УДК 004.056.55

М.А.Смирнова[1], И.А.Самойлова[1], Л.В.Устинова[2]

[1]Карагандинский государственный университет им. Е.А.Букетова;
[2]Назарбаев Интеллектуальная Школа химико-биологического направления, Караганда
(E -mail: smirnova_marina_alex@mail.ru)

Примеры реализации асимметричных криптоалгоритмов

В статье рассмотрены вопросы криптографических средств защиты информации. Показаны примеры реализации асимметричных криптоалгоритмов. Подробно исследованы технологии электронно-цифровых подписей. Изложены условия асимметричной криптографии. Приведены конкретные примеры алгоритмов, устраняющих избыточность записи данных. Проанализированы алгоритмы сжатия данных и архивации.

Ключевые слова: информационная безопасность, алгоритм, электронно-цифровая подпись, шифрование, дешифрование, кодирование, криптосхема.

Асимметричная криптография первоначально была задумана как средство передачи сообщений от одного объекта к другому, а не для конфиденциального хранения информации, обеспечиваемого исключительно симметричными алгоритмами. Вследствие этого введем следующие термины: «отправитель» — это лицо, которое шифрует и отправляет информацию по незащищенному каналу; «получатель» — это лицо, принимающее и восстанавливающее информацию в ее исходном виде.

Главная идея асимметричных криптоалгоритмов заключается в использовании двух ключей: для шифрования сообщения применяют один ключ, а при дешифровании — другой.

Вторым необходимым условием асимметричной криптографии является необратимость процедуры шифрования, даже по известному ключу шифрования. То есть при наличии ключа шифрования и зашифрованного текста отсутствует возможность восстановления исходного сообщения. Расшифровка текста возможна только при наличии второго ключа — ключа дешифрования. Следовательно, ключ шифрования для отправки сообщений можно вообще не скрывать. Этот ключ шифрования носит название «открытого ключа». А получателю сообщений ключ дешифрования, называемый «закрытым ключом», нужно держать в секрете. Третьим необходимым условием асимметричной криптографии является условие того, что знание открытого ключа не даст возможности вычислить закрытый ключ.

Рассмотрим систему переписки при использовании асимметричного шифрования. Для каждого из N абонентов создается своя пара ключей: «открытый» E_j и «закрытый» D_j, где j — номер абонента. Все открытые ключи доступны всем пользователям. Каждый закрытый ключ известен только владельцу. Например, абонент № 2, передавая информацию абоненту № 7, использует ключ для шифрования E_7. Несмотря на доступность к ключу E_7 и каналу, расшифровка текста будет невозможна, так как процедура шифрования необратима по открытому ключу. Восстановление сообщения возможно только при помощи ключа D_7. Если сообщение пересылается в обратном направлении (от абонента 7 к абоненту 2), то применяются ключи E_2, D_2.

Следовательно, во-первых, в асимметричных системах связь количества существующих ключей с количеством абонентов линейная (N пользователей, $2*N$ ключей), а не квадратичная подобно симметричным системам.

Во-вторых, если нарушена конфиденциальность k-ой рабочей станции, злоумышленник может узнать только ключ D_k. Что дает ему возможность прочитывать все сообщения, поступающие к абоненту № k, но не дает возможность выдавать себя за него при отправке писем.

Алгоритм RSA. Первой наиболее известной системой электронно-цифровой подписи (ЭЦП) стала RSA, алгоритм которой был разработан тремя исследователями-математиками Рональдом Ривестом, Ади Шамиром и Леонардом Адльманом в 1977–78 гг.

Обобщенная схема формирования и проверки цифровой подписи RSA показана на рисунке 1.

