## Методы и средства обеспечения целостности и достоверности используемого программного кода.

###### 3.3.1. Методы защиты программ от несанкционированных изменений

Решение проблемы обеспечения целостности и достоверности электронных данных включает в себя решение, по крайней мере, трех основных взаимосвязанных задач: подтверждения их авторства и подлинности, а также контроль целостности данных. Решение этих трех задач в случае защиты программного обеспечения вытекает из необходимости защищать программы от следующих злоумышленных действий:

* РПС может быть внедрены в авторскую программу или эта программа может быть полностью заменена на программу-носитель РПС;
* могут быть изменены характеристики (атрибуты) программы;
* злоумышленник может выдать себя за настоящего владельца программы;
* законный владелец программы может отказаться от факта правообладания ею.

Наиболее эффективными методами защиты от подобных злоумышленных действий предоставляют криптографические методы защиты. Это обусловлено тем, что хорошо известные способы контроля целостности программ, основанные на контрольной сумме, продольном контроле и контроле на четность, как правило, представляют собой довольно простые способы защиты от внесения изменений в код программ. Так как область значений, например, контрольной суммы сильно ограничена, а значения функции контроля на четность вообще представляются одним-двумя битами, то для опытного нарушителя не составляет труда найти следующую коллизию: f(k1)=f(k2), где k1 - код программы без внесенной нарушителем закладки, а k2- с внесенной программным закладкой и f - функция контроля. В этом случае значения функции для разных аргументов совпадают при тестировании и, следовательно, закладка обнаружена не будет.

Для установления подлинности (неизменности) программ необходимо использовать более сложные методы, такие как аутентификация кода программ, с использованием криптографических способов, которые обнаруживают следы, остающиеся после внесения преднамеренных искажений.

В первом случае аутентифицируемой программе ставится в соответствие некоторый аутентификатор, который получен при помощи стойкой криптографической функции. Такой функцией может быть криптографически стойкая хэш-функция (например, функция ГОСТ Р 34.11-94) или функция электронной цифровой подписи (например, функция ГОСТ Р 34.10-94). И в том, и в другом случае аргументами функции может быть не только код аутентифицируемой программы, но и время и дата аутентификации, идентификатор программиста и/или предприятия - разработчика ПО, какой-либо случайный параметр и т.п. Может использоваться также любой симметричный шифр (например, DES или ГОСТ 28147-89) в режиме генерации имитовставки. Однако, это требует наличия секретного ключа при верификации программ на целостность, что бывает не всегда удобно и безопасно. В то время как при использовании метода цифровой подписи при верификации необходимо иметь только некоторую общедоступную информацию, в данном случае открытый ключ подписи. То есть контроль целостности ПО может осуществить любое заинтересованное лицо, имеющее доступ к открытым ключам используемой схемы цифровой подписи.

Можно еще более усложнить действия злоумышленника по нарушению целостности целевых программ, используя схемы подписи с верификацией по запросу. В этом случае тестирование программ по ассоциированным с ними аутентификаторам можно осуществить только в присутствии лица, сгенерировавшего эту подпись, то есть в присутствии разработчика программ или представителей предприятия-изготовителя программного обеспечения. В этом случае, если даже злоумышленник и получил для данной программы некий аутентификатор, то ее обладатель может убедиться в достоверности программы только в присутствии специалистов-разработчиков, которые немедленно обнаружат нарушения целостности кода программы и (или) его подлинности.

###### 3.3.2. Краткое описание криптографических средств контроля целостности и достоверности программ

Основные положения криптологии и базовые криптографические понятия

Термин "криптология" происходит от двух греческих слов: "крипто", что означает "тайный" и "логос", т.е. - учение.

Криптология как наука, состоит из двух тесно теоретически и практически связанных дисциплин: криптографии и криптоанализа. Криптография - наука о способах преобразования (шифрования) информации с целью ее защиты от незаконных пользователей. Криптоанализ - наука (и практика ее применения) о методах и способах вскрытия шифров. Криптография и криптоанализ очевидным образом связаны друг с другом, так как не бывает хороших криптографов, не владеющих методами криптоанализа, и наоборот - хороший криптоаналитик должен быть знаком со всеми известными способами построения шифров. Ниже даются базовые понятия и определения криптологии, в т.ч. используемые и в настоящем разделе.

Шифр (криптосистема) - способ, метод преобразования информации с целью ее защиты от незаконных пользователей (от противника). Для противника возникает сложная задача вскрытия шифра. Вскрытие (взламывание) шифра - процесс получения информации из шифрованного сообщения (шифртекста) без знания примененного шифра.

Шифрование - процесс применения шифра к защищаемой информации, т.е. преобразование информации в шифрованное сообщение с помощью определенных правил, содержащихся в шифре.

Дешифрование - процесс, обратный шифрованию, т.е. преобразование шифрованного сообщения в защищаемую информацию с помощью определенных правил, содержащихся в шифре.

Исходное сообщение, имеющее, как правило, смысловое (логически значимое) содержание, которое необходимо зашифровать называется открытым текстом. Зашифрованное сообщение, имеющее, как правило, вид случайного набора символов (цифр) называется шифртекстом или криптограммой.

Под ключом в криптографии понимают сменный элемент шифра, который применен для шифрования конкретного открытого текста (сообщения).

В криптографии обычно общепринято следующее допущение. Криптоаналитик (противник) почти всегда имеет полный шифртекст. Помимо этого в криптографии принято правило Керкхоффа, которое гласит, что "стойкость шифра должна определяться только секретностью его ключа".

В этом случае задача противника сводится к попытке раскрытия шифра (попытке осуществления атаки) на основе шифртекста. Если же противник имеет к тому же некоторые отрывки открытого текста и соответствующие им элементы шифртекста, тогда он пытается осуществлять атаку на основе открытого текста. Атака на основе выбранного открытого текста заключается в том, что противник, используя свой открытый текст, получает правильный шифртекст (например, используя "вслепую" некоторую шифрмашину) и пытается в этом случае вскрыть шифр. Попытку раскрытия шифра можно осуществить, если противник подставляет свой ложный шифртекст и при дешифровании получает необходимый для раскрытия шифра открытый текст. Такой способ раскрытия называется атакой на основе выбранного шифртекста.

Теоретически существует абсолютно стойкий шифр, но единственным таким шифром является какая-нибудь форма так называемой ленты однократного использования (или так называемый "одноразовый блокнот"), в которой открытый текст "объединяется" с полностью случайным ключом такой же длины. Этот результат был доказан К. Шенноном с помощью разработанного им теоретико-информационного метода исследования шифров.

Для абсолютной стойкости существенным является каждое из следующих требований к ленте однократного использования:

* полная случайность (равновероятность) ключа (это, в частности, означает, что ключ нельзя вырабатывать с помощью какого-либо детерминированного устройства);
* равенство длины ключа и длины открытого текста;
* однократность использования ключа.

В то же время, именно эти условия и делают абсолютно стойкий шифр очень ресурсозатратным и непрактичным. Прежде чем пользоваться таким шифром, необходимо обеспечить всех абонентов достаточным запасом случайных ключей и исключить возможность их повторного применения. А это сделать необычайно трудно и дорого. В силу данных причин абсолютно стойкие шифры применяются только в сетях связи с небольшим объемом передаваемой информации, обычно это сети для передачи особо важной государственной информации.

