемых рекуррентными уравнениями, на конечном множестве состояний посвящено ряд работ, например [3, 4]. В этих же работах можно найти эвристические и экспериментальные оценки средней длины цикла для различных отображений при разной точности вычислений. Длина циклов, как правило, значительно меньше предельного значения 2^L, а вхождению в цикл предшествует некоторый (как правило, относительно небольшой) уникальный участок траектории. Длина уникального участка (длительность вхождения в цикл) и длина самого цикла зависят от структуры рекуррентного уравнения, выбранных начальных значений, управляющих параметров и точности представления данных (вычислений). Иллюстрация поведения хаотических систем на конечном множестве состояний в цифровых вычислительных системах схематически приведена на рис. 1.

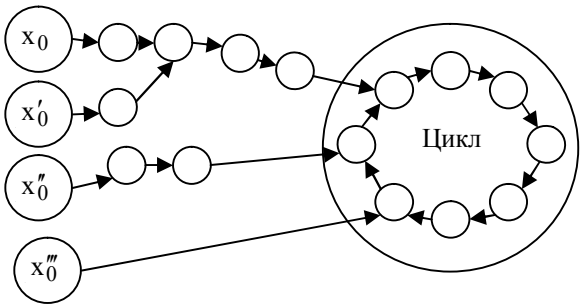

Рис. 1. Поведение хаотических систем на конечном множестве состояний

Следует отметить, что хаотические системы, их режимы работы и траектории при реализации на вычислительных средствах часто называются псевдохаотическими.

Применительно к рассматриваемой хеш-функции все данные представляются восьми битными двоичными числами. Хотя авторы явно не указали требуемую точность вычислений, целесообразно предположить (особенно с учетом потенциальных аппаратных реализаций), что и вычисления будут производиться с соответствующей точностью представления данных. Для упрощения дальнейшего анализа, а также с точки зрения простоты аппаратных реализаций и платформонезависимости результата вычислений, предположим, что вычисления организовываются в формате данных с фиксированной запятой.

Таким образом, при вычислении вектора $\vec{x}$ предложенные в работе [1] значения $x_0 = 0.7654$ и $a = 0.6$ при восьмибитной точности представления данных будут округлены до значений $x_0 \approx .11000100_2$, и $a \approx .10011010_2$, т.е. $x_0 \approx 0.765625$ и $a \approx 0.6015625$. При этом длина цикла отображения (1) составит 31 итерацию, а вхождение в цикл произойдет уже на первой итерации. Этот факт является немаловажным при поиске уязвимостей в предложенной хеш-функции. Также следует отметить, что траектории используемых отображений при малой точности представления данных являются неустойчивыми и требуется их коррекция для исключения сходимости к циклу 0-1, т.е. выхода за область определения.

Далее проанализируем предложенную функцию на уязвимости, вызванные особенностями реа-

лизации хаотических отображений на конечном множестве состояний.

Атака дополнением исходного сообщения

Анализируя общую структуру хеш-функции и в частности ее финализирующей функции можно показать, что при добавлении к исходному сообщению двух векторов $\vec{m}_i$ (или блоков таких векторов), дающих идентичный промежуточный хеш, хеш-значение $H(M)$ всего сообщения не изменится. Атака дополнением исходного сообщения является «классической» в криптоанализе хеш-функций, а устойчивость к этому виду атак – определяющим фактором при оценке функций.

Для успеха атаки дополнением сообщения на рассматриваемую хеш-функцию необходимо:

а) обеспечить выбор таких двух векторов $\vec{m}$ (либо блоков векторов), что их промежуточные хеши будут идентичны и при вычислении значения функции финального сжатия не повлияют на хеш-значение всего исходного сообщения $H(M) = \bigoplus_{i=1}^{n} h_i$;

б) устранить влияние дополняемых к исходному сообщению новых данных на хеш-значение исходного сообщения $H(M)$ путем:

1) устранить влияние изменений элементов матрицы M, кодирующих длину сообщения, на хеш-значение сообщения;

2) устранить влияние новых блоков данных на значения элементов векторов $\vec{c}$ и $\vec{r}$;

3) устранить эффект изменения переменных параметров хаотических отображений в функциях сжатия $f(\ )$ при увеличении количества строк матрицы M.

Описанные выше требования к организации атаки на хеш-функцию являются взаимосвязанными. Для устранения влияния новых блоков данных на значения элементов векторов $\vec{c}$ и $\vec{r}$ исходное сообщение необходимо дополнять блоками с числом векторов кратным длительности цикла траектории отображения (1), т.е значений вектора $\vec{x}$.