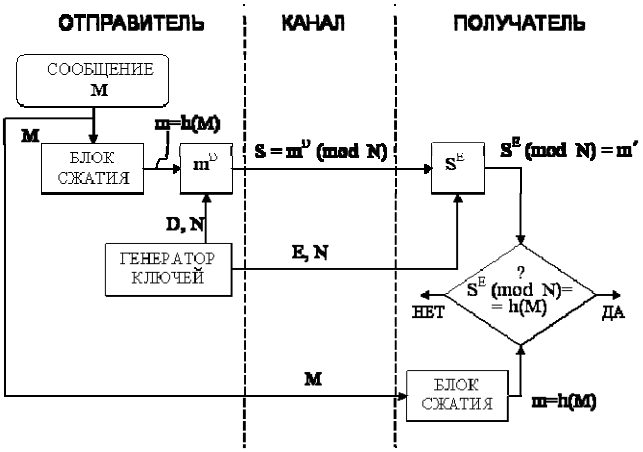


Рисунок 1. Обобщённая схема цифровой подписи RSA

Первый этап любого асимметричного алгоритма заключается в создании пары ключей, открытого и закрытого, и в распространении открытого ключа. Для алгоритма RSA этап создания ключей заключается в выполнении следующих операций [1]:

1. Выбираются два простых числа p и q.
2. Вычисляется $n = p \cdot q$.
3. Выбирается число e, взаимно простое, с числом $(p-1)\cdot(q-1)$.
4. Уравнение $e \cdot d + (p-1)(q-1) \cdot y = 1$ решается в целых числах методом Евклида, где d, y — неизвестные переменные.
5. Пара (e,n) публикуется как открытый ключ.
6. Число d — закрытый ключ, предназначенный для чтения сообщений, зашифрованных с помощью пары чисел (e,n).

Рассмотрим процесс шифрования при помощи этих чисел:

1. Отправитель разбивает свое сообщение на блоки, равные $k = trunc \cdot (\log_2(n))$ бит.
2. Такой блок можно представить, как число m_i из диапазона $(0; 2^k - 1)$. Для всех таких чисел можно вычислить выражение $c_i = ((m_i)^e) \cdot \bmod n$. Блоки c_i и составляют зашифрованное сообщение. Эти блоки можно передавать по открытому каналу, так как операция возведения в степень по модулю простого числа представляет собой необратимую математическую задачу. Обратная ей задача называется «логарифмированием в конечном поле» и является на несколько порядков более сложной задачей. Таким образом, даже если злоумышленник знает числа e и n, то по c_i прочесть исходные сообщения m_i он может, лишь применив только полный перебор m_i.

В соответствии с теоремой Эйлера, если число n представимо в виде двух простых чисел p и q, то для любого x имеет место равенство $x^{(p-1)(q-1)} \cdot \bmod n = 1$.

Следовательно, на приемной стороне для дешифрования используем число d. Для дешифрования RSA сообщений применим эту формулу. Возведем обе ее части в степень и умножим на x

$$\frac{y}{x^{-y\cdot(p-1)\cdot(q-1)} \bmod n} = 1^{-y} = 1;$$

$$\frac{x}{x^{-y\cdot(p-1)(q-1)+1} \bmod n} = 1 \cdot x = x.$$

Вообще говоря, для современных процессоров операции возведения в степень больших чисел являются достаточно трудоемкими, даже если они совершаются по алгоритмам, которые оптимизированы по времени. Поэтому, как правило, текст сообщения кодируется блочным шифром с исполь-

зованием ключа сеанса, а сам ключ сеанса шифруется асимметричным алгоритмом при помощи открытого ключа получателя и помещается в начало файла.

Технологии цифровых подписей. Еще одна проблема информационной безопасности — проверка подлинности автора сообщения — может быть красиво решена при помощи теории асимметричного шифрования. Если решать эту проблему при помощи симметричной криптографии, то необходимо применять очень трудоемкую и сложную схему. В то время как с помощью, например, того же алгоритма RSA достаточно просто создается алгоритм проверки подлинности автора и неизменности сообщения.