В 1976 г. опубликовав свою работу "Новые направления в криптографии", американские ученые У. Диффи и М. Хеллман выдвинули следующую удивительную гипотезу: "Возможно построение практически стойких криптосистем, вообще не требующих передачи секретного ключа". Такие криптосистемы, получившие название криптосистем с открытым ключом, основываются на введении понятий "односторонней функции" и "односторонней функции с секретом". Понятие односторонней функции было введено в подразделе 2.4

Вопрос о существовании односторонней функции с секретом является столь же гипотетическим, что и вопрос о существовании односторонней функции. Для практических целей было построено несколько функций, которые могут оказаться односторонними, а это означает, что задача инвертирования эквивалентна некоторой давно изучаемой трудной математической задаче.

Применение односторонних функций в криптографии позволяет: во-первых, организовать обмен шифрованными сообщениями с использованием только открытых каналов связи и, во-вторых, решать новые криптографические задачи, такие как электронная цифровая подпись.

В большинстве схем электронной подписи используются хэш-функции. Это объясняется тем, что практические схемы электронной подписи не способны подписывать сообщения произвольной длины, а процедура, состоящая в разбиении сообщения на блоки и в генерации подписи для каждого блока по отдельности, крайне неэффективна. Под термином "хэш-функция" понимается функция, отображающие сообщения произвольной длины в значение фиксированной длины, которое называется хэш-кодом.

Далее рассмотрим базовые криптографические методы, широко применяющиеся в современных системах обеспечения безопасности информации.

Краткое описание основных криптографических методов защиты данных

Симметричный шифр ГОСТ 28147-89

Пусть L и R - последовательности битов, LR означает их конкатенацию. Под обозначением http://citforum.ru/security/articles/kazarin/plus.gif будет пониматься операция сложения по модулю 2 или логическая операция XOR (исключающая ИЛИ), символом [+] - операция сложения по модулю 232 двух 32-разрядных чисел. Числа суммируются по следующему правилу:

A[+]B=A*http://citforum.ru/security/articles/kazarin/plus.gif*B, если A*http://citforum.ru/security/articles/kazarin/plus.gif*B < 232

A[+]B=A*http://citforum.ru/security/articles/kazarin/plus.gif*B-232, если A*http://citforum.ru/security/articles/kazarin/plus.gif*B*http://citforum.ru/security/articles/kazarin/m.gif*232.

Символом {+} обозначается операция сложения по модуль 232-1 двух 32-разрядных чисел. Правила суммирования чисел следующие:

A{+}B=A*http://citforum.ru/security/articles/kazarin/plus.gif*B, если A*http://citforum.ru/security/articles/kazarin/plus.gif*B < 232-1

A{+}B=A*http://citforum.ru/security/articles/kazarin/plus.gif*B-232, если A*http://citforum.ru/security/articles/kazarin/plus.gif*B*http://citforum.ru/security/articles/kazarin/m.gif*232-1.

Во всех режимах работы алгоритма используется ключ длиной 256 битов, который представляется в виде восьми 32-разрадных чисел X(i).Если обозначить ключ через W, то

W=X(7)X(6)X(5)X(4)X(3)X(2)X(1)X(0).

Дешифрование, как и в любой симметричной криптосистеме осуществляется на том же ключе, что и шифрование. Ниже приводится описание двух наиболее используемых режимов шифра: режима простой замены и режима генерации имитовставки [[6]](http://citforum.ru/security/articles/kazarin/4.shtml#6).

Описание режима простой замены. Код программы T разбивается на блоки по 64 бита в каждом, которые обозначаются T(j). Очередная последовательность битов T(j) разбивается на две последовательности B(0) (левые или старшие биты) и A(0) (правые или младшие биты), каждая из которых содержит 32 бита. Затем выполняется итеративный процесс шифрования, который описывается следующими формулами:

при i=1,2,...,24; j=i-1(mod 8)

A(i)=f(A(i-1)[+]X(j)(+)B(i-1));

B(i)-A(i-1);

при i=25,26,...,31; j=32-i

A(i)=f(A(i-1)[+]X(j)(+)B(i-1));

B(i)-A(i-1);

при i=32

A(32)=A(31);

B(32)=f(A(31)[+]X(0)(+)B(31)),

где i обозначает номер итерации (i=1,2,...,32). Функция f называется функцией шифрования. Ее аргументом является сумма по модулю 232 числа A(i), полученного на предыдущем шаге итерации, в числа X(j) ключа (размерность каждого из этих чисел 32 знакам).

Функция шифрования включает две операции над полученной 32-разрядной суммой. Первая операция называется подстановкой K. Блок подстановки K состоит из восьми узлов замены K(1)...K(8) с памятью 64 бита каждый. Поступающий на блок подстановки 32-разрядный вектор разбивается на 8 последовательно идущих 4-разрядных векторов, каждый из которых преобразуется в 4-разрядный вектор соответствующим узлом замены, представляющим собой таблицу из 16 целых чисел в диапазоне 0,...,15.

Входной вектор определяет адрес строки в таблице, число из которой является выходным вектором. Затем 4-разрядные выходные векторы последовательно объединяются в 32-разрядный вектор. Таблицы блока подстановки блока подстановки K содержат редко изменяемые ключевые элементы, общие для некоторой компьютерной системы.

Вторая операция - циклический сдвиг влево 32-разрядного вектора, полученного в результате подстановки K. 64-разрядный блок зашифрованных данных Тш представляется в виде:

Тш=A(32)B(32).

Остальные блоки кода программы в режиме простой замены шифруются аналогично.

Режим генерации имитовставки. Для получения имитовставки код программы представляется в виде 64-разрядных блоков T(i), =1,2,..,m. Где mопределяет объем кода программы. Первый блок кода программы T(i) подвергается преобразованию, соответствующему первым 16 циклам алгоритма шифрования в режиме простой замены, причем в качестве ключа для выработки имитовставки используется ключ, по которому шифруются данные.

Полученное после 16 циклов работы 64-разрядное число суммируется по модулю 2 со вторым блоком открытых данных T(2). Результат суммирования снова подвергается преобразованию, соответствующему 16 циклам алгоритма шифрования в режиме простой замены. Полученное 64-разрядной число суммируется по модулю 2 с третьим блоком данных T(3) и т.д. Последний блок T(m), при необходимости дополненный до полного 64-разрядного блока нулями, суммируется по модулю 2 с результатом работы на шаге m-1, после чего шифруется в режиме простой замены по первым 16 циклам алгоритма. Из полученного 64-разрядного числа выбирается отрезок Ир длиной р битов. Данный отрезок и является имитовставкой Ир, полученной для кода программы T.

Открытый ключевой обмен Диффи-Хеллмана и криптосистемы с открытым ключом

Основная задача ключевого обмена Диффи-Хеллмана заключается в следующем: "Каким образом можно установить секретный ключ между абонентами **А** и **В** по открытому каналу связи а затем использовать его для шифрованной передачи сообщений?" Для этих целей, пусть абонент A выбирает какую-нибудь функцию fk с секретом k. Он публикует в открытом сертифицированном справочнике описание функции fk в качестве своего алгоритма шифрования. Однако, значение секрета k он никому не сообщает, т.е. держит его в тайне от других.

