УДК 519.713

ФУНКЦИИ ХЕШИРОВАНИЯ. ПОНЯТИЯ, ТРЕБОВАНИЯ, КЛАССИФИКАЦИЯ, СВОЙСТВА И ПРИМЕНЕНИЕ

ГОРБЕНКО И.Д., ШТАНЬКО И.А.

Рассмотрены алгоритмы хеширования, применяющиеся в современной криптозащите для решения таких наиболее важных проблем, как обеспечение целостности и причастности на всех этапах электронного документооборота, а также вопросы использования различных хеш-функций в системах обработки информации. Произведен сравнительный анализ подходов к реализации алгоритмов хеширования.

В современных информационных системах на всех этапах обращения информации (данных и программного обеспечения) должны реализовываться такие основные функции как целостность и причастность. При этом функция причастности обеспечивает контроль подлинности данных и программного обеспечения. Целостность данных (программного обеспечения) представляет собой способность данных сохраняться для использования по назначению. Причастность (неотказуемость) обеспечивает неотказуемость с подтверждением подлинности источника и самих данных (программного обеспечения), а также неотказуемость с подтверждением доставки. Названные функции обеспечиваются в результате применения механизмов аутентификации, включая и электронную цифровую подпись, причем цифровые подписи вырабатываются на основе использования хеш-функций. В некоторых случаях в качестве кода аутентификации или цифровой подписи могут использоваться и непосредственно хеш-функции. Поэтому весьма важно подробно их рассмотреть и проанализировать, включая хеш-функций, включая обсуждение требований, классификацию хеш-функций, анализ свойств и различных приложений.

1. Понятие и требования к хеш-функциям

Для информации M (данных, программного обеспечения) длины l_M бит определим хеш-функцию [1]

$$h=H(M) \qquad (1)$$

однозначного отображения M длины l_M бит в сжатый образ h длины l_h. Если h определятся с использованием начального вектора V_H и /или ключа K_I, то

$$h=H(V_H, M, K_I). \qquad (2)$$

Функции вида (1) и (2) получили название хеш-функций от английского слова "hash", что означает перемешивание. В ряде источников функции хеширования имеют другие названия – функция сжатия, криптографическая контрольная сумма, функция контроля целостности сообщения и др.

В дальнейшем под функцией хеширования, как правило, будем понимать функцию преобразования по законам (1) или (2), причем в некоторых случаях функция хеширования может быть и разжимающей, например, если $l_M<l_h$. Но во всех случаях хеш-функцию называют однонаправленной, понимая под этим однозначное отображение M в h (сжимающее или разжимающее) и получение h фиксированной или требуемой длины l_h.

К хеш-функциям предъявляются следующие требования:

1. Для каждого M несложно вычислить h, в том числе и в реальном масштабе времени.

2. По известному h сложно вычислить M, такое что h=H(M).

3. По известному M вычислительно сложно или практически невозможно сформировать (вычислить) M', такое что

$$h=H(M)=H(M'). \qquad (3)$$

4. Вычислительно сложно или практически невозможно найти случайные M и M', такие что

$$H(M)=H(M'). \qquad (4)$$

В случае 4 полагается, что M и M' порождаются путем изменения двух исходных сообщений, например добавлением незначащих пробелов. При выполнении только требований (1)-(3) существует возможность подбора двух сообщений, имеющих одинаковое значение хеширования, т.е. коллизии. Хотя требования (1)-(3) обеспечивают однонаправленность хеш-функции, в большинстве приложений необходимо выполнить и требование 4, реализовав защиту от коллизий.

Функция хеширования вида (2) может зависеть от сообщения (данных) M и начального вектора состояния V_H:

$$h=H(V_H, M). \qquad (5)$$

Для функции вида (5) возможно существование псевдо-коллизий, т.е. для различных начальных значений V_H и V_H' и возможно идентичных входных значений M и M' справедливо соответствие

$$H(VH, M)=H(V_H', M'). \qquad (6)$$

Специалисты считают [2], что на практике псевдо-коллизии имеют значительно меньшую важность, чем коллизии.

