Захист інформації

УДК 004.021+681.3.05

А.В. Антонов, В.Б. Бзот, И.Е. Кужель

Харьковский университет Воздушных Сил им. И. Кожедуба, Харьков

АКТУАЛЬНОСТЬ СОЗДАНИЯ МЕТОДОЛОГИИ ПРОЕКТИРОВАНИЯ ХЕШ-ФУНКЦИЙ НА ОСНОВЕ ХАОТИЧЕСКИХ ОТОБРАЖЕНИЙ

На примере исследования т.н. «хеш-функции на основе хаотических итераций», проанализированы причины возникновения недостатков при конструировании хеш-функций на основе хаотических отображений, показана актуальность создания методологии их проектирования (применения хаотических отображений для конструирования хеш-функций).

Ключевые слова: *хеш-функция, хаотическое отображение, методология проектирования, коллизия, качество.*

Введение

В последнее время в качестве одного из альтернативных подходов к конструированию хеш-функций все чаще предлагается использовать достижения теории динамического хаоса. Хаотические системы обладают рядом свойств, востребованных в этой сфере, как-то: высокая чувствительность к начальным значениям параметров (состояниям) хаотической системы (лавинный эффект), непредсказуемость на больших интервалах наблюдения, достаточно простые математические модели систем, облегчающие их анализ и изучение, возможность использования для исследований как аналитических методов, так и методов теории вероятности и т.д. Примерами хеш-функций на основе хаотических отображений (ХФоХО) являются: CHA-1, CBHF и другие.

В то же время при решении задачи конструирования ХФоХО разработчики сталкиваются с рядом проблем, как-то: достаточно высокие требования к вычислительным ресурсам со стороны таких хеш-функций, специфика поведения дискретных аппроксимаций хаотических отображений на ограниченном множестве состояний в цифровых вычислительных системах и т.д. Обобщенному анализу недостатков ХФоХО, а также выработке общих принципов применения дискретных аппроксимаций хаотических отображений для конструирования хеш-функций посвящена работа [1]. В данной публикации указано, что при соблюдении некоторых базовых принципов построения ХФоХО можно добиться достаточно высоких показателей их безопасности и эффективности. В то же время, несоблюдение этих общих приемов и правил зачастую приводит к значительным изъянам в предлагаемых разработчиками ХФоХО с альтернативными принципами построения. Проиллюстрируем данное утверждение на основе анализа ХФоХО, предложенной коллективом исследователей из Университета Франш-Конте, Франция. Вкратце дадим описание предложенной в работах [2, 3] ХФоХО.

Хеш-функции на основе хаотических итераций (безключевая и ключевая версии)

Рассмотрим предложенную в работе [2] ХФоХО, а также ее ключевую версию, получившую развитие в работе [3]. Данная ХФоХО имеет три основных этапа: инициализация т.н. «исходной конфигурации», инициализация т.н. «стратегии» и основные вычисления («хаотические» итерации). Следует отметить, что хотя в своих работах авторы стараются оперировать терминами хаотической динамики и указывать на связь работы именно с этой теорией, модель предложенной хеш-функции (и как следствие часть терминологии, в частности «стратегия» и «конфигурация») в значительной степени заимствована из теории клеточных автоматов. Рассмотрим вариант реализации ХФоХО, предложенный его авторами.

Инициализация исходной конфигурации

Отметим, что важность разработки корректного и эффективного алгоритма инициализации сложно переоценить, поскольку во многом именно эта процедура определяет стойкость хеш-функции к поиску коллизий для ряда атак, в частности связанных с дополнением исходного сообщения. Однако, несмотря на наличие достаточно подробных примеров, авторам не удалось привести четкое и формализованное описание этой процедуры, а также обоснование тех или иных принятых решений.

Так, авторы в тестовом примере оперируют текстовой строкой на латинице, которую заменяют

битовой строкой на основе численного представления латинских символов в ASCII кодировке. Причем авторы применяют семи битное представление символов, что является достаточно спорным решением. Так, для машинной обработки, как правило, используются восьми битное (или кратное восьми битам) представление данных. Именно на такую разрядность ориентировано подавляющее число современных вычислительных устройств.

