УДК 681.518

АЛГОРИТМ НА ОСНОВЕ МЕХАНИЗМОВ ХАОТИЧЕСКОЙ ДИНАМИКИ ДЛЯ КОНТРОЛЯ ЦЕЛОСТНОСТИ ЭЛЕКТРОННЫХ ДОКУМЕНТОВ

©2012 В. М. Довгаль[1], В. В. Гордиенко[2], М. О. Никитин[3]

[1]докт. техн. наук, профессор каф. программного обеспечения и администрирования информационных систем,
e-mail: vmdovgal@yandex.ru
[2]канд. техн. наук, ст. науч. сотрудник МНМЦВИТ
e-mail: vvgord@yandex.ru
[3]аспирант каф. программного обеспечения и администрирования информационных систем
e-mail: vilyx@yandex.ru

Курский государственный университет

В статье рассматривается способ применения дискретных отображений для генерации детерминированно-хаотических числовых рядов с целью решения задачи формирования и повышения криптостойкости хеш-функций. Проведен сопоставительный анализ разработанной хеш-функции и аналога в виде метода MD5. По результатам анализа установлено, что предлагаемая хеш-функция имеет высокий барьер по отношению к информационным атакам.

Ключевые слова: хэш-функция, защита информации, криптография, дискретное отображение, целостность файла.

Внимание специалистов в сфере управления и обмена электронными документами привлекают проблемы профилактики и своевременного обнаружения несанкционированных изменений содержательной части электронных документов или сообщений в условиях растущего тренда компьютерных преступлений. Традиционно для этих целей применялись и применяются хеш-функции (hash-function). К числу популярных хэш-функций в настоящее время эксперты относят *MD*5 и *SHA*-1. Вместе с тем они неудовлетворительны с точки зрения практики применения в условиях современного уровня методов криптоанализа и осуществления информационных атак. В феврале 2005 г. появилось первое сообщение о потенциальной уязвимости, впервые обнаруженной в алгоритме *SHA*-1 [Wang 2005]. Так как *SHA*-1 сегодня фактически является общепринятым стандартом для хеш-функций и он включен в состав множества функционирующих систем защиты информации, известие о признаках его уязвимости по отношению к информационным атакам было воспринято компьютерным сообществом очень напряженно. Среди первых эту проблему поднял американский Национальный институт стандартов и технологий, издавший спецификации *SHA*-1 в качестве *FIPS* 180-2 – официального федерального стандарта США на обработку информации. Новый безопасный алгоритм формирования дайджестов находится в стадии тестирования и верификации [*NIST* 2012].

Цель данной работы заключается в повышении скорости и криптостойкости хэширования с использованием современных достижений теории хаотических систем.

Рассмотрим в качестве прототипа *Secure Hash Algorithm,* который был разработан национальным институтом стандартов и технологии (*NIST*) и опубликован в качестве федерального информационного стандарта (*FIPS PUB* 180) в 1993 году. *SHA*-1, так же как и *MD5*, основан на алгоритме *MD4*.

Рассмотрим алгоритм выполнения *SHA*-1. На вход подаётся сообщение максимальной длины 2^{64} бит. На выходе создается дайджест сообщения длиной 160 бит.

Функционирование алгоритма *SHA*-1 основывается на выполнении следующей последовательности действий. Сообщение дополняется так, чтобы его длина была кратна 448 по модулю 512. Число добавляемых битов находится в диапазоне от 1 до 512. Добавление представляет собой одну единицу и необходимое количество нулей.

Затем к сообщению добавляется блок из 64 битов. Это 64-битное целое число без знака отражает длину исходного сообщения.

На этом этапе формируется сообщение, длина которого кратна 512 битам. Расширенное (дополненное) сообщение представляет собой последовательность 512-битных блоков длиной *L*, то есть общая длина расширенного сообщения равна L×512 бит и кратна шестнадцати 32-битным словам.