При разработке хеш-функции особое внимание должно быть уделено обоснованию и выбору длины хеш-функции l_H. При этом следует учитывать существование атак, получивших название "birthday attack" или "метода встречи посредине". Суть этого приема состоит в подготовке двух сообщений: одного, который вы подпишете, и другого, к которому потом "приклеют" вашу подпись. Эта атака потребует всего $2^{lh/2}$ итераций [3], что весьма опасно. Например, для хеш-функции длиной l_h=64 такого рода атака потребует всего 2^{32} итераций. Считается, что длина хеш-функции должна быть не менее 128 бит. В этом случае для случайного поиска двух документов M и M', имеющих одинаковое значение хеширования, необходимо осуществить хеширование $2^{lh/2} \geq 2^{64}$ документов.

С точки зрения возможных последствий описанное нападение является самым "безобидным" из всех возможных, поскольку в распоряжении злоумышленника оказывается лишь один-единственный поддельный документ. Для подделки других подписанных Вами документов ему снова потребуется значительная вычислительная работа. Кроме того, можно противостоять данному нападению, если перед формированием значения хеширования внести в документ некоторые изменения.

В то же время удлинение значения хеширования влечет за собой, с одной стороны, увеличение объема передаваемых данных, а с другой — вычислительной сложности. В последнем случае появляется возможность создания хеш-функции с большой вычислительной сложностью, что позволяет повысить стойкость алгоритма путем увеличения времени выполнения одной итерации. С другой стороны, удлинение хеш-функции увеличивает время ее вычисления, что в ряде случаев нежелательно, особенно в ситуациях, критичных ко времени формирования значения хеш-функции. Поэтому обычно стараются выбрать хеш-функцию с допустимой вычислительной сложностью и приемлемой длиной.

Изложенное позволяет сформулировать еще несколько требований к хеш-функции.

5. Длина l_h хеш-функции должна выбираться из условия устойчивости против нападения, например $l_h \geq\geq l_{доп}$, где $l_{доп}$ – допустимая длина хеш-функции, определенная с учетом прогноза развития средств криптоанализа.

6. Необходима открытость и общедоступность алгоритмов и средств хеширования.

7. Безопасность хеш-функций должна обеспечиваться их однонаправленностью. Это требование заключается в том, что h напрямую и в существенной мере зависит от M. При изменении хотя бы одного бита исходного сообщения значение хеширования должно в среднем измениться на $l_h/2$ битов.

При выполнении приведенных выше требований на основе использования хеш-функций может быть эффективно осуществлен процесс контроля целостности и подлинности информации. Действительно, с использованием хеш-функции для любой информации (данных, программного обеспечения, сообщения, документа) может быть вычислено значение h_l длины l, которое с требуемой вероятностью позволит обнаружить любое изменение информации. Однако это возможно лишь в случае, если информация M и значение h=H(M) хранятся раздельно. Если же при атаке известны $\{M, h_l\}$, существует возможность прямой атаки с незамеченной (необнаруживаемой) модификацией M на M′ и последующим вычислением с использованием общедоступных средств h_l' на $\{M', h_l'\}$. Поэтому h_l должно надежно храниться у пользователя (получателя) отдельно от M. Наиболее эффективно это может быть выполнено посредством защиты (шифрования) значения h_l.

2. Принципы построения и классификация однонаправленных хеш-функций

Алгоритмы хеш-функций строятся по принципу преобразования информации M в соответствии с (1), (2), (5). При этом если $l_M > l_h$, в процессе преобразования производится сжатие прообраза M в образ h(M). Если $l_M < l_h$, в процессе хеширования M растягивается в H(M) до длины l_h.

Таким образом, первым основополагающим принципом вычисления H(M) является преобразование M с использованием соотношений (1), (2) или (5).

