Методы конструирования хеш-функций.

Выполнил:
студент 116 группы
Королёв Сергей
2005 год

Методы конструирования хеш-функций.

1.Введение.

Современная криптография является наукой, связанной с решением таких проблем безопасности информации, как конфиденциальность, целостность, аутентификация и невозможность отказа сторон от авторства. В нашей работе мы подробно остановимся на обеспечение *целостности* передаваемой информации. Под этим термином понимается обеспечение невозможности любого несанкционированного изменения информации. Для обеспечения целостности необходим критерий обнаружения любых манипуляций с данными, то есть вставок, удаления или замены.

Как правило, решение этой задачи имеет целью предотвратить целенаправленное навязывание злоумышленником ложной информации. Для этого в передаваемую информацию вносится некоторая избыточность, которая представлена в виде некоторой контрольной суммы, то есть проверочной комбинации, которая вычисляется с помощью специального алгоритма. Такой алгоритм зависит от секретного ключа, поэтому навязывание неверной информации очень маловероятно и называется такая вероятность мерой *имитостойкости* шифра.

Как правило, контрольной суммой является значение некоторой *хеш-функции* от данного сообщения h(M)=S.

Хэш-функция – функция, аргументом которой является сообщение, выходным значением - строка символов фиксированного размера (дайджест сообщения). Изменения в тексте сообщения приводят к изменению значения хеш-функции. Поэтому все изменения, в тексте сообщения приведут к изменению дайджеста.

Множество видов хеш-функций можно разбить на две группы – ключевые и бесключевые.

Первые называются кодами аутентификации сообщений (message authentication code(MAC)) и применяются в системах с симметричными ключами. К ним предъявляются следующие требования:

- *простота вычисления* (для известной функции H_k, заданного значения k и входного значения x, легко вычислить $H_k(x)$);
- *сжатие* (H_k отображает входное значение x – конечную двоичную строку произвольной длины в выводное значение $H_k(x)$ – двоичную строку фиксированной длины n).
- *стойкость к вычислению* (по известным парам "сообщение – значение функции" вычислительно невозможно получить любую другую пару "сообщение – значение функции", для любого входного значения отличного от имеющихся). Свойство также включает не восстанавливаемость ключа.

Вторые известны как коды обнаружения ошибок (modification detection code(MDC)) и дают возможность с помощью дополнительных средств гарантировать целостность данных. На бесключевые хеш-функции накладываются следующие условия:

- *однонаправленность* (высокая сложность нахождения сообщения с заданным значением дайджеста);
- *устойчивость к коллизиям* (высокая сложность нахождения пары сообщений с одинаковыми значениями дайджеста);
- *устойчивость к нахождению второго прообраза* (высокая сложность нахождения второго сообщения с тем же значением дайджеста для заданного сообщения с известным значением дайджеста).

Ключевые хеш-функции применяются в случаях, когда стороны имеют общий секретный ключ (доверяют друг другу). В подобных ситуациях обычно не требуется обеспечение защиты в случае отказа получателя от факта получения сообщения или его подмены. Поэтому от ключевых хеш-функций не требуется устойчивости к коллизиям. Заметим, что для выполнения требований вычислительной устойчивости (невозможности модификации и фабрикации) вытекает невозможность определения секретного ключа. Для вычисления дайджеста ключевых хеш-функций могут использоваться блочные шифры, что неприемлемо по отношению к бесключевым хеш-функциям, так как обратимость блочного шифрования позволяет подбирать входное сообщение по значению дайджеста при фиксированном и общеизвестном ключе, что противоречит требованию устойчивости к нахождению второго прообраза.

Следует отметить, что ключевые хеш-функции могут строиться на основе бесключевых. При этом ключ приписывается к обрабатываемому сообщению, но не просто в начало его или конец, что приводит к потенциальным слабостям. Используются способы введения ключа, при которых он вставляется не один, а по крайней мере два раза. Укажем два способа: $H=h(k, y, M, k)$ и $H=h(k, y_1, h(k, y_2, M))$, где y, y1 и y2 – дополнения ключа k до размера, кратного длине блока n. Недостатком такого метода является слишком большая длина дайджеста.