Если абонент **B** хочет послать **А** защищаемую информацию x*http://citforum.ru/security/articles/kazarin/e.gif*X, то он вычисляет y=fk(x) и посылает y по открытому каналу к **A**. Поскольку **A** для своего секрета k умеет инвертировать fk, то он вычисляет x по полученному y. Так как никто другой не знает k и поэтому в силу свойств функции с секретом не сможет за полиномиальное время по известному шифрованному сообщению y вычислить защищаемую информацию x.

Описанная выше схема является криптосистемой с открытым ключом, поскольку алгоритм шифрования fk является общедоступным или открытым. Такие криптосистемы называют еще асимметричными, поскольку в них есть асимметрия в алгоритмах: алгоритмы шифрования и дешифрования различны. В отличие от таких систем традиционные шифры называют симметричными, так как в них ключ для шифрования и дешифрования один и тот же, и именно поэтому его нужно хранить в секрете. Для асимметричных систем алгоритм шифрования общеизвестен, но восстановить по нему алгоритм дешифрования за полиномиальное время невозможно.

Электронная цифровая подпись

Основные определения, обозначения и алгоритмы. Для реализации схем электронной цифровой подписи(или просто цифровой подписи) требуются три следующих эффективно функционирующих алгоритма:

* Ak - алгоритм генерации секретного и открытого ключей для подписи кода программы, а также проверки подписи, - s и p соответственно;
* As - алгоритм генерации (проставления) подписи с использованием секретного ключа s;
* Ap - алгоритм проверки (верификации) подписи с использованием открытого ключа p.

Алгоритмы должны быть разработаны так, чтобы выполнялось основное принципиальное свойство, - свойство невозможности получения нарушителем (противником) алгоритма As из алгоритма Ap.

Таким образом, если Ak - алгоритм генерации ключей, тогда определим значения (s,p)=Ak(α,β) как указанные выше сгенерированные ключи, где α - некоторый параметр безопасности (как правило, длина ключей), а β - параметр, характеризующий случайный характер работы алгоритма Ak при каждом его вызове.

Ключ s хранится в секрете, а открытый ключ p делается общедоступным. Это делается, как правило, путем помещения открытых ключей пользователей в открытый сертифицированный справочник. Сертификация открытых ключей справочника выполняется некоторым дополнительным надежным элементом, которому все пользователи системы доверяют обработку этих ключей. Обычно этот элемент называют Центром обеспечения безопасности или Центром доверия.

Непосредственно процесс подписи осуществляется посредством алгоритма As. В этом случае значениеc=As(m) - есть подпись кода программы m, полученная при помощи алгоритма As и ключа s.

Процесс верификации выполняется следующим образом. Пусть m\* и c\* - код программы и подпись для этого кода соответственно, Ap - алгоритм верификации. Тогда, выбрав из справочника общедоступный открытый ключ p, можно выполнить алгоритм верификации: b=Ap(m\*,c\*), где предикат bhttp://citforum.ru/security/articles/kazarin/e.gif{true,false}. Если b=true, то код m\* действительно соответствует подписи c\*, полученной при помощи секретного ключа s, который, в свою очередь, соответствует открытому ключу p, то есть m=m\*, c=c\* и наоборот еслиb=false, то код программы ложен и (или) код подписан ложным ключом.

Примеры схем электронной цифровой подписи. К наиболее известным схемам цифровой подписи с прикладной точки зрения относятся схемы RSA, схемы Рабина-Уильямса, Эль-Гамаля, Шнорра и Фиата-Шамира, а также схема электронной цифровой подписи отечественного стандарта ГОСТ Р 34.10-94.

Рассмотрим несколько подробнее четыре наиболее известные схемы цифровой подписи:

**Подпись RSA:** Вход: Числа n, p, q, где p и q - большие l - разрядные простые числа, n=pq. Открытый ключp=(e,n), секретный ключ s=d, такой, что edhttp://citforum.ru/security/articles/kazarin/3r.gif1(mod φ(n)) и наибольший общий делитель НОД(e,φ(n))=1, гдеφ(n)=(p-1)(q-1).

Генерация ключей:(d,(e,n))=Ak(l,b).

Подпись:As: 1.md(mod n)*http://citforum.ru/security/articles/kazarin/3r.gif*c, где c - подпись

кода программы m.

Верификация:Ap: 1.[c\*]e(mod n)*http://citforum.ru/security/articles/kazarin/3r.gif*m\*\*,

2.если m\*\*=m\*, подпись верна.

**Подпись Эль-Гамаля:** Вход: Числа p и g, где p - простое l - разрядное число, а g - первообразный корень по модулю p. Секретный ключ s=d, открытый ключ p=e, такой, что e*http://citforum.ru/security/articles/kazarin/3r.gif*gd (mod p), m - подписываемый код программы.

Генерация ключей:(d,(e,g,p))=Ak(l,b).

Подпись: As: 1.r*http://citforum.ru/security/articles/kazarin/3r.gif*gk(mod p), где khttp://citforum.ru/security/articles/kazarin/e.gifRZp-1;

2.находится такое c, что

mhttp://citforum.ru/security/articles/kazarin/3r.gif[kc+dr](mod p-1),

где (c,r) - подпись кода m.

Верификация:Ap: 1.если gm\*=errc\*(mod p),  
 то подпись верна.

**Подпись Фиата - Шамира:** Вход: Числа n, p, и q, где p и q большие l - разрядные простые числа, открытый ключ p есть вектор (v1,v2,...,vk), где vj - квадратичные вычеты по модулю n, j=http://citforum.ru/security/articles/kazarin/59.gif , секретный ключ p есть вектор (s1,s2,...,sk), где каждый sj - наименьший квадратный корень из v-1j, m - подписываемый код, f - псевдослучайная функция. Генерация ключей: ((s1,s2,...,sk),(v1,v2,...,vk))=Ak(l,b).

|  |  |
| --- | --- |
| Подпись: | As: 1.xi*http://citforum.ru/security/articles/kazarin/3r.gif*r2i(mod n), где {r1,r2,...,rk}http://citforum.ru/security/articles/kazarin/e.gifRZn; 2.вычисляется значение a=f(m,x1,x2,...,xt); 3.выбирается первые kt битов числа a как матрица http://citforum.ru/security/articles/kazarin/60.gif 4.вычисляется: http://citforum.ru/security/articles/kazarin/61.gif где http://citforum.ru/security/articles/kazarin/62.gif Тогда (eij,yi) - подпись кода m. |
| Верификация: | http://citforum.ru/security/articles/kazarin/63.gif 2.если первые kt битов значения  функции f(m\*,z1,z2,...,zt) равны e\*ij,  подпись верна. |

**Подпись стандарта ГОСТ Р 34.10-94.** Вход: Числа p, g и q где p - простое l -разрядное число, g - первообразный корень по модулю p, а q - большой простой делитель p-1. Пусть также gqhttp://citforum.ru/security/articles/kazarin/3r.gif1(mod p), ghttp://citforum.ru/security/articles/kazarin/nerav.gif1. Секретный ключ подписи x (1 < x < q) и открытый ключ y*http://citforum.ru/security/articles/kazarin/3r.gif*gx(mod p).