Поскольку обычно алгоритмы хеширования обрабатывают входное сообщение по блокам фиксированной длины, они производят добавление некоторой незначащей части к сообщению ("расширение сообщения"), чтобы его длина стала кратной длине обрабатываемого блока. При этом в ряде случаев существует потенциальная возможность получения одних и тех же значений хеш-функции для информации, отличающейся длиной. Это может произойти, если при "расширении сообщения" добавляется незначащая часть (возможно, пробелы, нули…), которая может быть интерпретирована как часть сообщения. Для защиты от этой угрозы к сообщению M может быть добавлено значение длины исходной информации M. Такая технология получила название MD усиления. В результате ее применения хешируемое исходное значение M заменяется преобразованным значением M_n:

$$M_n = \{M, l_M\} = M. \qquad (7)$$

Следовательно, вторым важным принципом является построение хеш-функции в виде

$$h = H(V_H, M) = H(V_H, \{M, l_M\}),$$

как функции от исходной информации M и ее длины l_M одновременно.

Третьим основополагающим принципом, обеспечивающим выполнение предъявляемых к хеш-функции требований, является применение однонаправленной (нелинейной) функции над каждым блоком исходной информации M. В результате такого преобразования при $l_M > l_h$ происходит сжатие M в h, а при $l_M < l_h$ – растяжение M до длины l_h.

Основой функции хеширования является ее ядро, представляющее собой нелинейную функцию, которая последовательно применяется к каждому блоку информации (рисунок). Входами функции H является блок хешируемой информации и значение h_{I-1} хеш-функции для M_{I-1} блока, причем в качестве h_0, как правило, используется начальное значение V_H. Как следует из приведенного, хеш-функцией h информации M является значение h_n хеш-функции последнего блока M_n:

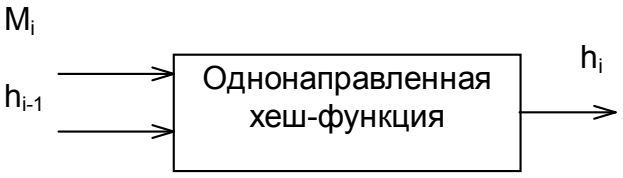
$$h(V_H, M_{n-1}) = f(M_n, h_{n-1}).$$

Все созданные к настоящему моменту функции хеширования базируются на использовании:

1) различных битовых нелинейных функций (MD2, MD4, MD5, SHA,) [4-7];

2) блочных симметричных алгоритмов шифрования (MDC2, MDC4, ГОСТ Р 34.11-94);

3) несимметричных алгоритмов шифрования (RSA, Ель-Гамаля, …) (рисунок) [8].



Структура алгоритма хеширования

Идея использования симметричных блочных алгоритмов шифрования как однонаправленной функции хеширования состоит в том, что если блочный алгоритм безопасен, то и однонаправленная функция хеширования также безопасна. На этом же базируется и применение несимметричных алгоритмов шифрования. Ограничивающим фактором их применения является увеличенная по сравнению с битовыми нелинейными функциями вычислительная сложность и, как следствие, меньшая скорость хеширования.

Естественно, что при выборе той или иной функции хеширования необходимо использовать обоснованные критерии, оценки эффективности и показатели качества. Базируясь на [1], выберем в качестве основных частных показателей устойчивость против криптоаналитичестких атак, понимая под ней вычислительную сложность их реализации, а также скорость хеширования R. В ряде случаев в качестве показателей эффективности функции хеширования используется и объем программного обеспечения, необходимого для его реализации.

3. Алгоритмы, основанные на использовании битовых операций

Алгоритмы хеширования, основанные на использовании битовых нелинейных функций, потенциально имеют большее быстродействие, поскольку они оптимизированы по скорости выполнения и используют аппаратные возможности применяемых процессоров. В них не используются алгоритмы шифрования, при создании которых необходимо удовлетворить ряду условий. Например, при использовании для вычисления значения хеширования алгоритма блочного шифрования должен существовать адекватный алгоритм расшифрования. Для алгоритма хеширования данное требование необязательно, более того, на такой алгоритм, возможно, проще придумать атаку.

Стойкость функций хеширования данного класса сильно зависит от выбранных нелинейных битовых функций, поэтому при создании битовых алгоритмов хеширования данному факту уделяется наибольшее внимание.

