О повышении криптостойкости однонаправленных хеш-функций

В. Ю. Лёвин

В статье приводится конструктивные предложения решения задачи обеспечения подлинности и достоверности цифровых документов с использованием однонаправленных хеш-функций. Численно оценена стойкость однонаправленных хеш-функций при различных видах их взлома. Предложен ряд алгоритмов позволяющих серьезно повысить криптостойкость хеш-функций без переделки их внутренных алгоритмов. Среди предложенных методов по повышению криптостойкости выбран лучший по скорости и качеству. Показано, что метод суффиксной суперпозиции, предложенный Б. Шнайером, не годится для использования в этих целях. Предложенные в статье методы могут быть использованы для улучшения большинства однонаправленных хеш-функций, таких как, MD4, MD5, RIPEMD, SHA, ГОСТ 34 11–94.

Ключевые слова: однонаправленные хеш-функции, метод Шнайера, целостность, информационная безопасность.

Применение хеш-функций довольно обширное, однако, основное их предназначение заключается в составлении уникального идентификационного кода передаваемого сообщения (message digest). Подобная задача особенно актуальна в электронном документообороте и технологии цифровой подписи (DSA, DSS). Каждый человек имеет уникальные: отпечатки пальцев, сетчатки глаза, строение и состав ДНК; каждое сообщение имеет уникальный идентификационный код (хеш-значение), но с одной оговоркой — вероятность существования двух одинаковых кодов пренебрежимо мала. Рассмотрим данный вопрос подробнее. Пусть $M \in \{0,1\}^*$ — произвольное цифровое сообщение.

Определение. Функция $h(M): \{0,1\}^* \to \{0,1\}^n$, $n \in \mathbb{N}$ называется односторонней хеш-функцией порядка n, если выполнены следующие условия:

1) Значение хэш-функции h должно быть определено для любого цифрового сообщения $M \in \{0,1\}^*$;
2) Для любого цифрового сообщения $M \in \{0,1\}^*$ хэш-функция h имеет фиксированный порядок $n \in \mathbb{N}$;
3) $\forall M \in \{0,1\}^*$ значение $h(m)$ вычисляется за полиномиальное время;
4) $\forall M_1 \in \{0,1\}^*$ вычислительно сложно найти $M_2 \in \{0,1\}^*: M_1 \neq M_2$, $h(M_1) = h(M_2)$;
5) Вычислительно невозможно найти произвольную пару (M_1, M_2): $M_1 \neq M_2$, $M_i \in \{0,1\}^*$, $i = 1,2$, такую что $h(M_1) = h(M_2)$.

Заметим, что существует много различных определений хеш-функций, однако приведенное выше определение является определением классической хеш-функции применяющейся в криптографии [1]. Свойства 4, 5 являются важнейшими криптографическими свойствами. Действительно, рассмотрев: циклически избыточный код CRC, побитный XOR (или ротационный RXOR), можно установить, что все они не удовлетворяют свойством 4, 5. Поэтому использовать данные алгоритмы для решения задачи аутентификации (обеспечения подлинности и достоверности) как криптографически хорошие хеш-функции нецелесообразно. Злоумышленнику не составит особого труда подделать исходное сообщения таким образом, чтобы получить сообщения с одинаковым хеш-значением. Отметим, что задача обеспечения достоверности является одной из ключевых задач в криптографии. Мы часто сталкиваемся с подобной задачей: отправляя от нашего имени распоряжение в банк, отдавая команды, высылая договоры и, наконец, скачивая файлы из сети Интернет.

В настоящее время существует довольно много различных хеш-функций, предложенных для решения задачи аутентификации. Описание соответствующих алгоритмов можно найти в [1, 2]. Оценим криптографическую стойкость однонаправленных хеш-функций.

Так как множество аргументов хеш-функции счетно, а значения имеют определенный фиксированный порядок, то коллизии неизбежны. Цель злоумышленника научиться заготавливать коллизии, то есть научиться фальсифицировать хеш-значения. Рассмотрим трудоемкости основных, необходимых злоумышленнику, процедур.

