Лабораторная работа № 4

Однонаправленные хэш-функции. Электронная цифровая подпись.

*Цель работы:* Изучить различные алгоритмы однонаправленного хэширования данных, основанные на симметричных блочных алгоритмах шифрования. Ознакомиться со схемами цифровой подписи и получить навыки создания и проверки подлинности электронной цифровой подписи.­­

**Однонаправленные хэш-функции**

Однонаправленная функция H(M) применяется к сообщению произвольной длины и возвращает значение фиксированной длины h:

h = H (M),

где h имеет длину m.

Многие функции позволяют вычислять значение фиксированной длины по входным данным произвольной длины, но у однонаправленных хэш-функций есть дополнительные свойства, делающие их однонаправленными:

* зная M, легко вычислить h. Зная H, трудно определить M, для которого H(M) = h ;
* зная M, трудно определить другое сообщение , для которого H (M) = H ().

Смысл однонаправленных хэш-функций состоит в обеспечении для уникального идентификатора («отпечатка пальца»).

В некоторых приложениях однонаправленности недостаточно, необходимо, чтобы выполнялось другое требование, называемое **устойчивостью к столкновению:** должно быть трудно найти два случайных сообщения M и , для которых H (M) = H ().

Это возможно сделать методом дня рождения. Он основан не на поиске другого сообщения , для которого H (M) = H (), а на поиске двух случайных сообщений M и , для которых H (M) = H ().

Следующий протокол, впервые описанный Гидеоном Ювалом, показывает, как, если требование устойчивости к столкновению не выполняется, Алиса может использовать вскрытие методом дня рождения для обмана Боба:

1. Алиса готовит две версии контракта: одну, выгодную для Боба, и другую, приводящую его к банкротству.
2. Алиса вносит несколько незначительных изменений в каждый документ и вычисляет хэш-функции. (Этими изменениями могут быть действия, подобные следующим: замена «пробела» комбинацией «пробел» – «забой» – «пробел», вставка одного-двух «пробелов» перед возвратом каретки и т.д. Делая или не делая по одному изменению в каждой из 32 строк, Алиса может легко получить различных документа).
3. Алиса сравнивает хэш-значения для каждого изменения в каждом из двух документов, разыскивая пару, для которой эти значения совпадают. (Если выходом хэш-функции является всего лишь 64-разрядное значение, Алиса, как правило, сможет найти совпадающую пару, сравнив версий каждого документа). Она восстанавливает два документа, дающих одинаковое хэш-значение.
4. Алиса получает подписанную Бобом выгодную для него версию контракта, используя протокол, которым он подписывает только хэш-значения.
5. Спустя некоторое время, Алиса подменяет контракт, подписанный Бобом, другим, который он не подписывал. Теперь она может убедить арбитра в том, что Боб подписал другой контракт.

**Длина однонаправленных хэш-функций**

64-битные хэш-функции слишком малы, чтобы противостоять вскрытию методом дня рождения. Более практичны однонаправленные хэш-функции, выдающие 128-битные хэш-значения. При этом, чтобы найти два документа с одинаковыми хэш-значениями для вскрытия методом дня рождения, придётся хэшировать случайных документа, что недостаточно, если нужна длительная безопасность.

Для удлинения хэш-значений, выдаваемых конкретной хэш-функцией, был предложен следующий метод:

1. Для сообщения с помощью одной из однонаправленных хэш-функций генерируется хэш-значение.
2. Хэш-значение добавляется к сообщению.
3. Генерируется хэш-значение объединения сообщения и хэш-значения этапа 1.
4. Создаётся большее хэш-значение, состоящее из объединения хэш-значения этапа 1 и хэш-значения этапа 3.
5. Этапы 1–4 повторяются нужное количество раз для обеспечения требуемой длины хэш-значения.