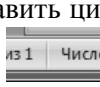
Допустим, необходимо передать текст по незащищенному каналу, который будет защищен при передаче от модификации. В этом случае для передаваемого текста вычисляют хеш-функцию. Можно подобрать другой текст, дающий такое же значение хеш-функции, но изменить в тексте десять-двадцать байт таким образом, чтобы текст остался полностью осмысленным, да еще и изменился в выгодную сторону (например, уменьшение суммы к оплате в три раза), — крайне сложно. Для того чтобы устранить такую возможность, хеш-функции создаются такими же сложными, как и криптоалгоритмы. Если методом полного перебора можно подобрать текст с таким же значением хеш-функции, а множество значений будет составлять, как и для блочных шифров, 2^{32}–2^{128} возможных вариантов, то для расшифровки подобного текста необходимы миллионы лет.

Следовательно, если мы передаем получателю защищенную от изменения методом хэш-сумму от пересылаемого текста, то он всегда сможет самостоятельно вычислить хэш-функцию от текста уже на приемной стороне и сверить ее с присланной нами. Если хотя бы один бит в контрольной сумме текста не совпадает с соответствующим битом в полученном хэш-значении, значит, текст во время пересылки подвергался несанкционированному изменению.

Разобьем теперь хэш-сумму, готовую к передаче, на несколько k-битных блоков h_i, где k — это размер сообщений по алгоритму RSA. Вычислим над каждым блоком значение $s_i = ((h_i)^d) \cdot \mathrm{mod} \cdot n$, где d — закрытый ключ отправителя. Теперь сообщение, состоящее из блоков s_i, можно передавать по сети. Опасность найти по известным h_i и s_i секретный ключ почти нулевая. При этом любой получатель сообщения может прочесть исходное значение h_i, выполнив операцию $((s_i)^e) \cdot \mathrm{mod} \cdot n = ((h_i)^{d*e}) \cdot \mathrm{mod} \cdot n = h_i$ — открытый ключ (e,n). Не зная закрытого ключа d, нельзя изменить текст, а значит и хэш-сумму, и нельзя вычислить такие s'_i, чтобы при их возведении в степень e получилась хэш-сумма h'_i, совпадающая с хэш-суммой фальсифицированного текста.

Следовательно, действия с хэш-суммой текста представляют «асимметричное шифрование наоборот»: при отправке используется закрытый ключ отправителя, для проверки сообщения — открытый ключ отправителя. Описанная технология называется *электронной подписью*. Информацией, уникально идентифицирующей отправителя, является закрытый ключ d (виртуальная подпись). Без доступа к этой информации невозможно создать такую пару (*текст,* s_i), чтобы изложенный ранее алгоритм проверки принес бы положительный результат.

Поменять таким образом местами открытый и закрытый ключи для создания из процедуры асимметричного шифрования алгоритма электронной подписи можно лишь в таких системах, в которых выполняется свойство коммутативности ключей.

Приведем пример использования ЭЦП в документе MSWord. Для добавления ЭЦП выполняется команда: кнопка Microsoft Office — Подготовка — Добавить цифровую подпись. В статусной строке отобразится значок «Этот документ содержит подписи» 

. Для получения сведений об ЭЦП необходимо выполнить щелчок по иконке ЭЦП. При изменении документа подпись становится недействительной (рис. 2).

