Сергей Баричев

КРИПТОГРАФИЯ БЕЗ СЕКРЕТОВ

Содержание

От автора

Эта книга - краткое введение в криптографию. С одной стороны, здесь изложен материал, который отвечает на многие вопросы, которые возникают у тех кто делает на ниве этой науке первые шаг, с другой стороны здесь есть тот минимум информации, который достаточен для того чтобы самостоятельно оценивать любые реальные криптосистемы или даже создавать свои собственные.

Язык книги делался по возможности доступным, но не освобождает Читателя от необходимости владения элементарными основами математики, в частности алгебры и теории групп и полей.

Многие вопросы к сожалению остались за обложками этой книги. В частности после долгих сомнений Автор решил отказаться от рассмотрения DES, ввиду его крайней непрактичности и неуживчивости на российской почве[1].

Массу полезной информации можно найти на сервере `ftp.rsa.com`. В faq5.doc Вы если и не найдете ответ на любой вопрос по криптографии, то обнаружите большое количество ссылок на другие источники.

Автор будет признателен за любые замечания и вопросы, которые проще всего направить по адресу: bar@glasnet.ru

Баричев Сергей

[1] IMHO

Введение

Проблема защиты информации путем ее преобразования, исключающего ее прочтение посторонним лицом волновала человеческий ум с давних времен. История криптографии - ровесница истории человеческого языка. Более того, первоначально письменность сама по себе была криптографической системой, так как в древних обществах ею владели только избранные. Священные книги Древнего Египта, Древней Индии тому примеры.

С широким распространением письменности криптография стала формироваться как самостоятельная наука. Первые криптосистемы встречаются уже в начале нашей эры. Так, Цезарь в своей переписке использовал уже более менее систематический шифр, получивший его имя.

Бурное развитие криптографические системы получили в годы первой и второй мировых войн. Начиная с послевоенного времени и по нынешний день появление вычислительных средств ускорило разработку и совершенствование криптографических методов.

Почему проблема использования криптографических методов в информационных системах (ИС) стала в настоящий момент особо актуальна?

С одной стороны, расширилось использование компьютерных сетей, в частности глобальной сети Интернет, по которым передаются большие объемы информации государственного, военного, коммерческого и частного характера, не допускающего возможность доступа к ней посторонних лиц.

С другой стороны, появление новых мощных компьютеров, технологий сетевых и нейронных вычислений сделало возможным дискредитацию криптографических систем еще недавно считавшихся практически не раскрываемыми.

Проблемой защиты информации путем ее преобразования занимается *криптология* (*kr*yptos - тайный, l*og*o*s* - наука). Криптология разделяется на два направления - *криптографию* и *криптоанализ*. Цели этих направлений прямо противоположны.

Криптография занимается поиском и исследованием математических методов преобразования информации.

Сфера интересов *криптоанализа* - исследование возможности расшифровывания информации без знания ключей.

В этой книге основное внимание будет уделено криптографическим методам.

Современная криптография включает в себя четыре крупных раздела:

1. Симметричные криптосистемы.
2. Криптосистемы с открытым ключом.
3. Системы электронной подписи.
4. Управление ключами.

Основные направления использования криптографических методов - передача конфиденциальной информации по каналам связи (например, электронная почта), установление подлинности передаваемых сообщений, хранение информации (документов, баз данных) на носителях в зашифрованном виде.

Терминология

Итак, криптография дает возможность преобразовать информацию таким образом, что ее прочтение (восстановление) возможно только при знании ключа.

В качестве информации, подлежащей шифрованию и дешифрованию, будут рассматриваться *тексты*, построенные на некотором *алфавите*. Под этими терминами понимается следующее.

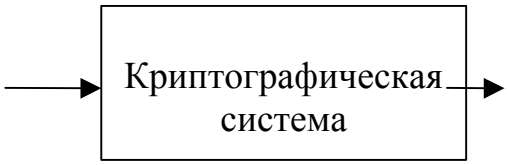
Алфавит - конечное множество используемых для кодирования информации знаков.

Текст - упорядоченный набор из элементов алфавита.