Грубый взлом хеш-функций (метод простого перебора)

Предположим злоумышленнику известен алгоритм построения хеш-функции h, первоначальное сообщение $M \in \{0,1\}^*$ и хеш-значение $h(M)$. Требуется найти $N \in \{0,1\}^*$ такое, что $h(M) = h(N)$. Для произвольных $M, N \in \{0,1\}^*$ вероятность того, что $h(M) = h(N)$, равна 2^{-n}, где n — порядок хеш-функции h. Обозначим, через $P_1(k,n)$ вероятность того, что для фиксированного $M \in \{0,1\}^*$ и $N_1, \ldots, N_k \in \{0,1\}^*$ существует номер $i = 1, \ldots, k : \ h(M) = h(N_i)$. Тогда получим:

$$P_1(k,n) = 1 - (1 - 2^{-n})^k \approx k2^{-n}.$$

$P_1(k,n)$	k
0,01	$\approx 2^{n-7}$
0,5	$\approx 2^{n-1}$
0,99	$\approx 2^{n}$

Таблица 1.

В таблице 1 представлены значения вероятности $P_1(k,n)$ и длинны перебора текстов k. Как следует из данной таблицы, злоумышленнику для подделки хеш-значения порядка n фиксированного текста методом подбора потребуется перебрать не менее 2^{n-7} текстов. При этом вероятность успеха будет не выше 1%. Данное свойство иллюстрирует, что для взлома фиксированного значения перебор не годится. Действительно уже для 128 битных хеш-значений (MD2, MD4, MD5, RIPEMD128, HAVAL3–4) потребуется перебрать около 2^{120} текстов. Осуществить это за обозримое время невозможно.

Взлом хеш-функций на основе парадокса дней рождений

Атака на хеш-функции на основе парадокса дней рождений является одной из самых распространенных атак. Рассмотрим следующую задачу: обозначим через $P_2(n,k)$ вероятность того, что на множестве из k элементов, каждый из которых может принимать 2^n значений, есть хотя бы два с одинаковыми значениями. Выведем формулу для $P_2(n,k)$. Число различных способов выбора элементов таким образом, чтобы при этом не было дублей, равно $2^n(2^n-1)\cdot\ldots\cdot(2^n-k+1) = \frac{2^n!}{(2^n-k)!}$. Всего возможных способов выбора элементов 2^{kn}. Следовательно, $P_2(n,k) = 1 - \frac{2^n!}{(2^n-k)!}\cdot 2^{-kn}$. Заметим, что $P_2(n,k) = 1 - (2^n\cdot(2^n-1)\cdot\ldots\cdot(2^n-k-1))\cdot 2^{-kn} = 1-\left(\frac{2^n-1}{2^n}\cdot\frac{2^n-2}{2^n}\cdot\ldots\cdot\frac{2^n-k+1}{2^n}\right) = 1-\left(\left(1-\frac{1}{2^n}\right)\cdot\left(1-\frac{2}{2^n}\right)\cdot\ldots\cdot\left(1-\frac{k-1}{2^n}\right)\right)$.

Используя то, что $1-x \leqslant e^{-x}$, получаем, $P_2(n,k) > 1 - e^{-\frac{k(k-1)}{2^n}}$.

$P_2(k,n)$	k
0,01	$\approx 2^{\frac{n}{2}-3}$
0,5	$\approx 2^{\frac{n}{2}}$
0,99	$\approx 2^{\frac{n}{2}+2}$

Таблица 2.

