Методы построения хэш-функций. Их анализ.

Выполнила Суслова Вера,
группа 019
курс: Защита информации
2004 год

Методы построения хэш-функций. Их анализ.

1.Что такое хэш-функция.

Криптографические методы не только позволяют обеспечивать конфиденциальность, но и контролировать целостность передаваемых данных. Контроль целостности, как правило, осуществляется методом вычисления контрольной суммы данных. В настоящее разработано большое число алгоритмов, вычисляющих контрольные суммы передаваемых данных. Дело в том, что в большинстве случаев достаточно простой контрольной суммы, например, в тех случаях, когда объем информации не известен заранее или имеет значение скорость обработки.

Но при использовании простой контрольной суммы можно подобрать несколько массивов данных, у которых будет получаться одинаковая контрольная сумма. Криптографически стойкие контрольные суммы получаются в случае применения к исходному тексту хэш-функции.

Хэш-функция — это процедура обработки сообщения, в результате действия которой формируется строка символов фиксированного размера (дайджест сообщения). Изменения в тексте сообщения приводят к изменению дайджеста, получаемого после применения хэш-функции. Поэтому любые искажения, внесенные в текст сообщения, отразятся в дайджесте.

Алгоритм применения хэш-функции заключается в следующем:

- перед отправлением сообщение обрабатывается при помощи хэш-функции. В результате получается его сжатый вариант (дайджест). Само сообщение при этом не изменяется и для передачи по каналам связи нуждается в шифровании;
- полученный дайджест шифруется закрытым ключом отправителя и пересылается получателю вместе с сообщением;
- получатель расшифровывает дайджест сообщения открытым ключом отправителя;
- получатель обрабатывает сообщение той же хэш-функцией, что и отправитель и в результате получает его дайджест. Если дайджест, присланный отправителем, и дайджест, полученный в результате обработки сообщения получателем, совпадают, это означает, что в сообщение не было внесено изменений.

Коллизии. Коллизией хэш-функции H(X) называется ситуация, когда существуют два различных текста X1 и X2, таких что H(X1) = H(X2).

Основное требование, предъявляемое криптографическими приложениями к хэш-функциям, состоит в отсутствии эффективных алгоритмов поиска коллизий.

Гипотеза о существовании односторонних функций. Под односторонней функцией подразумевают функцию, определенную на множестве натуральных чисел и не требующую больших вычислительных ресурсов для вычисления своего значения. В то же время вычисление обратной функции, (когда нужно по известному значению функции восстановить значение аргумента) оказывается невозможно теоретически (либо очень сложно для вычисления).

Существование односторонних функций пока не доказано, поэтому все используемые в настоящее время хэш-функции являются лишь "кандидатами" в односторонние функции, хотя и имеют достаточно хорошие свойства.

2. Методы построения криптографических хэш-функций и вопросы эффективности.

В настоящее время используют три основных метода для построения хэш-функций:

1. построение хэш-функции на основе существующего алгоритма шифрования (например, на основе шифров ГОСТ 28147 - 89, DES и FEAL.),
2. построение хэш-функции на основе известной математической задачи, как правило вычислительно трудной (например, функция из рекомендаций MKKTT X.509);
3. построение хэш-функции "с нуля".

При разработке криптографически стойких хэш-функций рекомендуется руководствоваться следующими принципами (сформулированными Райвестом (Rivest)):

- все возможные методы для построения коллизий должны быть менее эффективными, чем метод, с использованием "парадокса дня рождения";
- алгоритм должен быть реализован на 32-разрядном процессоре;
- по возможности избегать использования сложных структур данных;
- алгоритм должен быть оптимизирован с точки зрения его реализации на микропроцессорах типа Intel.

Как правило, хэш-функция последовательно применяются к блоку входного текста, при этом используется хеш-функция, полученная на предыдущем шаге. Поэтому хэш-кодом всего сообщения является код, получаемый после хэширования всего текста: $H_i = h(H_{i-1}, M_i)$, то есть для эффективности и надежности хэш-функции, элементарная хэш-функция тоже должна удовлетворять вышеперечисленным.

Наиболее стойкие хэш-функции получаются при построении “с нуля”. В этом случае рекомендуется использовать известные методы и принципы построения надежных шифраторов.

2.1 N-хэш.

Алгоритм был разработан фирмой Ниппон телефон энд телеграф (Nippon Telephone & Telegraph) в 1990.