Для инициализации *SHA*-1 используется 160-битный буфер для хранения промежуточных и окончательных результатов хэш-функции. Буфер можно представить как пять 32-битных регистров *A, B, C, D* и *E*. Эти регистры инициализируются следующими шестнадцатеричными числами:

A = 67452301
B = *EFCDAB*89
C = 98*BADCFE*
D = 10325476
E = *C*3*D*2*E*1*F*0

Затем проводится обработка сообщения в виде 512-битных блоков.

Основой алгоритма является модуль, состоящий из 80 циклических обработок, обозначенный как H_{SHA}. Все 80 циклических обработок имеют одинаковую структуру.

Каждый цикл получает на входе текущий 512-битный обрабатываемый блок и 160-битное значение буфера *ABCDE* и изменяет содержимое этого буфера.

В каждом цикле используется дополнительная константа K_t, которая принимает только четыре различных значения:

$0 <= t <= 19$ $K_t = 5A827999$
(целая часть числа $[2^{30} \times 2^{1/2}]$);
$20 <= t <= 39$ $K_t = 6ED9EBA1$
(целая часть числа $[2^{30} \times 3^{1/2}]$);
$40 <= t <= 59$ $K_t = 8F1BBCDC$
(целая часть числа $[2^{30} \times 5^{1/2}]$);
$60 <= t <= 79$ $K^t = CA62C1D6$
(целая часть числа $[2^{30} \times 10^{1/2}]$).

Для получения хэша следующей части сообщения выход 80-го цикла складывается со значением хэша предыдущей. Сложение по модулю 2^{32} выполняется независимо для каждого из пяти слов в буфере с каждым из соответствующих слов в предыдущем хэше.

После обработки всех 512-битных блоков выходом *L*-й стадии является 160-битный дайджест сообщения.

Приведем пошаговую реализацию алгоритма формирования хэш-функции с использованием детерминированной хаотической последовательности чисел. Генерируя дискретную хаотическую последовательность и корректируя её данными, получим дайджест, соответствующий этим данным.

В качестве генератора детерминированно-хаотической последовательности взято дискретное отображение Лоренца

$x_{i+1} = 1 - 2 \times |x_i|$.

Все операции производятся над числами с плавающей запятой с двойной точностью.

Шаг 1. Задаем инициализирующее (стартовое) значение хаотической последовательности числом, полученным в результате следующей операции:

x_i = 0*xFFFFFFFFFFFFFFFF*/0*x*10000000000000.

Шаг 2. Генерируем следующее значение хаотической последовательности.

Шаг 3. Берём очередной байт из предоставленных данных и делим его значение на 1000.

Шаг 4. Вычитаем полученное значение из значения детерминированно-хаотической последовательности, если сгенерированное значение хаотической последовательности больше нуля, в ином случае прибавляем.

Шаг 5. Делаем 10 последовательных генераций хаотической последовательности без смешивания с данными.

Шаг 6. Если все данные прошли смешивание, то дайджестом является знак и мантисса последнего элемента хаотической последовательности. Иначе возвращаемся к шагу 2.

Для теста производительности было написано приложение, которое на вход получало случайный файл, а на выходе выдавался размер полученного файла и время, за которое был обработан разработанным алгоритмом данный файл (рис. 1, 2).

На этих рисунках обозначены: по горизонтальной оси – объём данных в байтах, а по вертикальной – время обработки этого объёма в миллисекундах.