Предположим, что k – длительность цикла отображения (1) при заданных начальных условиях, т.е. $x_i = x_{i+k}$. В случае, если исходная матрица M будет дополнена двумя идентичными блоками векторов по k векторов в каждом (например, в качестве последних строк матрицы), то из выражения (3) следует, что новые значения элементов вектора $\vec{c}$ описываются следующим выражением:

$$c'_i = \left(c_i - x_{n+i-1} + \bigoplus_{j=n}^{n-1+2k} \left(m'_{j,i} \oplus x_j \right) \right) + x_{(n+2k)+i-1},$$

где c_i – значения элементов вектора для исходного варианта сообщения. Поскольку вследствие цикличности $x_{(n+2k)+i-1} = x_{n+i-1}$, $x_{j+k} = x_j$, а также при выполнении требования $m'_{j+k,i} = m'_{j,i}$ ($m'_{j,i}$ - элементы векторов, дополняющих исходное сообщение), то:

$$\bigoplus_{j=n}^{n-1+2k} \left(m'_{j,i} \oplus x_j \right) = 0, \left(x_{(n+2k)+i-1} - x_{n+i-1} \right) = 0,$$

а $c'_i = c_i$, т.е. вектор $\vec{c}$ после дополнения останется неизменным. Приведенные рассуждения верны как для случая дополнения исходного сообщения в его конце, так и при вставке новых данных в произвольную позицию кратную 127 байтам (длине вектора $\vec{m}'_i$) и при условии, что данные вставляются в цикличной области вектора $\vec{x}$.

Аналогичным образом можно показать, что при вставке новых данных не изменятся значения элементов $r'_i = r_i$ векторов исходного сообщения, кроме вектора предшествующего вставке, поскольку его 128 элемент $m_{i,128} = r_{i+1}$ рассчитывается со сдвигом по следующему вектору. Т.е., необходимо первый вектор $\vec{m}'_i$ (i – позиция/строка матрицы M с которой начинается дополнение) дополняемого блока из k векторов подбирать таким образом, что $r_i = r'_i = r'_{i+k} = r'_{i+2k}$, где r_i – значение элемента вектора $\vec{r}$ для исходного сообщения, а r'_i и r'_{i+k} – после его дополнения. В простейшем случае это достигается использованием в качестве первого вектора $\vec{m}'_i$ дополняемого блока вектора $\vec{m}_i$ исходного сообщения в соответствующей позиции i. Процесс дополнения можно проиллюстрировать следующим выражением (например, при дополнении матрицы M в последних ее строках):

$$M = \begin{bmatrix} m_{1,1} & m_{1,2} & \dots & m_{1,127} & r_2 \\ m_{2,1} & m_{2,2} & \dots & m_{2,127} & r_3 \\ \dots & \dots & \dots & \dots & \dots \\ m_{n-1,1} & m_{n-1,2} & \dots & m_{n-1,127} & r'_n = r_n \\ m'_{n,1} = c_{127} & m'_{n,2} = c_{126} & \dots & m'_{n,127} = c_1 & r'_{n+1} \\ \dots & \dots & \dots & \dots & \dots \\ m'_{n-1+k,1} & m'_{n-1+k,2} & \dots & m'_{n-1+k,127} & r'_{n+k} = r_n \\ m'_{n+k,1} = c_{127} & m'_{n+k,2} = c_{126} & \dots & m'_{n+k,127} = c_1 & r'_{n+1+k} \\ \dots & \dots & \dots & \dots & \dots \\ m'_{n-1+2k,1} = m'_{n-1+k,1} & m'_{n-1+2k,2} = m'_{n-1+k,2} & \dots & m'_{n-1+2k,127} = m'_{n-1+k,127} & r'_{n+2k} = r_n \\ c_{127} & c_{126} & \dots & c_1 & r_1 \end{bmatrix} = \begin{bmatrix} \vec{m}_1 \\ \vec{m}_2 \\ \dots \\ \vec{m}'_{n-1} = \vec{m}_{n-1} \\ \vec{m}'_n \\ \dots \\ \vec{m}'_{n+k-1} \\ \vec{m}'_{n+k} = \vec{m}'_n \\ \dots \\ \vec{m}'_{n+2k-1} = \vec{m}'_{n+k-1} \\ \vec{m}'_{n+2k} = \vec{m}_n \end{bmatrix}$$

Использование описанного выше алгоритма позволяет дополнить матрицу M двумя идентичными блоками данных, не поменяв структуру векторов $\vec{m}_i$, соответствующих исходному сообщению.