Как следует из таблицы 2 при $k = 2^{\frac{n}{2}}$ вероятность найти коллизию больше 50%. Подобный результат называется «парадоксом дня рождения» потому, что в соответствии с приведенными выше рассуждениями, для того чтобы вероятность совпадения дней рождения у двух человек была больше 0,5, в группе должно быть всего 23 человека. Этот результат кажется удивительным, возможно, потому, что для каждого отдельного человека в группе вероятность того, что с его днем рождения совпадет день рождения кого-то другого в группе, достаточно мала. Подобная атака показывает, что порядок хеш-значения должен быть более 256 бит, чтобы сделать перебор невозможным в современных условиях (2^{128} пока остается неосуществимым в реальных условиях). Однако, множество 128-битных хеш-функций можно удачно атаковать и в настоящее время. Но не стоит считать, что 256 битные хеш-функции надежны. В соответствии с законом Мура, мощность микропроцессоров увеличивается в 10 раз за каждые 6 лет, поэтому криптостойкость хеш-функций это вопрос времени.

Взлом хеш-функций за линейное время (туннельный эффект)

Данная атака основывается на специфике построения большинства хеш-функций. Действительно, большинство хеш-функций основаны на функции сжатия (или сдвиговой функции). Текст M разбивается на блоки M_i (в большинстве случаев размером в 512 бит) и включается итерационный процесс подсчета хеш-значения $h_i = f(h_{i-1}, M_i)$. Так как длинна раунда небольшая, то применяя дифференциальный анализ возможно найти такой текст ΔC : $h(M_i + \Delta C) = h(M_i)$. Приведем пример. Рассмотрим широко распространенную однонаправленную хеш-функцию MD4, предложенную Роном Ривестом. MD4 обладает 128 битным выходным значением, и является самой быстрой хеш-функцией. Пусть, согласно [3], $\Delta C = (0, 2^{31}, 2^{31} - 2^{28}, 0, 0, 0, 0, 0, 0, 0, 0, 0, -2^{16}, 0, 0, 0)$. Тогда для любого текста M: MD4$(M + \Delta C) =$ MD4(M). Соответствующие результаты приведенены в таблице 3.

Другой пример использования туннельного эффекта можно привести для RIPEMD128. RIPEMD была разработана для RIPE Европейского сообщества [2]. Данная хеш-функция представляет собой измененный вариант MD4, использующий другие циклические сдвиги и порядок слов сообщения. Туннельный эффект в этом случае выглядит следующим образом: $\Delta C = (0, 0, 0, 2^{20}, 0, 0, 0, 0, 0, 0, 2^{18} + 2^{31}, 0, 0, 0, 0, 2^{31})$. Тогда для любого текста M: RIPEMD128$(M + \Delta C) =$ RIPEMD128(M) (таблица 4).

Данная атака является самой эффективной, ведь ее трудоемкость минимальна и сводится к простым вычислениям.

Рассмотрев основные атаки можно сделать вывод о множестве недостатков в однонаправленных хеш-функциях. Использовать хеш-функции такие как MD4, MD5, RIPEMD128, SHA1 стало небезопасно, надежность остальных остается под вопросом. Еще одной пробле-

M_i	4d7a9c83 56cb927a b9d5a578 57a7a5ee de748a3c dcc366b3 b683a020 3b2a5d9f c69d71b3 f9e99198 d79f805e a63bb2e8 45dd8e31 97e31fe5 2794bf08 b9e8c3e9
M_2	4d7a9c83 d6cb927a 29d5a578 57a7a5ee de748a3c dcc366b3 b683a020 3b2a5d9f c69d71b3 f9e99198 d79f805e a63bb2e8 45dc8e31 97e31fe5 2794bf08 b9e8c3e9
$\mathrm{MD4}(M_1) = \mathrm{MD4}(M_2)$	**5f5c1a0d 71b36046 1b5435da 9b0d807a**
M_1	4d7a9c83 56cb927a b9d5a578 57a7a5ee de748a3c dcc366b3 b683a020 3b2a5d9f c69d71b3 f9e99198 d79f805e a63bb2e8 45dd8e31 97e31fe5 f713c240 a7b8cf69
M_2	4d7a9c83 d6cb927a 29d5a578 57a7a5ee de748a3c dcc366b3 b683a020 3b2a5d9f c69d71b3 f9e99198 d79f805e a63bb2e8 45dc8e31 97e31fe5 f713c240 a7b8cf69
$\mathrm{MD4}(M_1) = \mathrm{MD4}(M_2)$	**e0f76122 c429c56c ebb5e256 b809793**

Таблица 3.