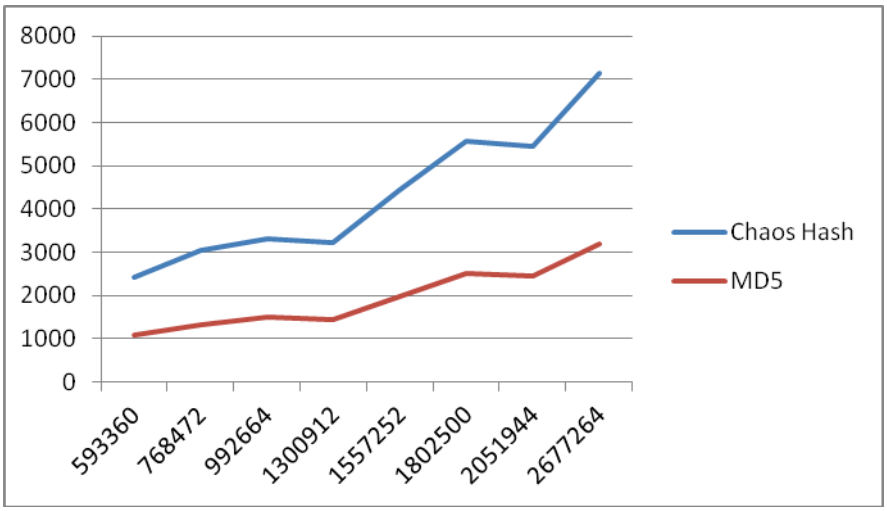

Рис. 1. График сравнения результатов скорости работы *MD5* и данного алгоритма

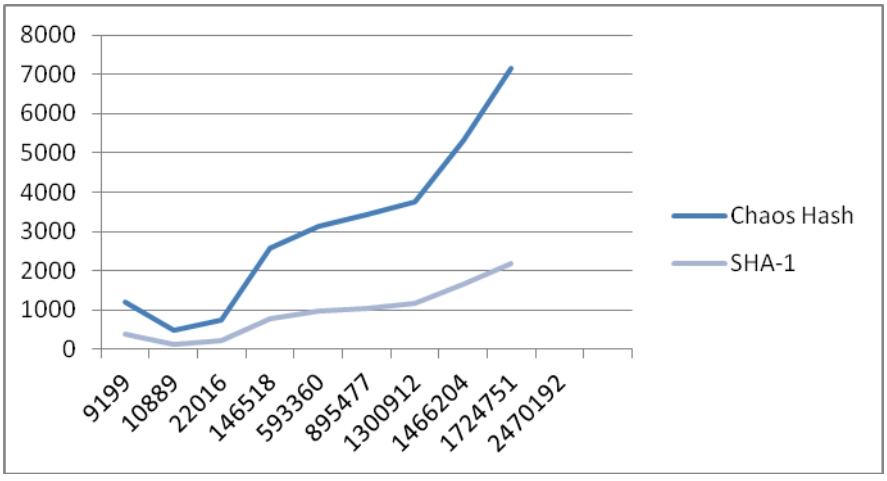

Рис. 2. График сравнения результатов скорости работы *SHA*-1 и разработанного алгоритма

Как видно из графиков, использование дискретных отображений, порождающих детерминированно-хаотические процессы для создания хэш-функции, затрудняет для

лиц, осуществляющих информационную атаку, подбор файлов электронных документов, имеющих вычисленный контрольный дайджест. А непредсказуемость поведения хаотической системы и огромное многообразие орбит дискретных отображений обеспечивают главные требования, предъявляемые к хэш-функции:

1) необратимость, вытекающая из свойства дискретного отображения Лоренца;

2) стойкость к коллизиям первого рода: для заданного сообщения *M* практически невозможно подобрать другое сообщение *N*, для которого *H(N)=H(M),* что обеспечивается сильной зависимостью орбиты дискретного отображения от величины приращений, определяемых значениями каждого байта файла электронного документа;

3) стойкость к коллизиям второго рода: практическая невозможность подобрать пару сообщений, имеющих одинаковый дайджест, как результат хеширования.

Библиографический список

Wang X., Yin Y.L., Yu H. Finding collisions in the full SHA-1 // Advances in cryptology – CRYPTO 2005 // Lecture Notes in Computer Science. Berlin: Springer-Verlag, 2005. V. 3621. P. 17–36.

NIST принял решение: SHA-3 будет использовать алгоритм Keccak [Сайт]. URL: http://habrahabr.ru/post/153349/ (дата обращения: 3.10.2012).