Для устранения влияния изменения блоков матрицы M, кодирующих длину, на хеш-значение сообщения достаточно осуществлять дополнение (вставку дополняемых данных) в конце исходного сообщения, при этом предпоследний вектор $\vec{m}'_{n+2k-1} = \vec{m}'_{n+k-1}$ должен содержать информацию о длине модифицированного сообщения согласно общих правил инициализации (формирования матрицы M).

Задачи устранения эффекта изменения значений параметров хаотических отображений в параллельных функциях сжатия при увеличении количества строк матрицы M, а также выбора двух векторов либо блоков векторов $\vec{m}$ таких, что их промежуточные хеши будут идентичны являются взаимосвязанными. Напомним, что параметры хаотических отображений в параллельных функциях сжатия вычисляются как $a_{i,j} = (i/n + j/L)/2$ и $\beta_{i,j} = a_{i,j}/2$, т.е. являются переменными и зависят от конкретной функции сжатия $f_i(\ )$ (относительной позиции обрабатываемого вектора $\vec{m}_i$ в матрице M) и номера итерации функции. Таким образом, необходимо обеспечить с одной стороны, чтобы все вектора дополняемых данных обрабатывались с одинаковыми параметрами a и β, а с другой – дополнение новых данных (увеличение количества строк n в матрице M) не влияло на параметры a и β для функций сжатия $f(\ )$ векторов исходного сообщения. Очевидно, что этого можно достичь с использованием особенностей реализации вычислений с ограниченной точностью представления данных.

Как было отмечено ранее, структура хеш-функции ориентирована на восьмибитное представление данных и точность вычислений. Таким образом, параметр a имеет не более 254 возможных значений, а β – не более 127. Относительная позиция обрабатываемого вектора $\vec{m}_i$ в матрице M при вычислении значения параметра a задается соотношением i/n. При достаточно больших значениях n (в частности $n > 256$) сразу несколько последовательно идущих строк (векторов $\vec{m}_i$) будут обрабатываться с одинаковыми значениями управляющих параметров отображений. Таким образом, при достаточной длине исходного сообщения (например $n > 254 \cdot 2 \cdot k$, где $2k$ – длина дополняющих исходное сообщение двух идентичных блоков данных, выбираемых в соответствии с методикой описанной выше) можно добиться того, что все дополняющие вектора будут обрабатываться функцией сжатия с одинаковыми параметрами a и β и позволят получить такие их промежуточные хеши, что:

$$\bigoplus_{i=n}^{n+k} h_i \oplus \bigoplus_{i=n+k}^{n+2k} h_i = 0 .$$

Однако, для устранения эффекта перераспределения относительной позиции обрабатываемых векторов $\vec{m}_i$ исходного сообщения при вставке новых строк в матрице M, число строк n в ней должно быть значительно больше значения $254 \cdot 2 \cdot k$. В частности n должно быть таким, что для любого i значение i/n и $i/(n+2k)$ при заданной точности вычислений (с учетом округлений) будет одинаково. Впрочем, успешную атаку дополнением можно организовать и для относительно небольших сообщений.

Предположим, что матрица M исходного сообщения имеет 255 строк. Напомним, что при заданных ранее ограничениях (предположениях) длина цикла траектории отображения (1) (т.е. вектора $\vec{x}$) составит 31 итерацию, а вхождение в цикл произойдет уже на первой итерации. Дополним исходное сообщение по алгоритму, описанному выше, 255-ю блоками данных по $2 \cdot 31$ векторов (по 127 байт в каждом векторе) каждый. Соответственно каждый блок пусть состоит из двух идентичных субблоков по 31 вектору. В каждом блоке первый вектор формируется по правилам описанным ранее и обеспечивает сохранение структуры элементов вектора $\vec{r}$ в матрице M (т.е. первый вектор дополняемого блока не должен влиять на предыдущий в позиции вставки), а последний вектор последнего блока должен содержать информацию о новой длине модифицированного сообщения. Остальные элементы дополняемых блоков (субблоков) выбираются произвольно. Блоки вставляются последовательно по одному после каждой строки (размером 127 байт) исходного сообщения. Таким образом, вставка новых данных в исходное сообщение не приведет к изменению промежуточных хешей векторов исходного сообщения, а промежуточные хеши вставляемых (новых) блоков компенсируют друг друга и не повлияют на хеш-значение всей функции. Данная атака на рассматриваемый вариант реализации хеш-функции была реализована в рамках практического эксперимента и найдена соответствующая коллизия.