|  |  |
| --- | --- |
| Генерация ключей: | (x,(g,p,q,y))=Ak(l,b). |
| Подпись: | Ax: 1.r*http://citforum.ru/security/articles/kazarin/3r.gif*[gk(mod p)](mod q), где k*http://citforum.ru/security/articles/kazarin/e.gif*RZq. 2.shttp://citforum.ru/security/articles/kazarin/3r.gif[xr+km)](mod q), где m=h(M) -рассматривается как значение хэш-функции h, соответствующей отечественному стандарту на функцию хэширования сообщений ГОСТ Р 34.11-94. Таким образом, пара (r,s) - есть подпись кода программы M. |
| Верификация: | Ay: 1.v*http://citforum.ru/security/articles/kazarin/3r.gif*m-1(mod q). 2.uhttp://citforum.ru/security/articles/kazarin/3r.gif[gsvy-rv(mod p)](mod q); 3.Если u=r, то (r,s) - есть подпись кода программы M, где m=h(M). |

Под стойкостью схемы цифровой подписи понимается стойкость в теоретико-сложностном смысле, то есть, как отсутствие эффективных алгоритмов ее подделки и (или) раскрытия. По-видимому, до сих пор основной является следующая классификация:

* атака с открытым ключом, когда противник знает только открытый ключ схемы;
* атака на основе известного открытого текста (известного кода программы), когда противник получает подписи для некоторого (ограниченного) количества известных ему кодов (при этом противник никак не может повлиять на выбор этих кодов);
* атака на основе выбранного открытого текста, когда противник может получить подписи для некоторого ограниченного количества выбранных им кодов программы (предполагается, что коды выбираются независимо от открытого ключа, например, до того как открытый ключ станет известен);
* направленная атака на основе выбранного открытого текста (то же, что предыдущая, но противник, выбирая код программы, уже знает открытый ключ);
* адаптивная атака на основе выбранного открытого текста, когда противник выбирает коды программы последовательно, зная открытый ключ и зная на каждом шаге подписи для всех ранее выбранных кодов.

Атаки перечислены таким образом, что каждая последующая сильнее предыдущей.

Угрозами для схем цифровой подписи являются раскрытие схемы или подделка подписи. Определяются следующие типы угроз:

* полного раскрытия, когда противник в состоянии вычислить секретный ключ подписи;
* универсальной подделки, когда противник находит алгоритм, функционально эквивалентный алгоритму вычисления подписи;
* селективной подделки, когда осуществляется подделка подписи для кода программы, выбранного противником априори;
* экзистенциональной подделки, когда осуществляется подделка подписи хотя бы для одного кода программы (при этом противник не контролирует выбор этого кода, которое может быть вообще случайным или бессмысленным).

Угрозы перечислены в порядке ослабления. Стойкость схемы электронной подписи определяется относительно пары (тип атаки, угроза).

Если принять вышеописанную классификацию атак и угроз, то наиболее привлекательной является схема цифровой подписи, стойкая против самой слабой из угроз, в предположении, что противник может провести самую сильную из всех атак. В этом случае, такая схема должна быть стойкой против экзистенциональной подделки с адаптивной атакой на основе выбранного открытого текста.

Разновидности схем подписи электронных сообщений. Для обеспечения целостности и достоверности информации при ее передаче и хранении, а также для проведения аутентификации абонентов системы и решения других задач защиты существует достаточно много всевозможных разновидностей схем подписи. Без подробного описания ниже приводится одна из разновидностей схем подписи, которая, как правило, использует рассмотренный выше математический аппарат и может использоваться для защиты программ.

Схема подписи с верификацией по запросу. В работах Д. Шаума впервые была предложена схема подписи с верификацией по запросу, в которой абонент **V** не может осуществить верификацию подписи без участия абонента **S**. Такие схемы могут эффективно использоваться в том случае, когда фирма - изготовитель поставляет потребителю некоторый информационный продукт (например, программное обеспечение) с проставленной на нем подписью указанного вида. Однако проверить эту подпись, которая гарантирует подлинность программы или отсутствие ее модификаций, можно только уплатив за нее. После факта оплаты фирма - изготовитель дает разрешение на верификацию корректности полученных программ.

Схема состоит из трех этапов (протоколов), к которым относятся непосредственно этап генерации подписи, этап верификации подписи с обязательным участием подписывающего (протокол верификации) и этап оспаривания, если подпись или целостность подписанных сообщений подверглась сомнению (отвергающий протокол).

I. Генерация подписи. Пусть каждый пользователь S имеет один открытый ключ P и два секретных ключаS1 и S2. Ключ S1всегда остается в секрете, - он необходим для генерации подписи. Ключ S2 может быть открыт для того, чтобы конвертировать схему подписи с верификацией по запросу в обычную схему электронной цифровой подписи.

Вместе с обозначениями секретного и открытого ключей x*http://citforum.ru/security/articles/kazarin/e.gif*RZq и y*http://citforum.ru/security/articles/kazarin/e.gif*RZ\*p (взятых из отечественного стандарта на электронную цифровую подпись) введем также обозначения S1=x и S2=u, u*http://citforum.ru/security/articles/kazarin/e.gif*RZq, а также открытый ключ P=(g,y,w), где w*http://citforum.ru/security/articles/kazarin/3r.gif*gu(mod p). Открытый ключ P публикуется в открытом сертифицированном справочнике.

Подпись кода m вычисляется следующим образом. Выбирается k*http://citforum.ru/security/articles/kazarin/e.gif*RZq и вычисляется rhttp://citforum.ru/security/articles/kazarin/3r.gifgk(mod p). Затем вычисляется s*http://citforum.ru/security/articles/kazarin/3r.gif*[xr+mku](mod q). Пара (r,s) является подписью для кода m. Подпись считается корректной тогда и только тогда, когда ru*http://citforum.ru/security/articles/kazarin/3r.gif*gswy-rw , где w*http://citforum.ru/security/articles/kazarin/3r.gif*m-1(mod q).

Проверка подписи (с участием подписывающего) осуществляется посредством следующего интерактивного протокола.

II. Протокол верификации. Абонент вычисляет γ*http://citforum.ru/security/articles/kazarin/3r.gif*gswy-rw(mod p) и просит абонента S доказать, что пара (r,s) есть его подпись под кодом m. Эта задача эквивалентна доказательству того, что дискретный логарифм γ по основанию r равен (по модулю p) дискретному логарифму w по основанию g, то есть, чтоlogg(p)w*http://citforum.ru/security/articles/kazarin/3r.gif*logr(p)γ. Для этого:

1. Абонент **V** выбирает a,b*http://citforum.ru/security/articles/kazarin/e.gif*RZq, вычисляет δhttp://citforum.ru/security/articles/kazarin/3r.gifragb(mod p) и посылает δ абоненту S.
2. Абонент **S** выбирает t*http://citforum.ru/security/articles/kazarin/e.gif*RZq, вычисляет h1*http://citforum.ru/security/articles/kazarin/3r.gif*δg1(modp), h2*http://citforum.ru/security/articles/kazarin/3r.gif*h1u(mod p) и посылает h1и h2 абоненту **V**.
3. Абонент **V** высылает параметры a и b.
4. Если δhttp://citforum.ru/security/articles/kazarin/3r.gifragb(mod p) , то абонент S посылает V параметр t; в противном случае - останавливается.
5. Абонент **V** проверяет выполнение равенств h1*http://citforum.ru/security/articles/kazarin/3r.gif*ragb+1(mod p) и h2*http://citforum.ru/security/articles/kazarin/3r.gif*γawb+1(mod p) .