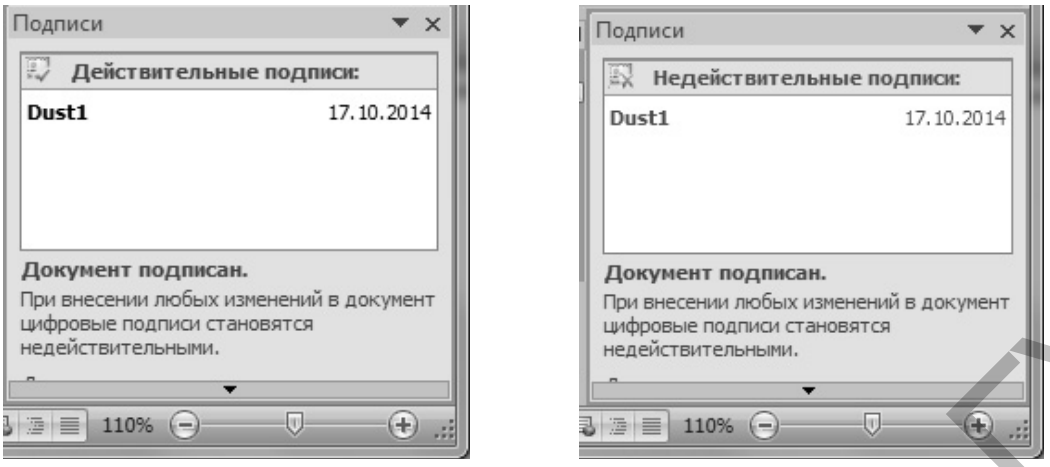


Рисунок 2. Результат проверки ЭЦП

Общие принципы архивации. Классификация методов

Большинство современных форматов (видео, аудио) занимают неоправданно большой объем, который на самом деле не требуется для их хранения. Алгоритмы, устраняющие избыточность записи данных, носят название алгоритмов сжатия данных, или алгоритмов архивации.

Все алгоритмы сжатия данных подразделяются на две группы: сжатие с потерями и без потерь. При использовании алгоритма сжатия без потерь данные на приемной стороне восстанавливаются в исходном виде.

Алгоритмы сжатия с потерями удаляют ту часть информации, которая может незначительно влиять на суть данных или не воспринимаемую человеком (для аудио- и видеоизображений). В криптосистемах применяется лишь первая группа алгоритмов.

Рассмотрим два основных алгоритма архивации без потерь:

– алгоритм Хаффмана, ориентированный на сжатие последовательностей байт, которые не связаны между собой;
– алгоритм Лемпеля-Зива (LZ77), ориентированный на сжатие любых видов текстов, который использует факт повторения последовательностей байт.

Большинство известных и часто используемых программ архивации без потерь (ARJ, RAR, ZIP и т.п.) применяют объединение этих двух алгоритмов.

В основе алгоритма Хаффмана лежит тот факт, что некоторые символы в тексте встречаются чаще средней величины повтора. Значит, если для записи распространенных символов применять короткие последовательности бит длиной меньше 8, а для записи редких символов — длинные, то объем файла значительно уменьшится.

Хаффман предложил простой алгоритм сопоставления кодирования символов кодом с целью получения файла длины, которая очень близка к его энтропии.

Кодирование осуществляется в два этапа:

1. Составление таблицы частот символов исходного текста.
2. Строится дерево кодирования Хаффмана.

Символы созданной таблицы записываются в вершины графа. Выбираются два символа (узел) с наименьшим весом (частота) в тексте. Создается родитель выбранных узлов (потомков) с весом, равным их сумме. Родитель добавляется в список свободных узлов, его два потомка удаляются из списка. Операция объединения вершин (2.2–2.4) повторяется до тех пор, пока не останется одна вершина. Значение вершины равняется длине кодируемого файла.

Дугам, выходящим из родителя, ставится в соответствие 0 или 1.

Для определения кода каждого символа необходимо пройти от вершины дерева до нее, записывая 0 и 1 по маршруту следования.

Пример. Закодируем методом Хаффмана слово «Барабан» с учетом таблицы 1.

Таблица 1

Таблица частот символов

Символ	Н	Р	Б	А
Частота	7	10	15	20

Строим дерево по описанной выше схеме (рис. 3). Для автоматизации построения дерева Хаффмана можно использовать визуализатор «Haffman» [2].