В качестве примеров алфавитов, используемых в современных ИС можно привести следующие:

- алфавит Z_{33} - 32 буквы русского алфавита и пробел;
- алфавит Z_{256} - символы, входящие в стандартные коды ASCII и КОИ-8;
- бинарный алфавит - $Z_2 = \{0,1\}$;
- восьмеричный алфавит или шестнадцатеричный алфавит;

Шифрование - преобразовательный процесс: *исходный текст*, который носит также название *открытого текста*, заменяется *шифрованным текстом*.



Дешифрование - обратный шифрованию процесс. На основе ключа шифрованный текст преобразуется в исходный.

Ключ - информация, необходимая для беспрепятственного шифрования и дешифрования текстов.

Криптографическая система представляет собой семейство **T** преобразований открытого текста. Члены этого семейства индексируются, или обозначаются символом *k*; параметр *k* является *ключом*. Пространство ключей *K* - это набор возможных значений ключа. Обычно ключ представляет собой последовательный ряд букв алфавита.

Криптосистемы разделяются на *симметричные* и *с открытым ключом*.

В *симметричных криптосистемах* и для шифрования, и для дешифрования используется *один и тот же ключ*.

В *системах с открытым ключом* используются два ключа - *открытый* и *закрытый*, которые математически связаны друг с другом. Информация шифруется с помощью открытого ключа, который доступен всем желающим, а расшифровывается с помощью закрытого ключа, известного только получателю сообщения.

Термины *распределение ключей* и *управление ключами* относятся к процессам системы обработки информации, содержанием которых является составление и распределение ключей между пользователями.

Электронной (цифровой) подписью называется присоединяемое к тексту его криптографическое преобразование, которое позволяет при получении текста другим пользователем проверить авторство и подлинность сообщения.

Криптостойкостью называется характеристика шифра, определяющая его стойкость к дешифрованию без знания ключа (т.е. криптоанализу). Имеется несколько показателей криптостойкости, среди которых:

- количество всех возможных ключей;
- среднее время, необходимое для криптоанализа.

Преобразование T_k определяется соответствующим алгоритмом и значением параметра *k*. Эффективность шифрования с целью защиты информации зависит от сохранения тайны ключа и криптостойкости шифра.

Требования к криптосистемам

Процесс криптографического закрытия данных может осуществляться как программно, так и аппаратно. Аппаратная реализация отличается существенно большей стоимостью, однако ей присущи и преимущества: высокая производительность, простота, защищенность и т.д. Программная реализация более практична, допускает известную гибкость в использовании.

Для современных криптографических систем защиты информации сформулированы следующие общепринятые требования:

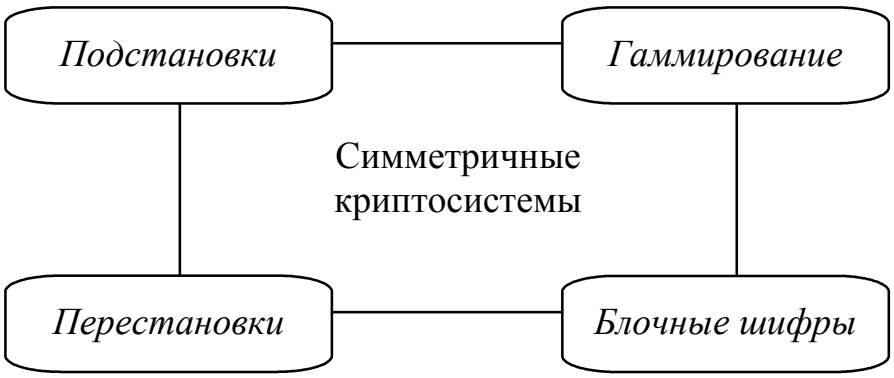
- зашифрованное сообщение должно поддаваться чтению только при наличии ключа;
- число операций, необходимых для определения использованного ключа шифрования по фрагменту шифрованного сообщения и соответствующего

ему открытого текста, должно быть не меньше общего числа возможных ключей;