Если проверка завершена успешно, то подпись принимается как корректная.

III. Отвергающий протокол. В отвергающем протоколе **S** доказывает, что logg(p)whttp://citforum.ru/security/articles/kazarin/nerav.giflogr(p)γ. Следующие шаги выполняются в циклеl раз.

1. Абонент **V** выбирает d,e*http://citforum.ru/security/articles/kazarin/e.gif*RZq, dhttp://citforum.ru/security/articles/kazarin/nerav.gif1, βhttp://citforum.ru/security/articles/kazarin/e.gifR{0,1}. Вычисляет a*http://citforum.ru/security/articles/kazarin/3r.gif*ge(mod p), b*http://citforum.ru/security/articles/kazarin/3r.gif*we(mod p), если β=0 и a*http://citforum.ru/security/articles/kazarin/3r.gif*re(mod p) , b*http://citforum.ru/security/articles/kazarin/3r.gif*γe(mod p), еслиβ=1. Посылает **S** значения a, b, d.
2. Абонент **S** проверяет соотношение au(mod p)http://citforum.ru/security/articles/kazarin/3r.gifb. Если оно выполняется, то a=0, в противном случаеa=1. Выбирает R*http://citforum.ru/security/articles/kazarin/e.gif*RZq, вычисляет c*http://citforum.ru/security/articles/kazarin/3r.gif*dagR(mod p) и посылает **V** значение c.
3. Абонент **V** посылает абоненту **S** значение e.
4. Абонент **S** проверяет, что выполняются соотношения из следующих двух их пар: a*http://citforum.ru/security/articles/kazarin/3r.gif*ge(mod p), b*http://citforum.ru/security/articles/kazarin/3r.gif*we(mod p) и a*http://citforum.ru/security/articles/kazarin/3r.gif*re(mod p), b*http://citforum.ru/security/articles/kazarin/3r.gif*γe(mod p). Если да, то посылает **V** значение R. Иначе останавливается.
5. Абонент **V** проверяет, что dβgR*http://citforum.ru/security/articles/kazarin/3r.gif*c .

Если во всех l циклах проверка в п.5 выполнена успешно, то абонент **V** принимает доказательства.

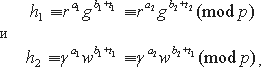
**Таблица 3.1.** Протокол верификации является интерактивным протоколом доказательств с абсолютно нулевым разглашением.

**Доказательство.** Требуется доказать, что вышеприведенный протокол удовлетворяет трем свойствам: полноты, корректности и нулевого разглашения.

Полнота означает, что если оба участника (**V** и **S**) следуют протоколу и (r,s) - корректная подпись для сообщения m, то **V** примет доказательство с вероятностью близкой к 1. Из описания протокола верификации очевидно, что абонент **S** всегда может надлежащим образом ответить на запросы абонента**V**, то есть доказательство будет принято с вероятностью 1.

Корректность означает, что если **V** действует согласно протоколу, то какие действия не предпринимал бы **S**, он может обмануть **V** лишь с пренебрежимо малой вероятностью. Здесь под обманом понимается попытка **S** доказать, что logg(p)w*http://citforum.ru/security/articles/kazarin/3r.gif*logr(p)γ, когда на самом деле эти логарифмы не равны.

Предположим, что logg(p)whttp://citforum.ru/security/articles/kazarin/nerav.giflogr(p)γ. Ясно, что для каждого a существует единственное значение b, то которое дает данный запрос δ. Поэтому δ не содержит в себе никакой информации об a. Если S смог бы найти h1, h2, t1и t2 такие, что



где a1http://citforum.ru/security/articles/kazarin/nerav.gifa2, то тогда выполнялось бы соотношение

logg(p)r*http://citforum.ru/security/articles/kazarin/3r.gif*[(a1-a2)-1((b2-b1)+(t2-t1))](modq)*http://citforum.ru/security/articles/kazarin/3r.gif*logw(p)γ.

Отсюда, очевидно, следует, что logg(p)w*http://citforum.ru/security/articles/kazarin/3r.gif*logr(p)γ. В самом деле, пусть logw(p)γ*http://citforum.ru/security/articles/kazarin/3r.gif*logg(p)r*http://citforum.ru/security/articles/kazarin/3r.gif*λ. Тогда

http://citforum.ru/security/articles/kazarin/65.gif

что противоречит предположению. Следовательно, какие бы h1, h2, t1и t2 не выбрал S, проверка, которую проводит **V**, может быть выполнена только для одного значения a. Отсюда вероятность обмана не превосходит 1/q. Отметим, что протокол верификации является безусловно стойким для абонента **V**, то есть доказательство корректности не зависит ни от каких предположений о вычислительной мощности доказывающего (**S**).

Свойство нулевого разглашения означает, что в результате выполнения протокола абонент **V** не получает никакой полезной для себя информации (например, о секретных ключах, используемых **S**). Для доказательства нулевого разглашения необходимо для любого возможного проверяющего V\* построить моделирующую машину MV , которая является вероятностной машиной Тьюринга, работает за полиномиальное в среднем время и создает на выходе (без участия **S**) такое же распределение случайных величин, которое возникает у V\* в результате выполнения протокола. В нашем случае, случайные величины, которые "видит" **V\***, - это h1, h2, и t. Необходимо доказать, что протокол верификации является доказательством с абсолютно нулевым разглашением, то есть моделирующая машина создает распределение случайных величин (h1,h2,t), которое в точности совпадает с их распределением, возникающим при выполнении протокола. Моделирующая машина MV использует в своей работе **V**\* в качестве "черного ящика".

Моделирующая машина

1. Запоминает состояние машины **V**\*, то есть содержимое всех ее лент, внутреннее состояние и позиции головок на лентах. Затем получает от **V**\* значение δ и после этого снова запоминает состояние машины **V**\*.
2. Выбирает η*http://citforum.ru/security/articles/kazarin/e.gif*RZq и вычисляет h`1*http://citforum.ru/security/articles/kazarin/3r.gif*gη(modp) и h`2`*http://citforum.ru/security/articles/kazarin/3r.gif*rη(modp).
3. Получает от **V**\* значения a и b. Если δ*http://citforum.ru/security/articles/kazarin/nerav.gif*ragb(modp) , то MV` останавливается.
4. Машина MV` "отматывает" **V**\* на состояние, которое было запомнено в конце шага 1. Выбирает t*http://citforum.ru/security/articles/kazarin/e.gif*RZq и вычисляет h1*http://citforum.ru/security/articles/kazarin/3r.gif*ragb+1(modp) и h2*http://citforum.ru/security/articles/kazarin/3r.gif*γawb+1(modp).
5. Машина MV` передает **V**\* h1, h2 и получает ответ (a`,b`). Возможны два варианта:

5.1. a=a` , b=b` . В этом случае моделирование закончено и MV` записывает на выходную ленту тройку (h1,h2,t) и останавливается.  
5.2. a*http://citforum.ru/security/articles/kazarin/nerav.gif*a` или b*http://citforum.ru/security/articles/kazarin/nerav.gif*b`. Отсюда следует, что δhttp://citforum.ru/security/articles/kazarin/3r.gifragbhttp://citforum.ru/security/articles/kazarin/3r.gifra`gb`(modp) . Предположим, что b*http://citforum.ru/security/articles/kazarin/nerav.gif*b`. Из этого следует, что a*http://citforum.ru/security/articles/kazarin/nerav.gif*a`. Следовательно, MV` может вычислить r*http://citforum.ru/security/articles/kazarin/3r.gif*g(b-b`)/(a-a`)(modp). Отсюда(b`-b)/(a-a`)=l - дискретный логарифм r по основаниюg.