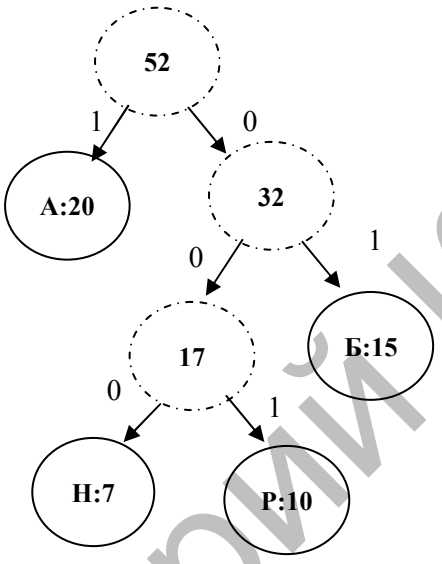


Рисунок 3. Код Хаффмана

Символ Р кодируется как 001, символ Н — 000, символ Б — 01, символ Н кодируется как 1. Коды символов представлены в таблице 2.

Код Хаффмана является префиксным, т.е. ни один из кодов символа не является началом кода другого символа. Следовательно, код Хаффмана однозначно восстанавливается получателем без дополнительной информации, например, при неизвестной длине кода каждого переданного символа.

Таблица 2

Таблица кодов слова «Барабан»

Символ	Р	Н	Б	А
Код	001	000	01	1

При приеме «0110011011000» отделяем первый символ «Б»: «**01**–10011011000», затем снова, начиная с вершины дерева, отделяем второй символ — «А» «01–**1**-0011011000», далее аналогично декодируется вся строка «01–1-001–1-01–1-000» «Барабан».

Алгоритм Лемпеля-Зива. Классический *алгоритм Лемпеля-Зива* (LZ77) формулируется так: «Если в прошедшем ранее выходном потоке уже встречалась подобная последовательность байт, причем запись о ее длине и смещении от текущей позиции короче, чем сама эта последовательность, то в выходной файл записывается ссылка (смещение, длина), а не сама последовательность» [3].

Пример кодирования строки «красная_строка» представлен на рисунке 4.

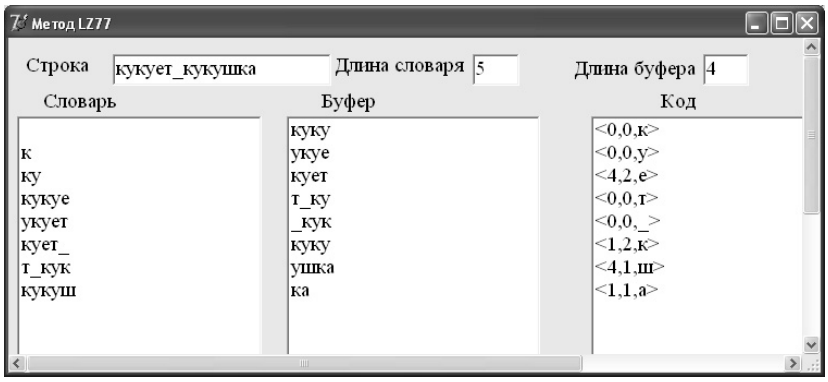


Рисунок 4. Кодирование методом LZ77

Длина закодированного сообщения методом LZ77 определяется как произведение количества кодов (N) на сумму длины двоичного кода:

$$L = N \cdot (L_{Subst} + L_{Smesh} + L_c),$$

где L_{Subst} — длина двоичного кода подстроки; L_{Smesh} — длина кода смещения; L_c = 8 бит — длина кодируемого символа (ASCII).

Длина подстроки не превышает размер буфера, а смещение меньше размера словаря. Следовательно,

$$L_{Subst} = \log_2(\textit{размер словаря}),$$

$$L_{Smesh} = \log_2(\textit{размер буфера} + 1).$$

С учетом того, что символ кодируется 8 битами, определим длину кода

$$((\log_2(5) + \log_2(4+1) + 8) \cdot 8 \approx (3+3+8) \cdot 8 = 126\ \textit{бит}.$$

Пример. Закодируем методом LZ77 строку «математика_информатика», длина словаря — 8, длина буфера — 4.