На вход подаются блоки длиной 128 битов, к которым применяется так называемая “размешивающая ” функция (аналогичная функции алгоритма шифрования FEAL), в результате на выходе получается хэш-код в128 бит.

На каждом шаге алгоритма в качестве аргумента обрабатывается блок сообщения длиной 128 бит и хэш-код предыдущего шага. Хэш-кодом всего сообщения является хэш-код, полученный после обработки целого текста.

Принцип работы “размешивающей” функции: старшая и младшая части хэш-кода, полученного от предыдущего блока, меняются местами, затем складываются с величиной 1010…1010 (длиной 128 бит) и блоком хэшируемого текста.

Хэш-код от блоку текста поступает на вход каскада из N (N=8) преобразующих функций. В качестве второго аргумента преобразующие функции используют хэш-код предыдущего шага, сложенный покоординатно с одной из восьми констант.

Идея использовать алгоритм блочного шифрования, с известной стойкостью, как схемы хэширования, на первый взгляд кажется неплохой. Однако в некоторых случаях это может повлечь за собой ряд проблем: необходимость задания некоторого ключа, с помощью которого происходит шифрование. Этот ключ необходимо сохранять в секрете, иначе зная ключ и окончательных хэш-код, можно выполнить процедуру в обратном направлении.

Еще один недостаток состоит в том, что размеры хэш-кода и блока в алгоритме совпадают. В общем случае размер блока достаточно мал, что делает схему неустойчивой к атаке на основе "парадокса дня рождения". Сейчас используются алгоритмы хэширования с размером хэш-кода в m раз (часто, m=2) большим, чем размер блока алгоритма шифрования.

2.2 MD4

Алгоритм MD4 (Message Digest) был разработан Райвестом (Rivest). При создании алгоритма разработчик стремился достичь следующих целей:

- Безопасность. Для построения коллизий не существует алгоритма эффективнее метода "грубой силы" (т. е. метода, основанного на "парадоксе дня рождения").
- Стойкость алгоритма должна, подобно шифраторам, обеспечиваться его собственной конструкцией (т.е. алгоритм построен без использования каких-либо предположительно трудных задач).
- Скорость. Существует эффективная программная реализация на 32-разрядном процессоре.
- Простота и компактность. Алгоритм MD4 не использует сложных структур данных и подпрограмм.
- Алгоритм оптимизирован для его реализации на микропроцессорах типа Intel.

Вскоре после того, как алгоритм стал известен, были найдены коллизии для 2 из 3 раундов этого алгоритма. Хотя ни один из предложенных методов построения коллизий не приводит к успеху для полного MD4, Райвест усилил алгоритм и предложил новую схему хэширования MD5.

2.3 MD5

Отличия MD5 от MD4:

- в основной цикл был добавлен еще один дополнительный раунд (четвертый);
- в каждом операторе при суммировании используется уникальная константа;
- изменена функция, использованная во втором раунде с целью сделать ее менее симметричной.
- к промежуточным результатам каждого шага прибавляются результаты предыдущих шагов, что позволяет усилить эффект распространения ошибки;
- изменен порядок считывания слов во втором и третьем раундах;
- величина циклического сдвига в каждом раунде оптимизирована с точки зрения усиления эффекта распространения ошибки. Сдвиги от раунда к раунду различаются.

Входной текст делится на блоки длиной 512 бит, каждый из этих блоков делится затем на 16 частей по 32 бита, конкатенация которых образует 128-битовый хэш-код.

Сначала текст дополняется таким образом, чтобы длина получаемого текста (в битах) была на 64 меньше числа, кратного 512. Это осуществляется путем добавления к концу текста 1 и нужного числа нулей. Затем к тексту приписывается 64-битовое представление длины исходного сообщения. Таким образом, получается текст, длина которого кратна 512.

Инициализируются 4 переменных по 32 бита:

A = 01 23 45 67;

B = 89 AB CD EF;

C = FE DC BA 98;

B = 76 54 32 10.

Основной цикл алгоритма. Создаются копии начальных переменных: AA для A, BB для B, CC для C, DD для D.

В каждом цикле, состоящем из 4 раундов действуют 16 операторов. Все операторы вычисляются как : $u=v+((F(v,w,z)+M_i+t_i<<s_j$.

Здесь u, v, w, z - это A, B, C и D в различных раундах.