1. Машина MV` "отматывает" **V**\* на состояние, которое было за-полнено в начале шага 1. Получает от**V**\* значение δ.
2. Выбирает η*http://citforum.ru/security/articles/kazarin/e.gif*RZq вычисляет h1*http://citforum.ru/security/articles/kazarin/3r.gif*gη(modp) и h2*http://citforum.ru/security/articles/kazarin/3r.gif*wη(modp) и передает их V\*.
3. Получает от **V**\* значения a и b. Если δ*http://citforum.ru/security/articles/kazarin/nerav.gif*ragb(modp) , то MV` останавливается. В противном случае вычисляет t=[η-al-b](mod q), выдает (h1,h2,t) на выходную ленту и останавливается.

К пп. 7 и 8 необходимо сделать следующее пояснение. Поскольку logg(p)w*http://citforum.ru/security/articles/kazarin/3r.gif*logr(p)γ, из этого следует, чтоlogw(p)γ*http://citforum.ru/security/articles/kazarin/3r.gif*logg(p)r. Отсюда

h2*http://citforum.ru/security/articles/kazarin/3r.gif*wη*http://citforum.ru/security/articles/kazarin/3r.gif*wb+1wal*http://citforum.ru/security/articles/kazarin/3r.gif*wb+1γa(modp)

Из описания моделирующей машины MV` очевидно, что она работает за полиномиальное время. Величины (h1,h2,t) на шаге 5.1 выбираются в точности как в протоколе и поэтому имеют такое же распределение вероятностей. Кроме того, значения (h1,h2), выбираемые на шаге 7, имеют такое же распределение, как и в протоколе. Чтобы показать что и t имеет одинаковое распределение, достаточно заметить, что машина **V**\* не может по h1 и h2 определить, с кем она имеет дело - с **S** или MV`.Поэтому, даже если бы **V**\* могла каким-либо "хитрым" образом строить a и b с учетом (h1,h2), распределение вероятностей величин a и b в обоих случаях одинаковы. Но значение t определяется однозначно четверкой величин a, b, h1, h2, при условии выполнения проверки на шаге 5 протокола.

**Таблица 3.2.** Отвергающий протокол является протоколом доказательства с абсолютно нулевым разглашением.

**Доказательство.** Полнота протокола очевидна. Если абоненты **S** и **V** следуют протоколу, тогда абонент **V**всегда примет доказательства абонента **S**.

Для доказательства корректности прежде всего заметим, что если logg(p)w*http://citforum.ru/security/articles/kazarin/3r.gif*logr(p)γ, то a и b, выбираемые абонентом V на шаге 1, не несут в себе никакой информации о значении β. Поэтому, если S может "открыть" c, сгенерированное им на шаге 2, лишь единственным образом (то есть выдать на шаге 4 единственное значение R, соответствующее данному a), то проверка на шаге 5 будет выполнена с вероятностью 1/2 в одном цикле и с вероятностью 1/2l во всех l циклах.

Если же **S** может сгенерировать c таким образом, что с вероятностью, которая не является пренебрежимо малой, он может на шаге 4 "открыть" оба значения α, то есть найти R1 и R2 такие, что http://citforum.ru/security/articles/kazarin/66.gif и http://citforum.ru/security/articles/kazarin/67.gif, то алгоритм, который использует **S** для этой цели, можно использовать для вычисления дискретных логарифмов: loggd=R2-R1. Так как при случайном выборе значения d логарифмloggd может быть вычислен с вероятностью, которая не является пренебрежимо малой, это противоречит гипотезе о трудности вычисления дискретных логарифмов.

Далее доказывается, что отвергающий протокол является доказательством с абсолютно нулевым разглашением. Для этого необходимо для всякого возможного проверяющего **V**\* построить моделирующую машину MV`, которая создает на выходе (без участия **S**) такое же распределение случайных величин (в данном случае, c и R), какое возникает у **V**\* в результате выполнения протокола.

Моделирующая машина

Следующие шаги выполняются в цикле l раз.

1. Машина MV` запоминает состояние машины **V**\*.
2. Получает от **V**\* значения a, b и d.
3. Выбирает αhttp://citforum.ru/security/articles/kazarin/e.gifR{0,1}, Rhttp://citforum.ru/security/articles/kazarin/e.gifRZq и вычисляет c*http://citforum.ru/security/articles/kazarin/3r.gif*dagR(modp) . Посылает **V**\* значение c.
4. Получает от **V**\* значение e.
5. Проверяет, было ли "угадано" на шаге 2 значение α (это значение было "угадано", если a*http://citforum.ru/security/articles/kazarin/3r.gif*ge(modp), b*http://citforum.ru/security/articles/kazarin/3r.gif*we(modp) и α=0, либо α*http://citforum.ru/security/articles/kazarin/3r.gif*re(modp), b*http://citforum.ru/security/articles/kazarin/3r.gif*γe(modp) и α=1). Если да, то записывает на входную ленту значение (c,R). В противном случае "отматывает" **V**\* на то состояние, которое было запомнено на шаге 1, и переходит на шаг 2.

Легко видеть, что распределения случайных величин (c,R), возникающее в процессе выполнения протокола и создаваемые моделирующей машиной MV`, одинаковы, поскольку R в обоих случаях - чисто случайная величина, а величина c записывается на выходную ленту машины MV` только тогда, когда αсовпало с β.

Поскольку значение α выбирается машиной MV` на шаге 3 случайным образом, а c не дает **V**\* никакой информации о значении α, на каждой итерации α будет угадано с вероятностью 1/2. Отсюда следует, что машина MV` работает за полиномиальное в среднем время.

В работе  показано, как строить схемы конвертируемой и селективно конвертируемой подписи с верификацией по запросу на основе отечественного стандарта ГОСТ Р 34.10-94. В таких схемах открытие определенного секретного параметра некоторой схемы подписи с верификацией по запросу позволяет трансформировать последнюю в обычную схему цифровой подписи. При этом открытие секретного параметра в конвертируемой схеме подписи с верификацией по запросу дает возможность верифицировать все имеющиеся и сгенерированные в дальнейшем подписи, в то время как в селективно конвертируемых схемах подписи с верификацией по запросу можно верифицировать лишь какую-либо одну подпись.

Процедуры арбитража. Часто разработчиками схем цифровой подписи при их создании, наряду с процедурами генерации и верификации цифровой подписи, не приводятся процедуры арбитража, которые позволяют установить корректность/ некорректность подписи в случае возникновения споров между участниками схемы. Проблеме арбитража в литературе уделяется мало внимание еще по-видимому и потому, что разработка арбитражных процедур считается прерогативой конкретного пользователя системы.