Алгоритмы такого класса, как правило, базируются на применении к блоку информации в цикле нелинейных битовых функций, с последующим сдвигом полученного значения на некоторое количество бит. При этом на каждом шаге цикла могут использоваться различные константы алгоритмов нелинейных функций и различные константы сдвига.

Следует отметить, что стойкость таких алгоритмов хеширования сильно зависит от количества используемых циклов. Поэтому наиболее общим подходом модификации алгоритма при нахождении успешной атаки является увеличение количества циклов обработки. При этом повышается сложность алгоритма, а быстродействие, естественно, снижается.

Рассмотрим наиболее известные алгоритмы хеширования этого класса с точки зрения устойчивости к атакам и удовлетворения другим требованиям, предоставляемым к алгоритмам хеширования.

Алгоритмы Snefru и N-Hash [9,10]. Для этих алгоритмов были проведены успешные атаки формирования коллизий.

Первоначально алгоритм Snefru был разработан Merkle как двухпроходный алгоритм. Используя различные методы криптоанализа, Biham и Shamir [11] продемонстрировали небезопасность двухпроходного алгоритма Snefru (для 128 бит). Их атака обнаружила пары сообщений, для которых значение хеширования совпадало в считанные минуты. На данный момент автор алгоритма рекомендует использовать в Snefru как минимум восемь проходов. Однако при таком количестве проходов алгоритм значительно медленнее алгоритмов MD5 или SHA.

Аналогично для N-Hash также была реализована успешная атака. Bert den Boer разработал метод создания коллизий в цикле функции N-Hash. Biham и Shamir [11], используя различные методы криптоанализа, вскрыли 6-й цикловой алгоритм N-Hash. Их атака, которая может быть и другой, работала для любых N, которые делятся на 3, и атака подбора коллизий эффективнее перебора как минимум в 15 раз. Разработчики алгоритма рекомендуют использовать N-Hash с как минимум восемью циклами. Учитывая небезопасность N-Hash и его скорость для восьми циклов, предпочтительным является использование более совершенных алгоритмов хеширования.

Ron Rivest [5] разработал алгоритм хеширования MD4, который изначально был ориентирован на конкретную архитектуру процессора, а именно процессоров фирмы INTEL. Этот алгоритм основывается на простом наборе операций над 32-битными значениями. Алгоритм MD4 прост "как только это возможно, не содержит больших структур данных или вычислительных подпрограмм" [5]. Подход, использованный при создании данного алгоритма, послужил основой для создания алгоритмов MD5 и SHA. После презентации этого алгоритма был проведен успешный криптоанализ отдельно двух первых и двух последних из трех циклов обработки. Но эти атаки не были расширены на весь алгоритм, поэтому его можно считать защищенным против известных атак.

В ответ на успешный криптоанализ части алгоритма Ron Rivest произвел модификацию алгоритма MD4. Полученный алгоритм был назван MD5. В результате модификации в алгоритм был добавлен 4-й цикл обработки, каждый шаг обработки MD5 содержит уникальную константу, изменена одна из используемых нелинейных функций и проведены небольшие изменения для придания большего лавинного эффекта. Несмотря на сделанные изменения, в алгоритме был успешно атакован каждый из циклов обработки в отдельности. Но не удалось реализовать успешную атаку против всех четырех циклов одновременно. Поэтому алгоритм MD5 считают защищенным.

MD2 является еще одним алгоритмом хеширования, который разработал Ron Rivest [4]. Безопасность MD2 зависит от случайной перестановки байт. Перестановки фиксированы и зависят от цифр числа π. Исследователями не было найдено ни одной успешной атаки на MD2, но алгоритм MD2 медленнее, чем большинство известных функций хеширования.

Для стандарта ЦП в США был разработан алгоритм хеширования SHA. Он основан на алгоритме MD4. На данный момент не найдено эффективной атаки на функцию хеширования SHA. Кроме того, поскольку она создает хеш-функцию длиной 160 бит, она более стойка к лобовой атаке (включая атаку создания коллизий), чем 128-битные функции, описанные выше.