Кроме того, при кодировке символов других алфавитов, как правило, используется минимум 8, а зачастую 16 или даже 32 бита (как то UTF-16 или UTF-32). Поэтому предложенное авторами решение не является рациональным, вычислительно оптимальным и универсальным. Оно может быть объяснено попытками авторов уменьшить избыточность англоязычного текста (старшие биты латинских символов всегда равны нулю), однако для этого существуют куда более эффективные алгоритмы (например, компрессоры), а любая хорошая хеш-функция не должна быть чувствительной к регулярностям в исходном сообщении. Более того, из описания, приведенного авторами, не ясно, каким образом осуществлять инициализацию исходной конфигурации при хешировании других последовательностей машинных данных, не связанных с представлением текста (машинного кода, изображений, компрессированных данных и т.д.) и в которых все биты данных являются значащими.

Также авторы при инициализации дописывают в конец сообщения значение его длины. Это общепринятая практика, позволяющая во многом противодействовать атакам с дополнением сообщения, однако длина сообщения записывается в семи значащих битах. Т.е. длина исходного сообщения ограничена 120 битами или 15 байтами, т.к. длина последовательности указывается в битах, а сама последовательность в процедуре в конечном итоге выравнивается с кратностью до 8 бит. Таким образом, из описания не ясно, как обрабатывать более длинные последовательности данных, а три последних бита блока, в котором хранится длина сообщения, всегда равны нулю. Т.е. и это решение является спорным и не универсальным, а изложение материала – методически не корректным.

При этом в описании этой процедуры авторы указывают, что позаимствовали ее из процедуры инициализации SHA-1 (которая является в некотором смысле «стандартом» инициализации данных для большинства хеш-функций). Однако авторы позаимствовали только «внешние признаки» (как то формальное дополнение и указание длины сообщения) процедуры инициализации исходного сообщения из SHA-1, без анализа принципов ее конструирования. Это привело к некорректной имплементации подходов к инициализации сообщения и к появлению изъянов в предложенной ХФоХО, что будет показано ниже.

Тем не менее, для проверки корректности реализации предложенной ХФоХО в виде программного обеспечения (приложения) она была воспроизведена в полном соответствии с описанными в работах [2, 3] правилами и процедурами и протестирована на эталонных тестовых примерах из этих же работ. А для дальнейшего исследования и анализа ХФоХО процедура инициализации исходного сообщения была вынужденно скорректирована на предмет обеспечения универсальности обработки различных данных с максимально возможным сохранением исходных подходов и решений из [2, 3]. Приведем эту скорректированную процедуру инициализации, которая выполняется в соответствии со следующим алгоритмом.

1. Исходное сообщение, подлежащее хешированию, представляется в виде битовой последовательности. В случае, если полученная последовательность не кратна восьми битам, она дополняется до необходимой длины единичными битами.

2. В конец этой битовой последовательности дописывается значение ее длины в байтах. Размер этого дописываемого блока со значением длины последовательности не является фиксированным, а выбирается исходя из необходимого для представления соответствующего значения числа бит. Размер дописываемого блока выбирается из последовательности 4, 12, 20, 28… бит, т.е. $l = 4 + 8 \cdot n$, где l – искомый размер блока (его длина) для вписывания длины последовательности, а n – коэффициент кратности.

3. Полученная битовая последовательность инвертируется, т.е. формируется ее дубликат с обратным порядком следования бит.

4. Полученные на шаге 2 и 3 последовательности объединяются, но между ними вставляется семи битовая последовательность вида «0001000» (рис. 1).

m_L	…	m_0	0001000	m_0	…	m_L

Рис. 1. Схема объединения последовательностей

На рис. 1 $m_0 ... m_n$ – дополненная (со значением длины) битовая последовательность, полученная на шаге 2.

5. Далее полученная на шаге 4 последовательность дублируется необходимое число раз (но не менее одного раза) для получения последовательности, длина которой будет не менее $l = 512 \cdot n$ бит, где l – искомая длина, а n – коэффициент кратности. После чего полученная битовая последовательность усекается до требуемого значения l. Полученная последовательность обозначается как D-последовательность и сохраняется в памяти.