В случае возникновения проблемы по поводу корректности подписи основные действия участников процесса должны сводиться к следующему:

1. АбонентV предъявляет арбитру сообщение и его подпись.
2. Арбитр требует предъявить секретный ключ подписи абоненту S. Если абонент S отказывается, тогда арбитр дает заключение, что подпись подлинная. (То есть, в данном случае возможное нарушение сводится к тому, что абонент отказывается признать представленную подпись своей).
3. Арбитр выбирает из открытого сертифицированного справочника открытый ключ абонента S и проверяет его соответствие секретному ключу подписи. При обнаружения несоответствия (например, по вине службы ведения такого справочника) арбитр признает подпись, предъявленную абонентом V, подлинной.
4. Арбитр проверяет корректность подписи при помощи проверочных соотношений, используемых как при генерации, так и при верификации данной подписи.

Одна из основных проблем при арбитраже связана с тем, что арбитр получает секретный ключ абонента S. В том случае, если арбитр является нечестным, то тогда никаких способов защиты пользователей системы от него не может быть в принципе. Помимо этого, следует также сказать о том, что желательно, чтобы пользователь генерировал свои секретные ключи самостоятельно, так как даже создание специального центра генерации ключей не может полностью обезопасить пользователей от слабых ключей и полностью исключает в данном случае возможность решения споров в суде.

Хэш-функции

Основные определения, обозначения и алгоритмы. Под термином "хэш-функция" понимается функция, отображающая электронные данные произвольной длины (иногда длина ограничена, но достаточно большим числом), в значения фиксированной длины. Последние часто называют хэш-кодами. Таким образом, у всякой хэш-функции h имеется большое количество коллизий, то есть пар значений x и y таких, что h(x)=h(y). Основное требование, предъявляемое криптографическими приложениями к хэш-функциям, состоит в отсутствии эффективных алгоритмов поиска коллизий. Хэш-функция, обладающая таким свойством, называется хэш-функцией, свободной от коллизий. Кроме того, хэш-функция должна быть односторонней, то есть функцией, по значению которой вычислительно трудно найти ее аргумент и, в то же время, функцией, для аргумента которой, вычислительно трудно найти другой аргумент, который давал бы то же самое значение функции.

Таким образом, хэш-функции вместе со схемами электронной цифровой подписи предназначены для решения задач обеспечения целостности и достоверности электронных данных. В прикладных компьютерных системах требуется применение так называемых криптографически стойких хэш-функций. Под термином "криптографически стойкая хэш-функция" понимается функция h, которая является односторонней и свободной от коллизий.

Введем следующие обозначения. Хэш-функция h обозначается как h(α) иh(α,β) для одного и двух аргументов соответственно. Хэш-код функции h обозначается как H. При этом H0=I обозначает начальное значение (вектор инициализации) хэш-функции. Под обозначением http://citforum.ru/security/articles/kazarin/plus.gif будет пониматься операция сложения по модулю 2 или логическая операция XOR (исключающая ИЛИ). Результат шифрования блока B блочным шифром на ключе k обозначается Ek(B).

Для лучшего понимания дальнейшего материала приведем небольшой пример построения хэш-функции. Предположим нам необходимо получить хэш-код для некоторой программы M. В качестве шифрующего преобразования в хэш-функции будут использоваться процедуры шифра DES с ключом k. Тогда, чтобы получить хэш-код H программы M при помощи хэш-функции h необходимо выполнить следующую итеративную операцию:

http://citforum.ru/security/articles/kazarin/68.gif

где код программы M разбит на n 64-битных блока. Хэш-кодом данной хэш-функции является значениеH=h(M,I)=Hn.

Стойкость (безопасность) хэш-функций. Один из основных методов криптоанализа хэш-функций заключается в проведении криптоаналитиком (противником) атаки, основанной на "парадоксе дня рождения", основные идеи которой, излагаются ниже. Для понимания сущности предлагаемого метода криптоанализа достаточно знать элементарную теорию вероятностей.

Атака, основанная на "парадоксе дня рождения", заключается в следующем. Пусть ahttp://citforum.ru/security/articles/kazarin/kn.gif предметов выбираются с возвращением из некоторого множества с мощностью n. Тогда вероятность того, что два из них окажутся одинаковыми, составляют http://citforum.ru/security/articles/kazarin/69.gif

Практически это означает, что в случайно подобранной группе из 24 человек вероятность наличия двух лиц с одним и тем же днем рождения составляет значение примерно 1/2. Этот старый и хорошо изученный парадокс и положен в основу криптоанализа хэш-функций.

Например, для криптоанализа хэш-функций, основанных на использовании криптоалгоритма DES, указанная атака может быть проведена следующим образом. Пусть n - мощность области хэш-кодов (для криптоалгоритма DES она равна 264). Предположим, что есть две программы m1 и m2. Первая программа достоверна, а для второй криптоаналитик пытается получить то же самое значение хэш-кода, выдав таким образом программу m1 за программу m2 (то есть криптоаналитик пытается получить коллизию). Для этого криптоаналитик подготавливает порядка http://citforum.ru/security/articles/kazarin/kn.gif различных, незначительно отличающихся версий m1 и m2 и для каждой из них вычисляет хэш-код. С высокой вероятностью криптоаналитику удается обнаружить пару версий m`1 и m`2, имеющих один и тот же хэш-код. В дальнейшем при использовании данного хэш-кода можно выдать программу m`2 вместо программы m`1, содержание которой близко к содержанию программы m1.

К основным методам предотвращения данной атаки можно отнести: увеличение длины получаемых хэш-кодов (увеличение мощности области хэш-кодов) и выполнение требования итерированности шифрующего преобразования.

Методы построения криптографически стойких хэш-функций. Практические методы построения хэш-функций можно условно разделить на три группы: методы построения хэш-функций на основе какого-либо алгоритма шифрования (пример, приведенный выше), методы построения хэш-функций на основе какой-либо известной вычислительно трудной математической задачи и методы построения хэш-функций "с нуля".

Рассмотрим примеры построения хэш-функций на основе алгоритмов шифрования. Наряду с примером, приведенным выше, покажем, как строить хэш-функции на основе наиболее известных блочных шифров ГОСТ 28147 - 89, DES и FEAL. В качестве шифрующего преобразования будут использоваться некоторые режимы шифров ГОСТ 28147-89, DES и FEAL с ключом k. Тогда, чтобы получить хэш-код H программы M при помощи хэш-функции h, необходимо выполнить следующую итеративную операцию (например, с использованием алгоритма ГОСТ 28147-89):

http://citforum.ru/security/articles/kazarin/70.gif

Таким образом, хэш-кодом данной хэш-функции является значение H=h(M,I)=Hn. При этом используется режим выработки имитовставки ГОСТ 28147-89, а шифрующее преобразование 64-битных блоков заключается в выполнении 16 циклов алгоритма шифрования в режиме простой замены.

Алгоритм ГОСТ 28147-89 в качестве базового используется в хэш-функции отечественного стандарта на функцию хэширования сообщений ГОСТ Р 34.11-94, являющегося основным практическим инструментом в компьютерных системах, требующих обеспечения достоверности и целостности электронных данных.