M_j обозначает j-тый подблок обрабатываемого блока. В каждом раунде порядок обработки очередным оператором подблоков определяется задаваемой в явном виде подстановкой на множестве всех подблоков (их, также как и операторов, 16).

t_i - случайные константы.

В конце основного цикла суммируются переменные A, B, C и D с текущими AA, BB, CC и DD. Это и является выходом алгоритма, после обработки последнего блокаю

2.4 SECURE HASH ALGORITHM (SHA)

SHA разработан в 1993 году Национальным институтом стандартов и технологий (NIST) США совместно с Агентством национальной безопасности США. Разработчики считают, что для SHA невозможно найти алгоритм, не требующий больших вычислительных ресурсов, который позволил бы найти коллизии.

В результате применения алгоритма получается хэш-код длиной 160 бит. Процедура дополнения хэшируемого текста до кратного 512 битам аналогична подобной процедуре алгоритма MD5.

Назначаются 5 переменных по 32 бита (в алгоритме MD5 таких переменных было 4):

A = 67 45 23 01;

B = EF CD AB 89;

C = 98 BA DC FE;

D= 10 32 54 76;

У = C3 D2 E1 F0.

Также как и в Md5 создаются копии этих переменных AA, BB, CC, DD и EE. Основный цикл данного алгоритма состоит из 4 раундов, каждый из которых состоит из 20 операторов.

SHA скорее является модификацией MD4, нежели новой разработкой. Сравним изменения MD4, сделанные при разработке MD5, и изменения, сделанные в MD4 при разработке SHA.

1. В каждом из алгоритмов используется 4 раунд. В SHA функция, используемая в 4 раунде аналогично функции из 2 раунда.

2. В SHA используется тот же принцип аддитивной константы для каждого оператора в раунде, что и в MD4. (в MD5 константа каждого оператора отличается от других).

3. Во втором раунде SHA используется та же функция что и в MD4, тогда как в MD5 она была изменена, чтобы сделать ее менее симметричной.

4. Для усиления эффекта распространения ошибки в SHA добавлена 5-я переменная по которой идет "зацепление", в результате этого метод ден Бура -- Босселэра, эффективный для MD5, перестает работать в случае SHA.

5. Если в MD5 изменен порядок считывания слов во втором и третьем раундах, то в SHA входные слова считываются по правилу, созданному на основе кода, исправляющего ошибки.

6. В MD5 величина циклического сдвига меняется от раунда к раунду и от оператора к оператору, а в SHA используют простые циклические сдвиги

2.5 MDC

Разработано фирмой АйБиЭм (IBM). В качестве схемы хэширования использует блочный шифр (в оригинале DES) для получения хэш-кода, длина которого в два раза больше длины блока шифра. Существуют два варианта этой разработки: MDC2 и MDC4. Первая работает несколько быстрее, но является менее надежной, чем вторая.

Схема вычисления функции MDC4.

$H1_0 = I_1$, где I_1 -- начальное значение,

$H2_0 = I_2$, где I_2 -- еще одно начальное значение;

$X1_i \,||\, X2_i = E\,\mathrm{mod}1(H1_{i-1})\,(M1_i) \oplus M1_i$;

$X3_i \,||\, X4_i = E\,\mathrm{mod}2(H1_{i-1})\,(M2_i) \oplus M2_i$;

$Y1_i \,||\, Y2_i = E\,\mathrm{mod}1(X1_i \,||\, X4_i)\,(H2_{i-1}) \oplus H2_{i-1}$;

$Y3_i \,||\, Y4_i = E\,\mathrm{mod}2(X3_i \,||\, X2_i)\,(H1_{i-1}) \oplus H1_{i-1}$;

$H1_i = Y1_i \,||\, Y4_i$;

$H2_i = Y3_i \,||\, Y2_i$;

Хэш-кодом является объединение последних значений H1и H2 . Здесь обозначено: M1 и M2 -- соответственно левая и правая части i-того блока хэшируемого текста. Функция mod1 просто заменяет значения 1-го и 2-го битов аргумента на значения 1 и 0 соответственно, а mod2 -- на значения 0 и 1.

Хэш-функция MDC4 использована в проекте RIPE, хотя эффективность (скорость хэширования при программной реализации) у данной схемы низкая.

Литература.

1. http://www.jetinfo.ru/2001/3/2/article2.3.2001126.html
2. http://www.perspektiva.org/ecommerce/Glava3/Index61.html
3. http://www.cryptography.ru/db/msg.html?mid=1161287&uri=node55.html