2. Основные методы построения хэш-функций.

Все используемые в настоящее время хэш-функции работают как последовательное применение к выходу предыдущей итерации и очередному блоку обрабатываемого текста некоторой фиксированной функции, поэтому значением хеш-функции от всего сообщения будет выход, получаемый после последнего шага: $H_i=h(H_{i-1}, M_i)$.

Рассмотрим некоторые алгоритмы построения хеш-функций, и по возможности их сходства и различия.

2.1 MD4

Хеш-функция MD4 (Message Digest) была разработана Райвестом (Rivest). Для входного сообщения этот алгоритм хеширования выдаёт 128-битовый дайджест. При разработки алгоритма Райвест пытался достичь следующих результатов:

- *безопасность*. Малая вероятность коллизий. Нахождение коллизий должно быть вычислительно невозможным. Безопасность не должна основываться на каких-либо допущениях;
- *быстродействие.* Алгоритм MD4 должен подходить для высокоскоростных программных реализаций и состоять из набора простых битовых операций с 32-битовыми операндами;
- *простота и компактность*. Алгоритм должен быть предельно прост, и не содержать сложных структур данных и программных модулей;
- *унифицированность архитектуры*. Алгоритм должен легко адаптироваться для различной микропроцессорной архитектуры.

После выхода алгоритма MD4 исследователи Б. ден Бур и А. Босселер провели успешный криптоанализ последних двух из трёх этапов этого алгоритма, причём в это же время Р. Меркль совершенно независимо вывел метод компрометации первых двух раундов алгоритма. Было также рассмотрено использование дифференциального криптоанализа для первых двух этапов. И хотя все эти попытки приводят к успеху только в случае алгоритмов с неполным числом раундов, Райвест модернизировал его. В результате чего и появился алгоритм хеширования MD5.

2.2 MD5

Алгоритм MD5 – улучшенная версия MD4, причём схемы их похожи и в результате работы обоих получаются 128-битовые дайджесты.

Алгоритм MD5 обрабатывает входной текст 512-битовыми блоками, поделёнными на шестнадцать 32-битовых блоков, которые затем формируются в 128-битное хеш-значение.

Первым делом текст сообщения дополняется таким образом, чтобы его длина была на 64 короче длины числа, кратного 512. Прибавляется единица и необходимое количество нулей до конца сообщения. Эти 64 бита затем заполняются 64-битовым представлением длины текста, это приводит длину сообщения к равенству числу, длина которого кратна 512 бит. После такой операции появляется уверенность, что разные сообщения будут различаться после обработки.

Дальше создаются 4 переменных по 32 бита:

A=0x01234567

B=0x89abcdef

C=0xfedcba98

D=0x76543210

Опишем основной цикл алгоритма, который последовательно применяется , пока не закончатся 512-битовые блоки текста.

Создаются копии инициализированных переменных: **AA** для **A**, **BB** для **B**, **CC** для **C**, **DD** для **D**.

Каждый основной цикл состоит из 4 раундов и 16 операторов. Все операторы вычисляются как : $u = v + ((F(v, w, z) + M_j + t_j) << sj)$.

Здесь: **u**, **v**, **w** и **z** это **A**, **B**, **C**и **D** в зависимости от номера раунда и номера оператора. $\mathbf{M_j}$ является **j**-тым подблоком 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).

$\mathbf{t_j}$- случайные константы. F(v, w, z) – некоторая функция, которая фиксирована для каждого раунда, и действует покоординатно на биты своих трёх аргументов.

Основной цикл алгоритма заканчивается суммированием полученных **A**, **B**, **C** и **D** и накапливаемых **AA**, **BB**, **CC** и **DD**, после чего алгоритм переходит к обработке нового блока. Выход алгоритма - склейка полученных после последнего цикла **A**, **B**, **C** и **D**.