6. Далее D-последовательность разбивается на блоки по 256 бит, которые последовательно объеди-

няются битовой операцией $\oplus$ – «исключающее или». Полученная на выходе шага №6 битовая последовательность (вектор $\vec{x}$ из 256 однобитовых элементов) и будет искомой исходной (начальной) конфигурацией ХФоХО.

Инициализация стратегии

Процедура инициализации стратегии описана в работах [2, 3] более формализовано, чем инициализации конфигурации, и выполняется в соответствии со следующим алгоритмом.

1. Сформированная ранее D-последовательность разбивается на блоки по 8 бит, каждый из которых интерпретируется в виде натурального числа $u_i \in [0,255]$ и из которых формируется вектор $\vec{u}$.

2. Далее D-последовательность модифицируется путем циклического сдвига всей последовательности влево на один бит. Модифицированная D-последовательность так же разбивается на блоки по 8 бит, которые представляются в виде натуральных чисел и которыми дополняется вектор $\vec{u}$.

3. Шаг 2 повторяется еще 6 раз, таким образом, чтобы вектор $\vec{u}$ содержал ровно *l* элементов (натуральных восьми битных чисел), где *l* – длина D-последовательности в битах.

4. На основе сформированного вектора $\vec{u}$ формируется последовательность (вектор) $\vec{s}$, элементы которого итеративно вычисляются как:

$$s_{i+1} = (u_{i+1} + 2 \cdot s_i + i) \bmod 256, \qquad (1)$$

где $i = 0,1,\ldots,|\vec{u}|$, $s_0 = u_o$ – для безключевой версии ХФоХО, или $s_0 = k$ – для ключевой версии, при этом k – секретный ключ (представленный натуральным числом). Полученная S-последовательность (или вектор $\vec{s}$) и есть искомая «стратегия» эволюции исходной конфигурации.

Основные вычисления

Процедура основных вычислений (хеша) является достаточно простой и описывается следующим выражением:

$$x'_{(s_i)} = \overline{x}_{(s_i)}, \qquad (2)$$

где $x'_{(s_i)}$ – новое значение элемента вектора $\vec{x}$ на итерации $i = 0,1,\ldots,|\vec{s}|$, $\overline{x}_{(s_i)}$ – применение операции битового отрицания к элементу вектора $\vec{x}$ (его предыдущему значению). Таким образом, в процессе эволюции системы (вектора $\vec{x}$) на каждой итерации $i = 0,1,\ldots,|\vec{s}|$ меняется состояние только одного элемента вектора $\vec{x}$. Индекс этого элемента на каждой конкретной итерации определяется элементами s_i вектора $\vec{s}$ (S-последовательности).

Описанная ХФоХО была реализована в виде программного обеспечения для дальнейшего анализа и тестирования на предмет вычислительной эффективности и стойкости к возникновению (поиску) коллизий.

Анализ вычислительной эффективности хеш-функции на основе хаотических итераций

Описанная ХФоХО не является вычислительно эффективной и оптимальной с позиции простоты реализации в современных вычислительных средствах. Так, помимо предостережений, высказанных при описании процедуры инициализации исходной конфигурации (которые были высказаны больше исходя из невозможности приведения корректного и формализованного ее описания в данной работе), ХФоХО имеет следующие изъяны.

1. Так, в процедуре инициализации исходной конфигурации ХФоХО активно используются битовые операции с данными без их выравнивания, т.е. часть операций оперирует данными, длина которых не кратна восьми битам. Такой подход не является рациональным и вычислительно оптимальным.

2. Хотя оценки затрат процессорного времени при вычислении хешей относительно невелики, затраты памяти являются значительными. Так, рассмотренная ХФоХО с учетом максимально возможной оптимизации (при последовательном вычислении S-последовательности в ходе основных итераций) требует выделения памяти объемом не менее двойной длины исходного сообщения (для хранения D-последовательности). Т.к. стремительное развитие и широкое внедрение информационных технологий приводит к значительному увеличению объемов обрабатываемых в информационных системах данных, такие затраты зачастую могут оказаться неприемлемыми. Это приводит к значительному сужению сферы применения ХФоХО, в частности крайне затруднительной является ее реализация в аппаратных средствах.