Таким образом, рассматриваемая хеш-функция является уязвимой к атакам на дополнение сообщения, и нестойкой к коллизиям первого (с некоторыми оговорками, связанных с требуемой спецификой исходного сообщения) и второго рода соответственно.

Поиск коллизий без дополнения сообщения

Развивая подход к атакам на метод построения хеш-функции с распараллеливанием вычислений и

использованием хаотических отображений, основанный на особенностях их реализации на конечном множестве состояний, можно показать, что успешный поиск коллизий может быть осуществлен и без дополнения исходного сообщения. Для успеха такой атаки на рассматриваемую хеш-функцию необходимо внести такие изменения в исходное сообщение, которые не повлияют на хеш-значение $H(M) = \bigoplus_{i=1}^{n} h_i$. С одной стороны функция сжатия достаточно чувствительна к отдельным изменениям исходного сообщения и подбор векторов сообщения дающих одинаковый промежуточный хеш является достаточно трудоемким процессом (кроме случаев описанных в работе [2]). С другой – все элементы исходного сообщения в матрице M связываются с помощью векторов $\vec{c}$ и $\vec{r}$, а значит, внесение изменений только в одной строке матрицы с высокой вероятностью приведет также к изменениям других промежуточных хешей. Поэтому целесообразно искать коллизии внесением изменений в парах векторов $\vec{m}_i$ и $\vec{m}_j$ по аналогии с общими подходами рассмотренными в работе [2].

Предположим, что в исходном сообщении существуют векторы $\vec{m}_i$ и $\vec{m}_{i+k}$, такие что $\vec{m}_i = \vec{m}_{i+k}$, при этом k – длительность цикла отображения (1) при заданных начальных условиях, т.е. $x_i = x_{i+k}$ (в анализируемом варианте хеш-функции k кратно 31). Также предположим, что сообщение имеет длину такую, что в матрице M число строк $n > 254 \cdot k$, и векторы $\vec{m}_i$ и $\vec{m}_{i+k}$ обрабатываются функцией сжатия с одинаковыми параметрами a и β. При заданных ограничениях вектора $\vec{m}_i$ и $\vec{m}_{i+k}$ дают одинаковый промежуточный хеш и он не влияет на хеш-значение всего сообщения, а также элементы векторов $\vec{m}_i$ и $\vec{m}_{i+k}$ не влияют на элементы вектора $\vec{c}$, поскольку компенсируют друг друга (см. выражение (3) и (4)). Таким образом, если заменить вектора $\vec{m}_i$ и $\vec{m}_{i+k}$ на произвольный вектор $\vec{m}'_i = \vec{m}_i = \vec{m}_{i+k}$, такой что $r'_i = r_i = r_{i+k}$, то хеш-значение $H(M) = \bigoplus_{i=1}^{n} h_i$ функции не изменится. Данная атака была реализована в ходе практического эксперимента и была подтверждена ее успешность.

Другим вариантом этой атаки может быть случай, для которого вектора $\vec{m}_i$ и $\vec{m}_{i+k}$ не являются идентичными, но $r_i = r_{i+k}$ и $r_{i+1} = r_{i+k+1}$ (при прочих равных условиях). В этом случае при простой перестановкой векторов $\vec{m}_i$ и $\vec{m}_{i+k}$ хеш-значение $H(M) = \bigoplus_{i=1}^{n} h_i$ функции также не изменится (изменится только порядок вычисления промежуточных хешей). Эффективность этой атаки также была подтверждена практическим экспериментом.

Таким образом, рассматриваемая хеш-функция является уязвимой к атакам без дополнения сообщения, и нестойкой к коллизиям первого (с некоторыми оговорками, связанных с требуемой спецификой исходного сообщения) и второго рода соответственно.

Как в атаках с дополнением исходного сообщения так и без дополнения причина уязвимости кроется в недостаточном связывании элементов исходного сообщения в массиве M, несовершенстве последовательно-параллельной схемы хеш-функции, в которой промежуточные хеши объединяются простой операцией «исключающее или», особенностями реализации хаотических отображений на конечном множестве состояний.

Выводы

В данной работе, а также в работе [2] было показано, что:

1. В работе [1] был предложен оригинальный и перспективный механизм (метод) построения хеш-функции на основе хаотических отображений с переменными параметрами и параллельной организации вычислений. Данный метод использует достижения хаотической динамики для обеспечения стойкости и позволяет в значительной мере скомпенсировать недостатки криптографических примитивов, создаваемых на основе хаотических отображений. Таким недостатком является, в частности, высокая сложность организации вычислений, устраняемая путем их распараллеливания. Метод и его реализации ориентирован на мультиядерные (мультипроцессорные) платформы.