По сравнению с MD4 алгоритм MD5 имеет следующие улучшения:

- добавлен четвёртый этап;
- теперь, в каждом действие используется уникальная константа;
- результат каждой операции теперь суммируется с результатом предыдущего этапа, что приводит к достижению более быстрого лавинного эффекта;
- изменение порядка, в котором используются подблоки сообщения на раундах 2 и 3 увеличивает несходство шаблонов;
- для ускорения лавинного эффекта приближенно оптимизированы значения циклического сдвига влево на каждом этапе.

Попытка дифференциального криптоанализа к одному этапу MD5 оказалась безуспешной, однако атака ден Бура и Босселера, которая использовала функцию сжатия, привела к обнаружению коллизий в алгоритме MD5.

2.3 SECURE HASH ALGORITHM (SHA-1)

Алгоритм хеширования SHA-1 описан в стандарте США безопасного хеширования. Для входного сообщения, длина которого меньше 2^{64} бит алгоритм SHA-1 выдаёт 160-битовый результат. Предназначен SHA-1 для использования вместе с алгоритмом цифровой подписи DSA. Цифровая подпись формируется на основе дайджеста SHA-1 от сообщения, что повышает эффективность процесса подписания.

В результате применения алгоритма получается хэш-код длиной 160 бит. Процедура дополнения хэшируемого текста до кратного 512 битам совпадает с процедурой дополнения алгоритма MD5.

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

A=67452301
B=efcdab89

C=98badcfe
D=10325476
E=c3d2e1f0

Далее, как и в MD5, создаются копии **AA**, **BB**, **CC**, **DD**, **EE** инициализированных переменных и для каждого блока текста размером 512 битов выполняется основной цикл, состоящий из 4 раундов. В отличие от MD5 каждый раунд состоит из 20 операторов (в MD5 - 16 операторов).

Аналогично MD5, каждый оператор состоит из функции от 3 переменных (в случае SHA-1 это **B**, **C** и **D**), циклического сдвига и суммирования.

На сайте [6] упоминаются три китайских криптоаналитика, которые разработали метод поиска коллизий алгоритма SHA-1, который эффективнее метода “грубой силы”. Они научились находить коллизии за 2^{69} вычисления, что на пределе возможности современных вычислительных технологий.

2.4 Отечественный стандарт хеш-функции (ГОСТ Р 34.11-94)

Российский стандарт хеширования – ГОСТ Р 34.11-94 использует блочный алгоритм шифрования ГОСТ 28147-89. Хэш-функция формирует 256-битовый дайджест. Предназначена для использования совместно с российским стандартом электронной цифровой подписи.

Приведём краткое описание функции сжатия алгоритма ГОСТ Р 34.11-94 **f**. H_i=**f**(H_{i-1}, M_i)

- Смешивает линейно H_{i-1}, M_i и некоторых констант C_j, генерируются четыре ключа шифрования K_j, j = 1…4;
- Входной блок разбивается на четыре части равной длины, каждая из которых потом шифруется на одном из четырёх ранее полученных ключей, склейка результата этой операции затем заносится во временную переменную S;
- H_i представляет собой значение линейной функции от S, M_i и H_{i-1}.

Окончательное значение преобразования есть функция от следующих значений: H_n – значения последнего обработанного блока, Z – величины, которая является результатом сложения по модулю 2 всех блоков сообщения, L – длины сообщения и последнего блока сообщения M`. Функция выглядит следующим образом: H = f ( Z ⊕ M`, f (L, f (M`, H_n))) .

2.5 Сравнение хеш-функций ГОСТ Р 34.11-94 и SHA-1

Для начала несколько слов о задаче поиска коллизий для криптографических хеш-функций. Рассмотрим отображение множество обрабатываемых текстов **B** на множество дайджестов **H**. Одному и тому же дайджесту может быть сопоставлено несколько исходных текстов, поэтому на множестве обрабатываемых текстов вводят такое понятие, как классы эквивалентности по значениям функции, в нахождении которых и заключается задача построения коллизий.