3. Кроме того, процедура инициализации исходной конфигурации сконструирована таким образом, что исключает возможность последовательной (потоковой) обработки данных (т.е. требуется предварительная загрузка всего исходного сообщения) и распараллеливания процессов вычисления, что также сужает сферу применения ХФоХО, в частности крайне затруднительной является ее реализация в информационно-управляющих системах реального времени и в телекоммуникационных системах.

Анализ уязвимостей хеш-функции на основе хаотических итераций

Хотя «хорошая» хеш-функция должна быть в достаточной мере вычислительно эффективной, гораздо большее значение с позиции оценки ее качества имеет наличие и характер ее уязвимостей

(стойкости к возникновению и поиску коллизий). Так, к хеш-функциям выдвигаются следующие основные требования стойкости [4]:

– необратимость или стойкость к восстановлению прообраза;

– стойкость к коллизиям первого рода или восстановлению вторых прообразов;

– стойкость к коллизиям второго рода.

Помимо основных требований к хеш-функциям часто выдвигаются дополнительные требования, такие как:

– устойчивость к близким коллизиям (near-collision);

– устойчивость к псевдоколлизиям;

– отсутствие корреляции между входом и выходом хеш-функции;

– устойчивость к нахождению частичного прообраза и другие требования.

Исходя из проведенного анализа рассмотренной ХФоХО, она не является стойкой к возникновению и поиску коллизий. Покажем это.

Рассмотрим для начала вариант поиска коллизии (атаки) для случая, когда на шаге 4 процедуры инициализации конфигурации прямая и обратная битовые последовательности разделяются не набором бит «0001000», а набором из семи единичных бит «1111111». В этом случае легко можно предложить несколько исходных сообщений разной длины, но с одинаковыми хешами. Например, одним из таких сообщений является последовательность из 15 байт вида 0xFF, а вторым – последовательность из 4095 таких же байт. Очевидно, что исходя из особенностей процедуры формирования исходной конфигурации, для обоих сообщений мы получим D-последовательность, состоящую из одних единичных бит, и как следствие конфигурацию из одних нулевых бит, а S-последовательность в виде регулярной (циклической) последовательности натуральных чисел. В конечном итоге на выходе хеш-функции мы получим одинаковые хеши для этих сообщений разной длины.

Этот вариант атаки дополнением исходного сообщения на упрощенную версию хеш-функции приведен для иллюстрации причин возникновения уязвимостей. Этими причинами являются.

1. «Слабость» процедур инициализации. Хотя, как было указано ранее, авторы ссылаются на заимствование подходов инициализации и схожесть процедур с их аналогами из SHA-1, такое утверждение явно не соответствует действительности. Процедуры инициализации должны решать задачи обеспечения унификации и универсальности применения хеш-функции и быть максимально формализованы для обеспечения воспроизводимости вычислений. Кроме того, эти процедуры предназначены для противодействия ряду атак, связанных с дополнением исходного сообщения. Обе эти задачи оказались не решены в рассматриваемой ХФоХО.

2. Регулярность процедуры формирования S-последовательности. Авторы в работах [2, 3] указывают, что выражение (1) является хаотическим отображением, однако это выражение описывает линейный конгруэнтный генератор, единственным источником неопределенности в котором является само исходное сообщение (элементы вектора $\vec{u}$). Авторы сравнивают это выражение с выражением вида:

$$x_{i+1} = 2 \cdot x_i \pmod 1, \qquad (3)$$

которое действительно является одной из форм записи для вырожденного кусочно-линейного хаотического отображения «зуб пилы»:

$$x_{i+1} = \begin{cases} \dfrac{x_i}{a}, & 0 \le x_i < a, \\ \dfrac{x_i - a}{1-a}, & a \le x_i \le 1, \end{cases} \qquad (4)$$

где $x_i \in (0,1)$ – точки траектории, $a \in (0,1)$ – управляющий параметр отображения (часто называемый ключевым). Однако в форме записи (3) отображение является вырожденным (т.е. значение ключевого параметра $a = 0.5$ приводит к вырождению хаотических режимов), а его аппроксимация над конечным множеством натуральных чисел (мощностью всего в 256 элементов) в выражении (1) – вообще не может рассматриваться в качестве хаотического отображения или сколь-нибудь корректной ее дискретной аппроксимацией. Таким образом, называть выражение (1) хаотическим отображением методологически неверно. Следует отметить, что в работе [3] авторы уже делают ремарку о том, что они используют термин «хаотические» в качестве прилагательного и не ассоциируют его с математической теорией хаоса, однако продолжают в работе использовать математический аппарат, разработанный для анализа и описания хаотических систем, что также методологически неверно. Точно так же нельзя рассматривать эволюции исходной конфигурации по выработанной стратегии в качестве хаотической системы, поскольку никакого отношения к теории нелинейных динамических систем эта модель не имеет, а описывается в рамках теории клеточных автоматов.