Результаты кодирования методом LZ77 словаря, буфера и кодирования представлены в таблице 3.

Таблица 3

Результат кодирования методом LZ77

Словарь	Буфер	Код
	МАТЕ	<0,0, М>
М	АТЕМ	<0,0, А>
МА	ТЕМА	<0,0, Т>
МАТ	ЕМАТ	<0,0, Е>
МАТЕ	МАТИ	<5,3, И>
МАТЕМАТИ	КА_И	<0,0, К>
АТЕМАТИК	А_ИН	<1,1,_>
ЕМАТИКА_	ИНФО	<5,1, Н>
АТИКА_ИН	ФОРМ	<0,0, Ф>
ТИКА_ИНФ	ОРМА	<0,0, О>
ИКА_ИНФО	РМАТ	<0,0, Р>
КА_ИНФОР	МАТИ	<0,0, М>
А_ИНФОРМ	АТИК	<1,1, Т>
ИНФОРМАТ	ИКА	<1,1, К>
ФОРМАТИК	А	<0,0, А>

Недостатки кодирования методом LZ77:

– скорость кодирования обратно пропорциональна размеру словаря;

– кодирование одиночных символов неэффективно.

Для кодирования последовательности одинаковых символов применятся метод сжатия RLE, который является подклассом алгоритма LZ77 и заключается в записи вместо одного символа их

количества. Например, результатом кодирования последовательности «AABBBAA» методом RLE является строка «(A,2), (B,3), (A,2)».

Применение алгоритмов архивации приводит к устранению избыточности и уменьшает вероятность взлома криптосхемы.

Список литературы

1 *Мусиралиева Ш.Ж.* Прикладная криптография: Учеб.пособие. — Алматы: Print-S, 2004. — 73 с.

2 *Безбогов А.А., Яковлев А.В., Шамкин В.Н.* Методы и средства защиты компьютерной информации: Учеб. пособие. — Тамбов: Изд-во ТГТУ, 2006. — 196 с.

3 *Ватолин Д., Ратушняк А., Смирнов М.* Методы сжатия данных: Учеб.-справоч. лит. — М.: ДИАЛОГ-МИФИ, 2003. — 381 с.

М.А.Смирнова, И.А.Самойлова, Л.В.Устинова

Ассиметриялық криптоалгоритмдердің жүзеге асырылу мысалдары

Мақалада криптографикалық ақпаратты қорғаудың құралдары туралы сұрақтар қарастырылған. Асимметриялық криптоалгоритмдердің жүзеге асырылу мысалдары көрсетілген. Электронды-цифрлық қолтаңбалар технологиялардың мағыналары толық ашылған. Ассиметриялық криптографияның шарттары зерттелген. Берілгендерді сығу, архивация алгоримдері жан-жақты талданған.

M.A.Smirnova, I.A.Samoilova, L.V.Ustinova

Examples of implementation of asymmetric encryption algorithms

The article deals with cryptographic information security. Showing examples of asymmetric encryption algorithms. Details disclosed technology of electronic digital signatures. It sets out the conditions of asymmetric cryptography. Concrete examples of algorithms that eliminate redundant data recording. Analyzed data compression algorithms, algorithms for backup.

References

1 Musiralieva Sh.Zh. *Applied Cryptography*: textbooks, Almaty: Print-S, 2004, 73 p.

2 Bezbogov A.A., Yakovlev A.V., Shamkin V.N. *Methods and tools for the protection of computer information*: Textbook, Tambov: Publ. TSTU, 2006, 196 p.

3 Vatolin D., Ratushnyak A. Smirnov M. *Methods of data compression*: training and reference literature, Moscow: Dialog-MIFI, 2003, 381 p.