**Обзор однонаправленных хэш-функций**

Нелегко построить функцию, вход которой имеет произвольный размер, а тем более сделать её однонаправленной. В реальном мире однонаправленные хэш-функции строятся на идее функции сжатия. Такая однонаправленная функция выдаёт хэш-значение длины n при заданных входных данных большей длины m. Входами функции сжатия являются блок сообщения и выход предыдущего блока текста (см. рисунок 1). Выход представляет собой хэш-значение всех блоков до этого момента, т.е. хэш-значение блока равно

Это хэш-значение вместе со следующим блоком сообщения становится следующим входом функции сжатия. Хэш-значением всего сообщения будет хэш-значение последнего блока.



Рис 1 – Однонаправленная функция

Хэшируемый вход должен каким-то способом содержать бинарное представление длины всего сообщения. Таким образом, преодолевается потенциальная проблема, вызванная тем, что сообщения различной длины могут давать одно и то же хэш-значение. Иногда такой метод называется **MD-усилением.**

В качестве однонаправленных хэш-функций можно использовать симметричные блочные алгоритмы шифрования. Самый очевидный способ – это шифрование сообщения в режиме CBC и CFB с помощью фиксированного ключа и IV, хэш-значением будет последний блок шифротекста.

Более хороший способ использует в качестве ключа блок сообщения, в качестве входа – предыдущее хэш-значение, а выходом служит текущее хэш-значение.

Действительные хэш-функции ещё сложнее. Размер блока обычно совпадает с длиной ключа, и размером хэш-значения будет длина блока. Т.к. большинство блочных алгоритмов 64-битные, то спроектирован ряд схем, дающих хэш-значение, в два раза большее длины блока.

При условии, что хэш-функция правильна, безопасность этой схемы основана на безопасности используемой блочной функции. Однако есть и исключения. Дифференциальный криптоанализ лучше работает против блочных функций в хэш-значениях, чем против блочных функций, используемых для шифрования: ключ известен, поэтому можно использовать различные приёмы. Для успеха нужна только одна правильная пара, и можно генерировать столько выбранного открытого текста, сколько нужно.

Полезной мерой для хэш-функций, основанных на блочных шифрах, является скорость хэширования или количество n-битовых блоков сообщения (n – это размер блока алгоритма), обрабатываемых при шифровании. Чем выше скорость хэширования, тем быстрее алгоритм.

**Схемы, в которых длина хэш-значения равна длине блока**

**Общая схема:**

, где – случайное начальное значение, задаваемое пользователем, или, например, длина сообщения.

где A, B и C могут быть либо , , ( ), либо константы (возможно, равные 0). – это некоторое случайное начальное число . Сообщение разбивается на обрабатываемые отдельно части в соответствии с размером блока :

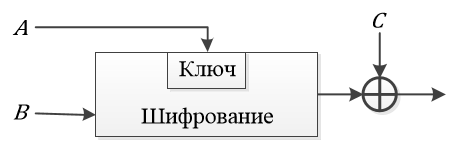


Рис 2 – Общая схема

Три различные переменные (A, B, C) могут принимать одно из четырёх возможных значений, поэтому всего существует 64 варианта схем этого типа.

Далее приведены четыре схемы безопасных хэш-функций:

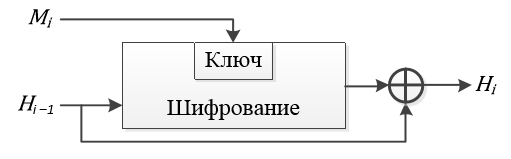


Рис 3 – Схема №1

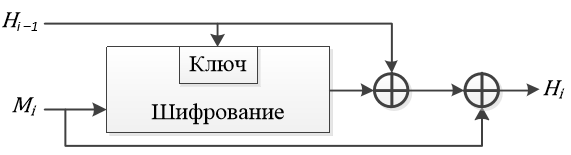


Рис 4 – Схема №2

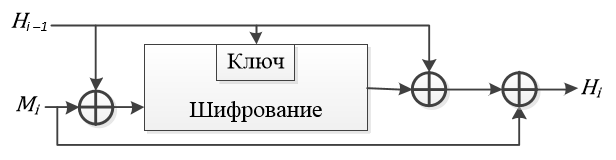


Рис 5 – Схема №3

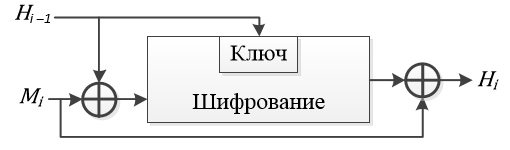


Рис 6 – Схема №4

**Схемы, в которых длина хэш-значения равна удвоенной длине блока**

**Схема Preneel-Bosselaers-Govaerts-Vandewalle**

При 64-битном блочном алгоритме схема выдаёт два 64-битных хэш-значения и , объединение которых даёт 128-битное хэш-значение. У большинства блочных алгоритмов длина блока равна 64 битам. Два соседних блока и , (размер каждого равен размеру блока) хэшируются вместе.