Алгоритм DES (в режиме CFB) можно использовать в качестве базового, например, в следующей хэш-функции (с получением хэш-кода H=h(M,I)=Hn):

http://citforum.ru/security/articles/kazarin/71.gif

Другие примеры хэш-функций на основе алгоритмов шифрования. N-хэш алгоритм разработан фирмой Nippon Telephone Telegraph в 1990 году. N-хэш алгоритм использует блоки длиной 128 битов, шифрующую функцию, аналогичную функции алгоритма шифрования FEAL, и вырабатывает блок хэш-кода длиной 128 битов. На вход пошаговой хэш-функции в качестве аргументов поступают очередной блок кодаMi длиной 128 битов и хэш-код Hi-1 предыдущего шага также размером 128 битов. При этом H0=I, а Hi=Ek(Mi,Hi-1)*http://citforum.ru/security/articles/kazarin/plus.gif*Mi*http://citforum.ru/security/articles/kazarin/plus.gif*Hi-1. Хэш-кодом всего кода программы объявляется хэш-код, получаемый в результате преобразования последнего блока текста.

Идея использовать алгоритм блочного шифрования, стойкость которого общеизвестна, для построения надежных схем хэширования выглядит естественной. Однако для некоторых таких хэш-функций возникает следующая проблема. Предлагаемые методы могут требовать задания некоторого ключа, на котором происходит шифрование. В дальнейшем этот ключ необходимо держать в секрете, ибо противник, зная этот ключ и значение хэш-кода, может выполнить процедуру в обратном направлении. Еще одной слабостью указанных выше схем хэширования является то, что размер хэш-кода совпадает с размером блока алгоритма шифрования. Чаще всего размер блока недостаточен для того, чтобы схема была стойкой против атаки на основе "парадокса дня рождения". Поэтому были предприняты попытки построения хэш-функций на базе блочного шифра с размером хэш-кода в k раз (как правило, k=2) большим, чем размер блока алгоритма шифрования.

В качестве примера можно привести хэш-функции MDC2 и MDC4 фирмы IBM. Данные хэш-функции используют блочный шифр (в оригинале DES) для получения хэш-кода, длина которого в два раза больше длины блока шифра. Алгоритм MDC2 работает несколько быстрее, чем MDC4, но представляется несколько менее стойким.

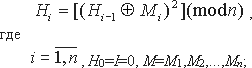
Следует также отметить, что существует два основных режима применения хэш-функций: поточный и блочный. Выбор режима зависит от используемого в хэш-функции шифрующего преобразования.

В качестве примера хэш-функций, построенных на основе вычислительно трудной математической задачи можно привести функцию из рекомендаций MKKTT X.509.

Криптографическая стойкость данной функции основана на сложности решения следующей труднорешаемой теоретико-числовой задачи. Задача умножения двух больших (длиной в несколько сотен битов) простых чисел является простой с вычислительной точки зрения, в то время как факторизация (разложение на простые множители) полученного произведения является труднорешаемой задачей для указанных размерностей.

Следует отметить, что задача разложения числа на простые множители эквивалентна следующей труднорешаемой математической задаче. Пусть n=pqпроизведение двух простых чисел p и q. В этом случае можно легко вычислить квадрат числа по модулюn: x2(mod n), однако вычислительно трудно извлечь квадратный корень по этому модулю.

Таким образом, хэш-функцию МККТТ X.509 можно записать как:



длина блока Mi представляется в октетах, каждый октет разбит пополам и к каждой половине спереди приписывается полуоктет, состоящий из двоичных единиц; n - произведение двух больших (512-битных) простых чисел p и q.

По-видимому, наиболее эффективными на сегодняшний день с точки зрения программной реализации и условий применения оказываются хэш-функции построенные "с нуля".

Алгоритм MD4 (Message Digest) был разработан Р. Ривестом. Размер вырабатываемого хэш-кода - 128 битов. По заявлениям самого разработчика при создании алгоритма он стремился достичь следующих целей:

После того, как алгоритм был впервые опубликован, несколько криптоаналитиков построили коллизии для последних двух из трех раундов, используемых в MD4. Несмотря на то, что ни один из предложенных методов построения коллизий не приводит к успеху для полного MD4, автор усилил алгоритм и предложил новую схему хэширования MD5.

Алгоритм MD5 является доработанной версией алгоритма MD4. Аналогично MD4, в алгоритме MD5 размер хэш-кода равен 128 битам.

После ряда начальных действий MD5 разбивает текст на блоки длиной 512 битов, которые, в свою очередь, делятся на 16 подблоков по 32 бита. Выходом алгоритма являются 4 блока по 32 бита, конкатенация которых образует 128-битовый хэш-код. В 1993 году Национальный институт стандартов и технологий (NIST) США совместно с Агентством национальной безопасности США выпустил "Стандарт стойкой хэш-функции" (Secure Hash Standard), частью которого является алгоритм SHA. Предложенная процедура вырабатывает хэш-код длиной 160 битов для произвольного текста длиной менее 264 битов. Разработчики считают, что для SHA невозможно предложить алгоритм, имеющий разумную трудоемкость, который строил бы два различных набора данных, дающих один и тот же хэш-код (то есть алгоритм, находящий коллизии). Алгоритм SHA основан на тех же самых принципах, которые использовал Р. Ривест при разработке MD4, более того, алгоритмическая структура SHA очень похожа на структуру MD4. Процедура дополнения хэшируемого текста до кратного 512 битам полностью совпадает с процедурой дополнения алгоритма MD5.

Обобщая вышесказанное, следует отметить, что при выборе практически стойких и высокоэффективных криптографических хэш-функций можно руководствоваться следующими эвристическими принципами:

* любой из известных алгоритмов построения коллизий не должен быть эффективнее метода, основанного на "парадоксе дня рождения";
* алгоритм должен допускать эффективную программную реализацию на 32-разрядном процессоре;
* алгоритм не должен использовать сложных структур данных и подпрограмм;
* алгоритм должен быть оптимизирован с точки зрения его реализации на микропроцессорах типа Intel.

Криптографические протоколы

Базовым объектом исследований в криптологии являются криптографические протоколы. Криптографические протоколы относится к сравнительно новому направлению в криптографии, которое в последнее время переживает бурное развитие. В то же время, они являются весьма нетривиальным объектом исследований. Даже их формализация на сегодняшний день выглядит весьма затруднительной. Тем не менее, неформально под протоколом будем понимать распределенный алгоритм, т.е. совокупность алгоритмов для каждого из участников вычислений, плюс спецификации форматов сообщений, пересылаемых между участниками, плюс спецификации синхронизации действий участников, плюс описание действий при возникновении сбоев.

Объектом изучения теории криптографических протоколов являются удаленные абоненты, взаимодействующие по открытым каналам связи. Целью взаимодействия абонентов является решение какой-то задачи. Имеется также противник, который преследует собственные цели. При этом противниками могут быть один или даже несколько абонентов, вступивших в сговор.

Приведенная выше схема подписи с верификацией по запросу, а также схемы, представленные в разделе 2.4. есть не что иное, как сложные двухсторонние или многосторонние протоколы, которые используют, в том числе и криптографические примитивы.