Задача оценки числа коллизий тесно связана с задачей о “парадоксе близнецов”. Кратко о ней напомним: имеется множество некоторых элементов, мощность которого **n**, и

множество признаков мощностью **m**. Получим, что среднее число одинаковых признаков для какого-то одного элемента равно n/m, отсюда среднее число пар элементов с одинаковым признаком равно $n/\sqrt{m}$ **.** Применив эти результаты к задаче о коллизиях в соответствующих случаях, найдём |B|/|H| и $|B|/\sqrt{|H|}$. В криптографии за меру сложности нахождения коллизий принимается величина $\sqrt{|H|}$.

Для построения методов отыскания коллизий, существенны следующие показатели:

- Аналитическая сложность отображения, реализуемого при получении дайджеста;
- Способ организации поблочного шифрования текста;
- Способ дополнения последнего блока.

SHA-1 обладает рядом особенностей, которые приводят к снижению его криптографической стойкости:

- Преобразования не обладает хорошими перемешивающими свойствами, из-за операций, которые в них выполняются – это операции сдвига, побитовые логические операции, операции сложения по модулю 2^{32};
- 2^{352} варианта обрабатываемого блока приводит к одному и тому же выходному значению, так как обрабатывается блок длиной 512 бит, а получается значение длиной 160 бит;
- также дополнительную уязвимость привносит метод дополнения последнего блока до длины 512 бит.

Всё вышеперечисленное справедливо также и для таких алгоритмов хеширования, как MD4, MD5, SHA – 256, SHA – 512 аналогичных в плане построения. Это позволяет применять для поиска коллизий метод дифференциального криптоанализа.

Совершенно по-другому обстоят дела со стандартом хеширования ГОСТ Р 34.11-94. Ниже приведён ряд его отличительных особенностей:

- при обработке блоков используются преобразования по алгоритму ГОСТ-89;
- обрабатывается блок длиной 256 бит, и выходное значение тоже имеет длину 256 бит.
- также применены меры борьбы против поиска коллизий, основанном на неполноте последнего блока.

Отметим также, что обработка блоков происходит по алгоритму шифрования ГОСТ - 89, который содержит преобразования на S – боксах, что существенно осложняет применение метода дифференциального криптоанализа к поиску коллизий алгоритма ГОСТ Р 34.11-94. Это существенный плюс по сравнению с SHA–1.

Сравним теоретическую стойкость двух алгоритмов. Для SHA-1 с его множеством значений 2^{160} этот показатель будет равен 2^{80}. Алгоритм ГОСТ Р 34.11-94 имеет теоретическую стойкость 2^{128}, что во много раз превосходит 2^{80}.

Но всё же есть показатель, по которому SHA-1 превосходит ГОСТ Р 34.11-94. Это скорость обработки данных. Для SHA-1 – $48.462*2^{20}$ байт/с, ГОСТ Р 34.11-94 – $9.977*2^{20}$ байт/с.

В заключении остаётся добавить, до сих пор не найдено эффективных методов поиска коллизий для алгоритма ГОСТ Р 34.11-94.

3. Литература.

1. Алфёров А.П., Зубов А.Ю., Кузьмин А.С., Черёмушкин А.В. “Основы криптографии: учебное пособие”.
2. Соколов А.В., Шаньгин В.Ф. “Защита информации в распределённых корпоративных сетях и системах”.
3. http://www.cryptography.ru/db/msg.html?mid=1169307&uri=node54.html
4. http://www.cryptopro.ru/CryptoPro/doc/SHA-1_collision.pdf
5. http://www.globaltrust.ru/security/knowbase/Integrity/hash_alg.htm
6. http://www.schneier.com/blog/archives/2005/02/cryptanalysis_o.html
7. http://www.security.ukrnet.net/modules/sections/index.php?op=printpage&artid=173