где и – два случайных начальных значения.

**Схема Quisquater-Girault**

Эта схема генерирует хэш-значение, в два раза большее длины блока. Она использует два хэш-значения и , и хэширует вместе два блока – и .

где и – два случайных начальных значения.

**Электронная цифровая подпись**

На протяжении многих веков при ведении деловой переписки, заключении контрактов и оформлении любых других важных бумаг подпись ответственного лица или исполнителя была непременным условием признания его статуса или неоспоримым свидетельством его важности. Подобный акт преследовал две цели:

* гарантирование истинности письма посредством сличения подписи с имеющимся образцом;
* гарантирование авторства документа (с юридической точки зрения).

Выполнение данных требований основывается на следующих свойствах подписи:

* подпись аутентична, т.е. с её помощью получателю документа можно доказать, что она принадлежит подписывающему;
* подпись служит доказательством, что только тот человек, чей автограф стоит на документе, мог подписать данный документ, и никто другой не смог бы этого сделать;
* подпись непереносима, т.е. она является частью документа, и поэтому перенести её на другой документ невозможно;
* документ с подписью является неизменяемым, т.е. после подписания его невозможно изменить, оставив данный факт незамеченным;
* подпись неоспорима, т.е. человек, подписавший документ, в случае признания экспертизой, что именно он засвидетельствовал данный документ, не может оспорить факт подписания;
* любое лицо, имеющее образец подписи, может удостовериться в том, что данный документ подписан владельцем подписи.

С переходом к безбумажным способам передачи и хранения данных, а также с развитием систем электронного перевода денежных средств, в основе которых – электронный аналог бумажного платёжного поручения, проблема виртуального подтверждения аутентичности документа приобрела особую остроту. Развитие любых подобных систем теперь немыслимо без существования электронных подписей под электронными документами. Однако применение и широкое распространение **электронно-цифровых** **подписей** (ЭЦП) повлекло целый ряд правовых проблем. Так, ЭЦП может применяться на основе договоренностей внутри какой-либо группы пользователей системы передачи данных, и в соответствии с договоренностью внутри данной группы ЭЦП должно иметь юридическую силу. Но будет ли электронная подпись иметь доказательную силу в суде, например, при оспаривании факта передачи платежного поручения?

Рассмотрим существующие схемы электронной цифровой подписи.

**Схема 1**

Данная схема предполагает шифрование электронного документа на основе симметричных алгоритмов и предусматривает наличие в системе третьего лица (арбитра), пользующегося доверием участников обмена. Взаимодействие пользователей данной системой производится по следующей схеме (см. рисунок 7):

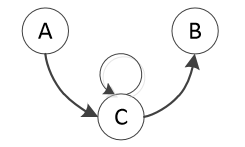


Рис 7 – Основные методы построения ЭЦП. Схема 1

Участник A зашифровывает сообщение своим секретным ключом , знание которого разделено с арбитром (C на рисунке 7), затем шифрованное сообщение передаётся арбитру с указанием адресата данного сообщения (информация, идентифицирующая адресата, передаётся также в зашифрованном виде).

Арбитр расшифровывает полученное сообщение ключом , производит необходимые проверки и затем зашифровывает его секретным ключом участника B (). Далее зашифрованное сообщение посылается участнику B вместе с информацией, что оно пришло от участника A.

Участник B расшифровывает данное сообщение и убеждается в том, что отправителем является участник A.