мой является то, что перечисленные хеш-функции стандартизованы и зашиты во множество служебных библиотек (PGP) и программ MS Windows, Linux. Появившиеся бреши становятся серьезной угрозой. Предложим несколько методов по увеличению криптостойкости однонаправленных хеш-функций. Предположим у нас имеется произвольная однонаправленная хеш-функция $h(\cdot)$ порядка n. Рассмотрим несколько методов повышения ее криптостойкости.

Метод последовательной суффиксной суперпозиции

Суть данного метода заключается в следующем. Пусть $M \in \{0,1\}^*$ — произвольный текст. Тогда получим хеш-значение данного текста согласно следующему правилу:

$$\overline{h}(M) = \cdots h(M \| h(M \| h(M))).$$

M_1	579faf8e 9ecf579 574a6aba 78413511 a2b410a4 ad2f6c9f b56202c 4d757911 bdeaae7 78bc91f2 47bc6d7d 9abdd1b1 a45d2015 817104ff 264758a8 61064ea5
M_2	579faf8e 9ecf579 574a6aba 78513511 a2b410a4 ad2f6c9f b56202c 4d757911 bdeaae7 78bc91f2 c7c06d7d 9abdd1b1 a45d2015 817104ff 264758a8 e1064ea5
RIPEMD128(M_1) = = RIPEMD128(M_2)	**1fab152 1654a31b 7a33776a 9e968ba7**
M_1	579faf8e 9ecf579 574a6aba 78413511 a2b410a4 ad2f6c9f b56202c 4d757911 bdeaae7 78bc91f2 47bc6d7d 9abdd1b1 a45d2015 a0a504ff b18d58a8 e70c66b6
M_2	579faf8e 9ecf579 574a6aba 78513511 a2b410a4 ad2f6c9f b56202c 4d757911 bdeaae7 78bc91f2 c7c06d7d 9abdd1b1 a45d2015 a0a504ff b18d58a8 670c66b6
RIPEMD128(M_1) = = RIPEMD128(M_2)	**1f2c159f 569b31a6 dfcaa51a 25665d24**

Таблица 4.

Описываемый метод предлагался Брюсом Шнайером в [2]. Метод последовательной суффиксной суперпозиции действительно увеличивает порядок хеш-функции, но не увеличивает ее криптостойкости. Проиллюстрируем сделанное замечание на примере.

Как следует из таблицы 5, метод последовательной суффиксной суперпозиции здесь не работает. Дело здесь в том, что значения внутренних регистров стабилизируются.

Метод последовательной преффиксной суперпозиции

Суть данного метода, как и метода последовательной суффиксной суперпозиции, заключается в следующем. Пусть $M \in \{0,1\}^*$ —

M_1	4d7a9c83 56cb927a b9d5a578 57a7a5ee de748a3c dcc366b3 b683a020 3b2a5d9f c69d71b3 f9e99198 d79f805e a63bb2e8 45dd8e31 97e31fe5 2794bf08 b9e8c3e9
MD4(M_1)	**d8024c54 82a68fec a61bb37e 35a75377**
M_2	4d7a9c83 d6cb927a 29d5a578 57a7a5ee de748a3c dcc366b3 b683a020 3b2a5d9f c69d71b3 f9e99198 d79f805e a63bb2e8 45dc8e31 97e31fe5 2794bf08 b9e8c3e9
MD4(M_2)	**d8024c54 82a68fec a61bb37e 35a75377**
MD4($M_1 | $MD4($M_1$))	**4fb7eb59 7b1020d3 d4429ec7 a18be02e**
MD4($M_2 | $MD4($M_2$))	**4fb7eb59 7b1020d3 d4429ec7 a18be02e**

Таблица 5.