Кроме названных, существует еще несколько алгоритмов хеширования, например RIPE-MD и HAVAL. Эти алгоритмы тоже являются модификацией алгоритма MD4. Против них также не было найдено ни одной успешной криптоатаки. Еще одним достоинством алгоритма HAVAL является возможность гене-

рирования значения хеширования различной длины (128, 160, 192, 224 или 256 бит).

Наиболее критичной частью данных алгоритмов являются используемые в них нелинейные функции, т.е. такие, в которых по их значению невозможно получить входное значение методом, отличным от полного перебора. Обычно алгоритмы хеширования данного класса разбивают обрабатываемый блок сообщения на некоторое количество слов, по длине (n-бит) равных используемому слову процессора. Далее над полученными словами в цикле некоторое количество раз выполняются нелинейные функции вида

$$X_1=F(X_1,X_2\ldots,X_m,C_1..C_k,S), \qquad (8)$$

где $X_1\ldots X_m$ — m-, n-битовые слова обрабатываемого блока информации либо n-битные внутренние переменные; $C_1\ldots C_k$ – константы; S – величина сдвига полученного значения.

Например в алгоритме MD4 на первом шаге обработки блока информации выполняется следующее преобразование:

$$A = F(A,B,C,D,X[i],C_1,S), \qquad (9)$$

где функция F имеет вид

$F(A,B,C,D,X[i],S)= (A + f(B,C,D) + X[i]) <<< S$. Здесь A,B,C,D – внутренние переменные; X[i] – текущее обрабатываемое слово блока сообщения; C_1 – константа, которая на первом шаге преобразования не используется.

На втором этапе обработки функция F выглядит следующим образом:

$$F(A,B,C,D,X[i],S)= (A + g(B,C,D) + X[i] + 5A827999) <<< S,$$

где C_1 = 5A827999 – константа, равная квадратному корню из 2 в шестнадцатеричном виде.

В результате использования в каждом цикле различных констант повышается стойкость алгоритма против вскрытия.

Так, после обнаружения успешной атаки на 2 из 3 циклов алгоритма MD4 был разработан более надежный вариант данного алгоритма – MD5, в котором на каждом шаге обработки уже используются уникальные константы.

Исходя из изложенного выше, можно сделать вывод, что использование алгоритмов хеширования, основанных на нелинейных битовых функциях, в большинстве случаев предпочтительнее, чем использование алгоритмов хеширования, основанных на использовании алгоритмов блочного шифрования [12]. Это объясняется аналогичными характеристиками по сложности вскрытия и высокой скорости хеширования. Естественно, при выборе конкретного алгоритма следует учитывать его возможность противостоять различным атакам на хеш-функцию.

4. Функции хеширования на базе алгоритмов симметричного шифрования

Алгоритмы хеширования, использующие в своей основе алгоритмы блочного шифрования, могут иметь почти такую же высокую скорость обработки, что и битовые алгоритмы. В дополнение к этому в них используется математический аппарат, который проверен на стойкость для режима шифрования.

Большинство алгоритмов хеширования данного класса основано на блочном шифровании в режиме связки шифроблоков, на фиксированном ключе и $V_н$ или НЗ (начальное значение). Последний зашифрованный блок является значением хеширования. Такие алгоритмы хеширования реализованы с использованием соответствующих режимов шифраторов DES [13], IDEA либо ГОСТ 28147.

Так, в качестве функции хеширования можно использовать алгоритм блочного шифрования в режиме выработки имитовставки. Однако при этом значение хеширования имеет длину, не превышающую размер обрабатываемого данным алгоритмом шифрования. Например, для алгоритма ГОСТ 28147 длина имитовставки (хеш-функции) равна всего 64 бита, что недостаточно для формирования стойкого значения хеширования. Поэтому данные алгоритмы далее не рассматриваются.

На наш взгляд, более перспективные и эффективные алгоритмы выработки функции хеширования используют блок сообщения в качестве ключа, предыдущее значение хеширования в качестве входа, и в качестве выхода – текущее значение хеширования. Фактически такие функции хеширования более сложные, чем рассмотренные выше. Размер блока обычно (но не обязательно) равен длине ключа, размер значения хеширования равен размеру блока. Поскольку большинство алгоритмов блочного шифрования 64-битные, некоторые разработанные схемы используют хеш в два (четыре) раза больше размера блока для формирования функции хеширования достаточной длины.