Авторизацией документа в данной схеме считается сам факт шифрования электронного документа секретным ключом и передачи зашифрованного электронного документа арбитру. Основным преимуществом этой схемы является наличие третьей стороны, исключающей какие-либо спорные вопросы между участниками информационного обмена, т.е. в данном случае не требуется дополнительной системы арбитража ЭЦП. Недостатком схемы являются необходимость участия в обмене информацией третьей стороны и использование симметричных алгоритмов шифрования. На практике эта схема не получила широкого распространения.

**Схема 2**

Фактом подписания документа в данной схеме (см. рисунок 8) служит шифрование документа секретным ключом его отправителя. Здесь используются асимметричные алгоритмы шифрования.

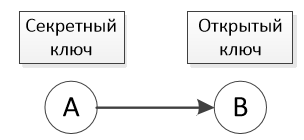


Рис 8 – Основные методы построения ЭЦП. Схема 2

Вторая схема используется довольно редко, поскольку длина электронного документа может оказаться очень большой (шифрование асимметричным алгоритмом может оказаться неэффективным по времени). Но в этом случае в принципе не требуется наличие третьей стороны, хотя она и может выступать в роли сертификационного органа открытых ключей пользователя.

**Схема 3**

Наиболее распространённая схема ЭЦП использует шифрование окончательного результата обработки электронного документа хэш-функцией при помощи асимметричного алгоритма. Структурная схема такого варианта построения ЭЦП представлена на рисунке 9:



Рис 9 – Основные методы построения ЭЦП. Схема 3

Процесс генерации ЭЦП происходит следующим образом.

Участник A вычисляет хэш-код от электронного документа. Полученный хэш-код проходит процедуру преобразования с использованием секретного ключа участника A. После этого полученное значение (которое и является ЭЦП) вместе с электронным документом отправляется участнику B.

Участник B должен получить электронный документ с ЭЦП и сертифицированный открытый ключ участника A, а затем произвести расшифрование на нём ЭЦП. Электронный документ подвергается операции хэширования, после чего результаты сравниваются, и если они совпадают, то ЭЦП признается истинной, в противном случае – ложной.

В настоящее время применяется несколько алгоритмов цифровой подписи:

* RSA (наиболее популярен);
* Digital Signature Algorithm, DSA (алгоритм цифровой подписи американского правительства, который применяют в стандарте цифровой подписи (Digital Signature Standard, DSS), также используется часто);
* алгоритм Эль-Гамаля (иногда можно встретить);
* алгоритм, который применяют в стандарте ГОСТ РЗ4.10-94 (в основе лежит DSA и является вариацией подписи Эль-Гамаля);
* так же существуют алгоритмы подписей, в основе которых лежит криптография эллиптических кривых; они похожи на все прочие, но в некоторых ситуациях работают эффективнее.

**Электронная подпись RSA**

Для осуществления подписи сообщения необходимо вычислить хэшфункцию которая ставит в соответствие сообщению M число m. На следующем шаге достаточно снабдить подписью только число m, и эта подпись будет относиться ко всему сообщению *M*[1, 2, 4].

Далее по алгоритму RSA вычисляются ключи *(e, n)* и *(d, n).*

Затем вычисляется (*d* – секретная степень).

Число *s* – это и есть цифровая подпись. Она просто добавляется к сообщению и получается подписанное сообщение *〈M, s〉.*

Теперь каждый, кто знает параметры подписавшего сообщение (т.е. числа *e* и *n*), может проверить подлинность подписи.

Для этого необходимо проверить выполнение равенства

**Алгоритм Эль-Гамаля**

Для генерации пары ключей сначала выбирается большое простое число *p*, один из его первообразных корней *g* и случайное число *x*(*g < p* , *x < p*). Затем вычисляется .

Открытым ключом являются *y*, *g* и *p*. Закрытым ключом является *x*.

Чтобы подписать *m*, являющееся хэш-значением некоторого сообщения *M*, сначала выбирается секретное случайное число *k*, взаимно простое с *p-1*. Затем вычисляется .