2. В тоже время структура метода имеет ряд недостатков, не позволяющих назвать его реализации (хеш-функции на его основе) криптографически стойкими. В ходе анализа было показано, что соответствующая хеш-функция не является стойкой к близким коллизиям, коллизиям первого и второго рода. Всего на рассматриваемую реализацию хеш-функции (построенную в рамках метода) было осуществлено пять различных вариантов успешных атак на поиск коллизий. Уязвимости функции можно разделить на две общих категории: вызванные несовершенством ее структуры и вызванные особенностями ее реализации на конечном множестве состояний.

3. Несмотря на выявленные недостатки, сам метод является перспективным и представляющим интерес для дальнейшего изучения. Для борьбы с выявленными уязвимостями необходимо в рассмотренном методе реализовать ряд мер, направленных на повышение стойкости путем:

– коррекции общей структуры метода путем более совершенной комбинации параллельной и последовательной итеративной схем, в частности (как вариант) наложения используемой схемы на структуру Меркла-Дамгарда, что позволит в значительной степени устранить структурные уязвимости метода;

– более корректного подбора параметров функций, коррекции структуры матрицы M и функции сжатия, направленные на устранение уязвимостей вызванных особенностями реализации хаотических отображений на конечном множестве состояний;

– повышение точности промежуточных вычислений, что позволит расширить пространство внутренних состояний и повысить стойкость функции (метода).

В дальнейших исследованиях будут обобщены полученные результаты и рассмотрены варианты развития и улучшения метода, которые позволят устранить выявленные уязвимости и построить более совершенные хеш-функции.

Список литературы

1. Yantao Li Parallel Hash function construction based on chaotic maps with changeable parameters [Text] / Yantao Li, Di Xiao, Shaojiang Deng, Qi Han, Gang Zhou // Neural Computing and Applications – 2011. – Vol. 20, №8: - P 1305-1312.

2. Антонов А.В. Анализ уязвимостей структуры хеш-функции на основе хаотических отображений с переменными параметрами и параллельной организации вычислений [Текст] / А.В. Антонов // Системи управління, навігації та зв'язку. – 2012. - № 2(22). – С. 157-162.

3. Wang S.H. Periodicity of chaotic trajectories in realizations of finite computer precisions and its implication in chaos communications [Text] / S.H. Wang, W.R. Liu, H.P. Lu, J.Y. Kuang, G. Hu // International Journal of Modern Physics. – 2004. - B 18(17-19). – P. 2617-2622.

4. Goitia C.B. A model for computational collapse of chaotic 1D maps [Text] / C.B. Goitia // International Symposium on Signals, Circuits and Systems, 2005 : Proceedings of ISSCS 2005 (14-15 July 2005). – 2005. – Vol. 2. – P. 741-744.

Поступила в редколлегию 11.07.2012

Рецензент: д-р техн. наук, проф. П.Ю. Костенко, Харьковский университет Воздушных Сил им. И. Кожедуба, Харьков.

АНАЛІЗ ВРАЗЛИВОСТЕЙ РЕАЛІЗАЦІЇ В ЦИФРОВИХ ОБЧИСЛЮВАЛЬНИХ СИСТЕМАХ ҐЕШ-ФУНКЦІЇ НА ОСНОВІ ХАОТИЧНИХ ВІДОБРАЖЕНЬ ЗІ ЗМІННИМИ ПАРАМЕТРАМИ Й ПАРАЛЕЛЬНОЇ ОРГАНІЗАЦІЇ ОБЧИСЛЕНЬ

А.В. Антонов

Досліджується геш-функція, побудована на основі хаотичних відображень зі змінними параметрами й паралельної організації обчислень. Показана її вразливість до атак, що використовують особливості реалізації хаотичних систем на кінцевій множині станів в цифрових обчислювальних системах. Доведено, що розглянута функція не є стійкою до виникнення колізій першого й другого роду.

Ключові слова: *геш-функція, хаотичне відображення, розпаралелювання, колізія.*

THE VULNERABILITY ANALYSIS OF IMPLEMENTATION IN DIGITAL COMPUTER SYSTEMS HASH FUNCTION BASED ON CHAOTIC MAPS WITH CHANGEABLE PARAMETERS AND PARALLEL CALCULATIONS

A.V. Antonov

Examines the hash function based on chaotic maps with changeable parameters and parallel calculations. It appeared to be weakness to attacks that rely on the implementation of chaotic systems on a finite set of states in the digital computer system. We prove that considered function is not collision resistance and 2nd pre-image resistance.

Keywords: *hash function, the chaotic map, paralleling, collision.*