Для них безопасность функции хеширования основывается на безопасности используемого блочного алгоритма. Однако это не всегда так. Различные методы криптоанализа легче реализовать против блочных функций шифрования, используемых для хеширования, чем против расшифрования сообщений, зашифрованных при помощи этих алгоритмов. Поскольку известен ключ, можно применить различные специальные приемы, основанные на этих известных данных.

Для успешной атаки необходима только одна пара хеш-функций. В то же время можно сгенерировать столько простых и удобных для аналитика тестов, сколько ему нужно, и провести над ними любое количество необходимых операций.

Так, в результате криптоанализа хеш-функций, построенных на основании схем Prennel-Bosselaers-Govaers-Vandewalle, Quisquater-Girault, LOKI Double-Block и некоторых других [1], удалось найти в них слабые места против описанных выше атак. Поэтому использование данных схем при создании алгоритма хеширования не всегда обосновано.

В то же время для схем построения функции хеширования не было найдено атаки лучше, чем полный перебор. Первая из них – параллельный и последовательный алгоритм на основании схемы Davies-Meyer [14].

Эти схемы работают на шифраторе IDEA с 64-битным блоком и 128-битным ключом. Обе схемы генерируют значение хеширования длиной 128 бит. Первая имеет вид

$G_0=I_G$, где I_G — случайное начальное значение;

$H_0=I_H$, где I_H — другое случайное начальное значение;

$$G_i= G_{i-1}\oplus E_{Mi,Gi-1}(G_{i-1}); \qquad (10)$$

$$H_i= H_{i-1}\oplus E_{Gi-1,Mi}(H_{i-1}). \qquad (11)$$

Здесь $E_{X,Y}$ (A) представляет собой шифрование 64-битного блока A на ключе, равном конкатенации X и Y.

Вторая может быть записана в виде

$G_0=I_G$, где I_G — случайное начальное значение;

$H_0=I_H$, где I_H — другое случайное начальное значение;

$$G_i= G_{i-1}\oplus E_{M_i,H_{i-1}}(\neg\neg G_{i-1}); \qquad (12)$$

$$H_i= H_{i-1}\oplus E_{G_{i-1},M_i}(H_{i-1}). \qquad (13)$$

Как видно, данные схемы отличаются только использованием во втором случае инверсии предыдущего значения G.

Результатом хеширования всего сообщения является конкатенация значений G и H, полученных для последнего блока сообщения.

Предполагается [1], что данные алгоритмы имеют идеальную безопасность для 128-битного значения функции хеширования. Для нахождения сообщения с заданным значением функции хеширования требуется 2^{128} попыток, атака создания коллизий требует 2^{64} операций. Это означает, что не существует лучшей атаки, чем полный перебор.

Вторая схема – стандарт хеширования России ГОСТ Р 34.11-94.

Данный алгоритм использует алгоритм блочного шифрования ГОСТ 28147, но теоретически может работать с любым шифратором, имеющим размер блока 64 бита и размер ключа 256 бит. Хеширование блока сообщения представляет собой функцию $H_i=\chi(M_i,H_{i-1},S)$ трех 256-битных значений: текущего блока сообщения, значения хеширования предыдущего блока и переменной S. Эта переменная вычисляется следующим образом:

— генерация ключей шифрования ГОСТ 28147 с применением определенного линейного перемешивания M_i, H_{i-1} и определенных констант;

— использование каждого из 4-х ключей для шифрования 4-х 64-битных блоков текущего блока в режиме простой замены на 4-х соответствующих ключах.

Данная функция представляет собой сложную, хотя и линейную функцию от S, M_I и H_{I-1}.

Значение функции хеширования представляет собой следующее:

$$H=\chi(Z\oplus M', f(L, f(M', H_n)), S),$$

где Z – сумма по модулю 2 всех блоков сообщения; M′ – последний расширенный блок сообщения; H_n – значение хеширования предпоследнего блока сообщения; L – длина сообщения.