Из соотношения определяется *b*. Выполнив преобразования, получим где – определяется из соотношения –

В результате подписью будет пара (*a, b*). Для проверки подписи нужно убедиться, что

***Пример.***

Пусть *p = 11, g =2, x = 8.* Тогда y =

Выбираем *k = 9.* Тогда a =

Из соотношения находим обратный элемент , применяя расширенный алгоритм Евклида (см. далее):



Подписью хэш-значения *m = 5* является пара (*a,b)=(6,3).*

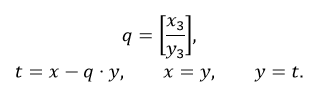
Проверка

**Нахождение обратного элемента с помощью расширенного алгоритма Евклида:**

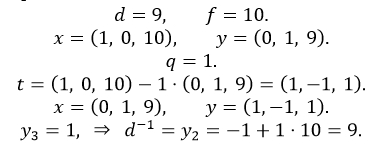
Пусть нужно найти элемент такой, что

Пусть *x = (1, 0, f), y = (0, 1, d).* В цикле выполняются следующие действия:

1. Если , то не существует элемента, обратного к d по модулю *f.*
2. Если , то
3. Иначе выполняются следующие преобразования, после которых выполняется переход на шаг 1:



**Пример** нахождения обратного элемента:

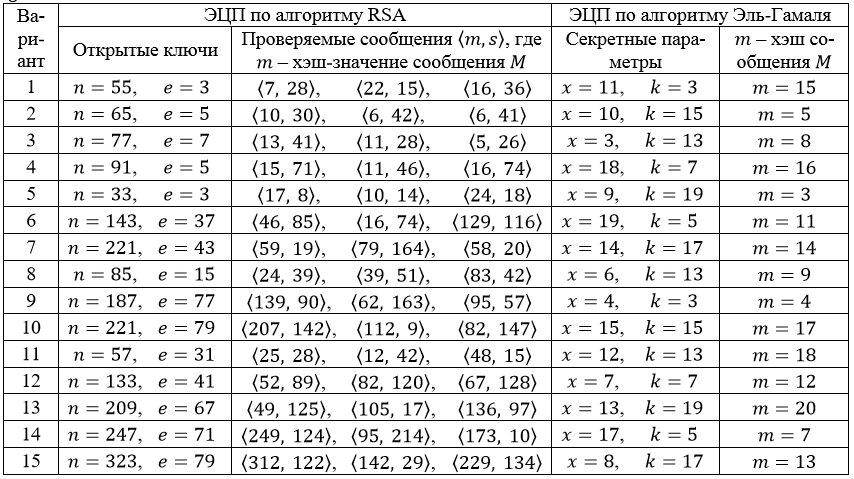


**Задание**

1. Реализовать приложение, позволяющее вычислять и проверять ЭЦП, сформированную по алгоритмам RSA и Эль-Гамаля.
2. С помощью реализованного приложения выполнить следующие задания:
   1. Протестировать правильность работы разработанного приложения.
   2. Для заданных в варианте открытых ключей пользователя проверить подлинность подписанных по алгоритму RSA хэш-значений *m* некоторых сообщений *M*.
   3. Абоненты некоторой сети применяют подпись Эль-Гамаля с известными общими параметрами *p* и *g*. Для указанных в варианте секретных параметров абонентов найти открытый ключ и построить подпись для хэш-значения *m* некоторого сообщения *M*. Проверить правильность подписи

**Варианты**

Для построения подписи Эль-Гамаля следует использовать открытые параметры *p = 23*, *g = 5*.



**Контрольные вопросы:**

1. Что такое хэш-функция, для чего она используется? В чём заключается устойчивость к столкновениям?
2. Как обмануть подписчика, если требование устойчивости к столкновению не выполняется?
3. Схемы хэширования с длиной хэш-значения, равной длине блока.
4. Схемы хэширования с длиной хэш-значения, равной удвоенной длине блока.
5. Для чего нужна цифровая подпись? Основные свойства цифровой подписи.
6. Какие схемы цифровой подписи существуют? Какая схема самая распространенная и почему?
7. Как осуществляется подпись RSA? В чем отличие подписи RSA от алгоритма шифрования RSA?
8. Как осуществляются подпись и проверка на подлинность подписи по алгоритму Эль-Гамаля?