3. Простота основных вычислений хеша. По сути, этап основных вычислений не представляет никакой дополнительной сложности для атак на ХФоХО, а основной задачей аналитика является поиск регулярностей в S-последовательностях.

Таким образом, регулярность и линейность используемых в вычислениях выражений позволяет осуществить (по аналогии с атакой на упрощенную версию) атаку на полный вариант ХФоХО. Для этого достаточно построить исходное сообщение, дающее после операций дополнения и инвертирования последовательности-палиндромы. Используем, например, исходное сообщение из 4112 байт содержащих значение 0x01. В результате получим D-последова-

тельность вида «$(00000001)_{8230}$ 00000000 $(00000010)_{26}$». Сформируем второе исходное сообщение из 1052688 таких же байт. В результате получим D-последовательность вида «$(00000001)_{2105382}$ 00000000 $(00000010)_{26}$». Обе эти D-последовательности позволяют сформировать цикличные и практически идентичные по структуре S-последовательности (различающиеся только числом циклических участков), и получить в итоге одинаковый хеш (рис. 2).

482FFFD204010B4303032F3CFCFCFC43
0303203CFCFCFCC3033CFC43030303F9

Рис. 2. Значение хеша, полученное при моделировании атаки

Таким образом, рассматриваемая безключевая версия ХФоХО не является безопасной (стойкой к возникновению и поиску коллизий) и не удовлетворяет выдвигаемым к хеш-функциям требованиям.

Аналогичным образом нельзя рассматривать в качестве безопасной и ключевую версию ХФоХО, поскольку в соответствии с правилами модульной арифметики выражение (1) можно заменить следующим выражением

$$\begin{aligned} s_{i+1} &= \left(u_{i+1} + 2\cdot s_i + i\right) \bmod 256 \equiv \\ &\left(u_{i+1} + \left(2\cdot s_i\right) \bmod 256 + i\right) \bmod 256 \equiv . \\ &\left(u_{i+1} + \left(s_i\right) \bmod 128 + i\right) \bmod 256. \end{aligned} \qquad (5)$$

Учитывая, что в ключевой версии $s_0 = k$, k – секретный ключ, то из выражения (5) видно, что все пространство ключей ограничивается всего лишь 128 возможными значениями. Т.е. подбор нужного ключа методом грубой силы (последовательного перебора) не составляет вычислительной сложности.

Выводы

В работе выполнен анализ ХФоХО, предложенной коллективом исследователей из Университета Франш-Конте в работах [2, 3]. Несмотря на значительный интерес, который представляет возможность использования достижений теории динамического хаоса при конструировании хеш-функций, авторам не удалось предложить вычислительно эффективную и безопасную (стойкую к возникновению и поиску коллизий) хеш-функцию. Авторы допустили ряд методологичиских просчетов при проектировании ХФоХО и изложении (представлении) результатов исследований. Авторы не провели сколь-нибудь значимого анализа сконструированной хеш-функции, в том числе и статистического. Так, выполнение статистической экспресс-оценки безопасности хеш-функций является в некотором смысле стандартом методологии оценки их качества. А для рассматриваемой ХФоХО ее анализ любым из широко распространенных пакетов статистических тестов (например NIST STS) с очень высокой долей вероятности вскрыл бы недостатки в ее структуре еще на раннем этапе проведения исследований. В работах [2, 3] присутствует некоторый экспериментальный анализ ХФоХО, но он в большей мере носит наглядный и иллюстративный характер, но не может считаться анализом и оценкой качественных характеристик ХФоХО. Кроме того, в качестве критериев и мер оценки качества предложенной ХФоХО авторы пытаются использовать инструменты теории хаотической динамики, что также методологически не верно, поскольку ни предложенная хеш-функция ни ее структурные элементы не могут рассматриваться в роли хаотической системы. Неверные допущения при конструировании хеш-функции (в частности о том, что выражение (1) описывает рекуррентное уравнение эволюции хаотического отображения) привели к неверному применению инструментов анализа, некорректным результатам анализа и ошибочным выводам. Еще большее недоумение вызывает предложенный вариант ключевой версии ХФоХО, поскольку в нем мощность множества значимых ключей ограничена всего 128 значениями.