Как видно из приведенного, в алгоритме применены все возможные методы защиты получаемого значения функции хеширования от различных атак, хотя это потребовало значительных вычислительных сложностей. Производительность алгоритма (табл. 1) не выдерживает никакой критики.

В то же время следует отметить, что если при вычислении функции хеширования требуется повышенная надежность против подделки, то использование алгоритма ГОСТ 34.11 и ему подобных является оправданным.

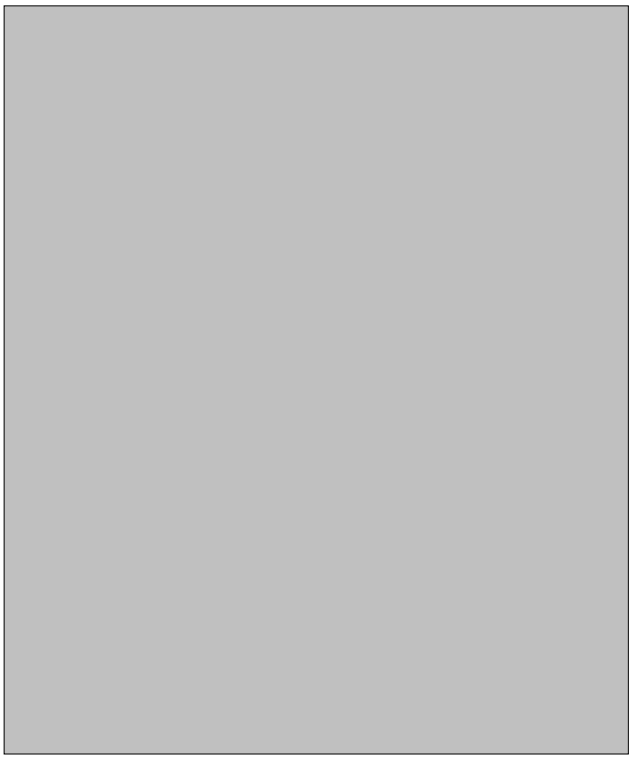
5. Заключение. Сравнительный анализ алгоритмов хеширования информации

Мы подробно не рассматриваем алгоритмы хеширования, базирующиеся на несимметричном сжатии данных. Это связано, прежде всего, с большой вычислительной сложностью такого сжатия, следовательно, низкой скоростью или большой стоимостью хеширования. На наш взгляд, хеширование с использованием несимметричных преобразований может применяться только для очень коротких данных (сообщений) – буквально один или несколько блоков, а также для "растяжения" данных (паролей), когда $l_m< l_h$. Компромиссом в разрешении этого противоречия является применение несимметричной ЦП, выработка которой производится с использованием значения h(M) хеш-функции подписываемых данных (сообщения). Например в стандарте США DSS [15] используется рассмотренная выше хеш-функция SHA, а в стандарте ГОСТ Р 34.10-94-хеш-функция ГОСТ Р 34.11-94.

Приведенное выше позволяет сделать вывод, что основным показателем качества алгоритма хеширования является защищенность алгоритма от атак и скорость обработки (хеширования) входного сообщения (данных).

Результаты анализа защищенности алгоритмов хеширования от атак приведены в табл.1, где показаны значения скорости работы различных функций хеширования, которая составлена с использованием результатов [1], а также на основании экспериментальных исследований авторов. При различных реализациях абсолютные значения скоростей хеширования могут изменяться, но соотношение между их скоростями будет выдержано.

Таблица 1



Как видно из приведенных значений, наилучшими, с точки зрения скорости хеширования, являются алгоритмы SHA, MD5, HAVAL, RIPE-MD и алгоритмы, основанные на блочном шифровании.

В табл. 2 приведены значения скоростей хеширования для некоторых алгоритмов, полученные на компьютере P133 экспериментальным путем.

Алгоритм ГОСТ Р-34.11-94 является низкоскоростным, он может применяться в случаях, когда стойкость важнее быстродействия.

Таблица 2

Приемлемыми в основном показателями обладает алгоритм хеширования SHA, разработанный специально для стандарта DSS.