произвольный текст. Тогда получим хеш-значение данного текста согласно следующему правилу:

$$\overline{h}(M) = \cdots h(h(h(M)\|M)\|M)\|M.$$

Предложенный метод последовательной префиксной суперпозиции дает положительный эффект. Действительно, данный метод позволяет успешно противостоять атаке, основанной на туннельном эффекте. Дело в том, что пара текстов M, $M + \Delta C$ построенная согласно туннельному эффекту ΔC, по построению имеет $h(M) = h(M + \Delta C)$. Однако пара текстов $h(M)\|M$, $h(M + \Delta C)\|M$ отличаются уже на $\Delta D \neq \Delta C$. Следовательно, во вновь полученном тексте вид туннельного эффекта не сохранится. Используя M_1, M_2 как в таблице 5, получим:

MD4(MD4(M_1)$\|M_1$)	**14bb2693 ebd1cbd9 28c04dc4 13652941**
MD4(MD4(M_2)$\|M_2$)	**08372524 65baf1ea 841e560c fed946c4**

Таблица 6.

Продолжая последовательно процесс префиксных суперпозиций, полученные новые хеш-значения будут сильно отличатся, в соответствии с присущим хеш-функции $h(\cdot)$ лавинообразным эффектом. Данный метод имеет недостаток лишь в том, что скорость выработки хеш-значения будет снижаться в пропорциональное число раз. Архитектура метода последовательной префиксной суперпозиции не подразумевает распараллеливание, что делает схему менее привлекательной. Однако, за счет увеличения порядка хеш-функции криптостойкость существенно повышается.

Метод конкатенации

Суть метода конкатенации заключается в конкатенации нескольких хеш-значений в одно: $h(M) = h_1(M)\|h_2(M)\|\ldots\|h_k(M)$. При этом в роли h_i участвуют различные хеш-функции, например можно построить хеш-функцию следующим образом: $h(M) = \mathrm{MD4}(M)\|$ $\mathrm{MD5}(M)\|\mathrm{SHA1}(M)\|\mathrm{RIPEMD128}(M)\|\mathrm{HAVAL128}(M)$. Данный метод не увеличивает криптостойкость хеш-функции, действительно, порядок вышеупомянутой хеш-функции будет равен 128. Поэтому использование метода конкатенации сомнительно. Отметим, что данный метод все же можно использовать для улучшения криптологических свойств хеш-функций.

Метод перестановок

Для повышения криптостойкости хеш-функций лучше всего подходит метод перестановок. Хеш-функция из этого метода строиться согласно правилу:

$$\overline{h}(M) = h(M)\|h(\pi_1(M))\|\ldots\|h(\pi_k(M)).$$

Здесь π_i, $i = 1, \ldots, k$ — произвольные перестановки текста M Например можно предложить использовать каскадную схему построения метода перестановок. Первая перестановка получена перестановкой двух половинок текста, вторая — перестановкой четвертинок текста, третья — одной восьмой текста и т. д. Полученное хеш-значения, будет являться хеш-функцией, как конкатенация хеш-значений однонаправленной хеш-функции. Данный метод эффективно увеличивает криптостойкость, за счет серьезного увеличения хеш-значения.

Также метод перестановок эффективен против туннельного эффекта, дело в том, что после перестановки полученный текст отличается от исходного не на туннельный эффект ΔC, что приводит к отличию хеш-значений данных текстов. Данный метод может быть эффективно распараллелен, что является серьезным преимуществом для использования его на маломощных микропроцессорах. Если в задаче скорость не является серьезным аргументом, то можно использовать усиление метода перестановок следующего вида:

$$\overline{h}(M) = \ldots h(\pi_2(h(\pi_1(h(M)\|M))\|M))\ldots$$

Если нет возможности изменить алгоритм вычисления хеш-значения, то последняя схема усиления является самой эффективной.

Список литературы

[1] Handbook of Applied Cryptography / A. Menezes, P. Oorschot, S. Vanstone (eds.). CRC Press, 1996.

[2] Шнайер Б. Прикладная криптография / 2-ое изд. 2002.

[3] Wang X., Feng D., Lai X., Yu H. Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD. 2004.