Таким образом, анализ реализации ХФоХО, рассмотренной в [2, 3], подтверждает необходимость создания методологии применения дискретных аппроксимаций хаотических отображений для конструирования хеш-функций, например на основе принципов, изложенных в работе [1]. Методология конструировании новых ХФоХО также должна включать механизмы оценки их качества, в частности, вычислительной эффективности при реализации в различных программных и аппаратных платформах, экспресс-оценку статистической безопасности, оценку устойчивости к наиболее распространенным видам «традиционных» для хеш-функций атак (как то на основе парадокса о днях рождениях, атаки расширением, дополнением и т.д.), анализ с учетом специфики реализации хаотических отображений в дискретных вычислительных системах. Нарушение этих подходов и условий приводит, как правило, к неудовлетворительным результатам и зачастую дискредитирует саму идею использования хаотических отображений (вернее их дискретных аппроксимаций) для построения хеш-функций, что и подтверждается, например, рассмотренной ХФоХО.

Список литературы

1. Антонов А.В. Общие принципы применения дискретных аппроксимаций хаотических отображений для конструирования хеш-функций / А.В. Антонов, И.Е. Кужель, Н.В. Шигимага // Системи обробки інформації. – Х.: ХУ ПС, 2013. – Вип. 8(115). – С. 189-194

2. Bahi J.M. Hash Functions Using Chaotic Iterations [Text] / Jacques M. Bahi, Christophe Guyeux // Journal of Algorithms & Computational Technology – 2010. – Vol. 4, №2: – P. 167-181. (Режим доступа к он-лайн версии статьи: http: www/hal.archives-ouvertes.fr/docs/00/56/31/13/PDF/bg10_ij.pdf).

3. Bahi J.M. Performance Analysis of a Keyed Hash Function based on Discrete and Chaotic Proven Iterations

[Text] / Jacques M. Bahi, Jean-François Couchot, and Christophe Guyeux // INTERNET 2011: The Third International Conference on Evolving Internet. – 2011. – P. 52-57 (Режим доступа: http: www/arxiv.org/pdf/1112.1271v1.pdf).

4. Al-Kuwari S. Cryptographic Hash Functions: Recent Design Trends and Security Notions [Text] / Saif Al-Kuwari, James Davenport, Russell Bradford // Proceedings of Inscrypt'10 – Science Press of China, 2010. – P. 133-150 (Режим доступа к полной он-лайн версии доклада: http: www/eprint.iacr.org/2011/565).

Поступила в редколлегию 29.10.2013

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

АКТУАЛЬНІСТЬ СТВОРЕННЯ МЕТОДОЛОГІЇ ПРОЕКТУВАННЯ ҐЕШ-ФУНКЦІЙ НА ОСНОВІ ХАОТИЧНИХ ВІДОБРАЖЕНЬ

А.В. Антонов, В.Б. Бзот, І.Є. Кужель

На прикладі дослідження т.зв. «геш-функції на основі хаотичних ітерацій», проаналізовано причини виникнення недоліків при конструюванні геш-функцій на основі хаотичних відображень, показана актуальність створення методології їх проектування (застосування хаотичних відображень для конструювання геш-функцій).

***Ключові слова**: геш-функція, хаотичне відображення, методологія проектування, колізія, якість.*

ACTUALITY OF DEVELOP DESIGNING METHODOLOGY OF HASH FUNCTION BASED ON CHAOTIC MAPS

A.V. Antonov, V.B. Bzot, I.Ye. Kuzhel

With studies of so-called "Hash function based on chaotic iterations," the causes of defects in the construction of hash functions based on chaotic maps were analyzed, and were shown the actuality of develop methodology of their design (using of chaotic maps to constructing a hash function).

***Keywords**: hash function, the chaotic map, develop methodology, collision, quality.*