Однако в большинстве случаев применяются алгоритмы MD5 и MD4, которые обеспечивают высокую скорость хеширования, если значение хеш-функции используется не в явном виде, а только как параметр при вычислении ЦП. В этом случае значение хеш-функции при ЦП шифруется (и в DSS, и в ГОСТ Р 34.10-94).

Но MD4 превосходит по быстродействию MD5. Проведенные на него атаки (на 2 из 3 блоков) не были расширены на весь алгоритм, поэтому можно считать его практически безопасным, особенно если он используется для ЦП.

Выбор алгоритма хеширования можно произвести на основании требований конкретной реализации системы. Например, если требуется алгоритм, обеспечивающий высокое быстродействие, и длина полученного значения хеширования может быть 128 бит, то лучше использовать алгоритм MD5. В то же время при использовании алгоритма хеширования в алгоритмах ЦП для стандарта DSS всегда необходимо использовать SHA либо другой алгоритм с длиной значения хеширования 160 бит; для ГОСТ Р-34.10 – алгоритм ГОСТ Р-34.11 либо, если требуется высокое быстродействие, — алгоритм MD4 в режиме формирования 256-битного значения (использование алгоритма 2 раза с различными начальными значениями).

Литература: 1. *Schneier Bruce.* Applied cryptography, second edition. 1996. 576p. 2. *B. der Boer, Bosselaers A.* Collisions for the compression function of MD5. Advances in Cryptology. Proc. Eurocrypt'93. LNCS 765. T. Helleseth, Ed.//Springer-Verlag. 1994. P.293-304. 3. *Yuval G.* How to Swindle Rabin// Cryptologia. 1979. Vol. 3, N 3. P. 187-190. 4. *Kaliski B.S.,* The MD2 Message Digest Algorithm. Advances in Cryptology// CRYPTO'90 Proceedings. 1992. P.120-123. 5. *Rivest R.L.* The MD4 Message Digest Algorithm. Advances in Cryptology – CRYPTO'90 Proceedings". 1992. P. 30-35. 6. *Rivest R.L.* The MD5 Message Digest Algorithm. Advances in Cryptology// CRYPTO'90 Proceedings. 1990. P. 40-44. 7. *FIPS 180-1* Secure hash standard. NIST, US Department of Commerce. Washington D.C. 1995. 8. *Rivest R.L.* RSA Chips (Past/Present/Future). Advances in Cryptology EUROCRYPT 84. 1985. P.159-168. 9. *Merkle R.C.* A fast Software One-Way Hash Function//Journal of Cryptology. 1990. Vol. 3, N 1. P. 43-58. 10. *Miyaguchi S., Ohta K., Iwata M.* 128 bit Hash Function (N-Hash) // Proceedings of SECURICOM'90. 1990. P. 127-137. 11. *Biham E., Shamir A.* Differential Cryptoanalysis of Data Encryption Standard. Springer-Verlag. 1993. P. 45. 12. *Preneel B.* Analysis and Design of Cryptographic Hash Function. Ph.D. dissertation. 1993. P. 180. 13. *National Bureau of Standards NBS FIPS PUB 46* Data Encryption Standard//National Bureau of Standards US Department of Commerce. 1977. P. 56. 14. *Lai X. and Massey J.* Hash function based on Block Chipers. Advances in Cryptology EUROCRYPT 92//Springer-Verlag. 1992. P. 50-70. 15. *National Institute of Standards and Technology, NBS FIPS PUB 186* Digital Signature Standard. US Department of Commerce. 1994. P.84.

Поступила в редколлегию 23.03.98

Горбенко Иван Дмитриевич, д-р техн. наук., профессор, проректор по научной работе ХТУРЭ. Научные интересы: защита информации в компьютерных системах и сетях. Адрес: 310726, Украина, Харьков, пр.Ленина, 14, тел. 30-24-50, 37-56-39.

Штанько Игорь Анатольевич, аспирант кафедры ПОЭВМ ХТУРЭ. Научные интересы: защита информации. Увлечения и хобби: программирование. Адрес: 310726, Украина, Харьков, пр. Ленина, 14, тел. 63-95-33.