- число операций, необходимых для расшифровывания информации путем перебора всевозможных ключей должно иметь строгую нижнюю оценку и выходить за пределы возможностей современных компьютеров (с учетом возможности использования сетевых вычислений);
- знание алгоритма шифрования не должно влиять на надежность защиты;
- незначительное изменение ключа должно приводить к существенному изменению вида зашифрованного сообщения даже при использовании одного и того же ключа;
- структурные элементы алгоритма шифрования должны быть неизменными;
- дополнительные биты, вводимые в сообщение в процессе шифрования, должен быть полностью и надежно скрыты в шифрованном тексте;
- длина шифрованного текста должна быть равной длине исходного текста;
- не должно быть простых и легко устанавливаемых зависимостью между ключами, последовательно используемыми в процессе шифрования;
- любой ключ из множества возможных должен обеспечивать надежную защиту информации;
- алгоритм должен допускать как программную, так и аппаратную реализацию, при этом изменение длины ключа не должно вести к качественному ухудшению алгоритма шифрования.

Симметричные криптосистемы

Все многообразие существующих криптографических методов можно свести к следующим классам преобразований:



Моно- и многоалфавитные подстановки.

Наиболее простой вид преобразований, заключающийся в замене символов исходного текста на другие (того же алфавита) по более или менее сложному правилу. Для обеспечения высокой криптостойкости требуется использование больших ключей.

Перестановки.

Также несложный метод криптографического преобразования. Используется как правило в сочетании с другими методами.

Гаммирование.

Этот метод заключается в наложении на исходный текст некоторой псевдослучайной последовательности, генерируемой на основе ключа.

Блочные шифры.

Представляют собой последовательность (с возможным повторением и чередованием) основных методов преобразования, применяемую к блоку (части) шифруемого текста. Блочные шифры на практике встречаются чаще, чем «чистые» преобразования того или иного класса в силу их более высокой криптостойкости. Российский и американский стандарты шифрования основаны именно на этом классе шифров.

Перестановки

Перестановкой σ набора целых чисел (0,1,...,N-1) называется его переупорядочение. Для того чтобы показать, что целое i перемещено из позиции i в позицию σ(i), где $0 \le (i) < n$, будем использовать запись

$$\sigma=(\sigma(0), \sigma(1),..., \sigma(N-1)).$$

Число перестановок из (0,1,...,N-1) равно $n!=1*2*...*(N-1)*N$. Введем обозначение σ для взаимно-однозначного отображения (гомоморфизма) набора $S=\{s_0,s_1, ...,s_{N-1}\}$, состоящего из n элементов, на себя.

$$\sigma: S \to S$$

$$\sigma: s_i \to s_{\sigma(i)}, 0 \le i < n$$

Будем говорить, что в этом смысле σ является *перестановкой элементов* S. И, наоборот, автоморфизм S соответствует перестановке целых чисел (0,1,2,.., n-1).

Криптографическим преобразованием T для алфавита Z_m называется последовательность автоморфизмов: $T=\{T^{(n)}: 1 \le n < \infty\}$

$$T^{(n)}: Z_{m,n} \to Z_{m,n}, 1 \le n < \infty$$

Каждое $T^{(n)}$ является, таким образом, перестановкой n-грамм из $Z_{m,n}$.

Поскольку $T^{(i)}$ и $T^{(j)}$ могут быть определены независимо при $i \ne j$, число криптографических преобразований исходного текста размерности n равно $(m^n)!$[2]. Оно возрастает непропорционально при увеличении m и n: так, при m=33 и n=2 число различных криптографических преобразований равно 1089!. Отсюда следует, что потенциально существует большое число отображений исходного текста в шифрованный.

Практическая реализация криптографических систем требует, чтобы преобразования $\{T_k: k \in K\}$ были определены алгоритмами, зависящими от относительно небольшого числа параметров (ключей).

Системы подстановок

Определение *Подстановкой* π на алфавите Z_m называется автоморфизм Z_m, при котором буквы исходного текста t замещены буквами шифрованного текста π(t):

$$Z_m \to Z_m; \pi: t \to \pi(t).$$

Набор всех подстановок называется симметрической группой Z_m и будет в дальнейшем обозначаться как $SYM(Z_m)$.

[2] Здесь и далее **m** - объем используемого алфавита.

Утверждение $SYM(Z_m)$ с операцией произведения является группой, т.е. операцией, обладающей следующими свойствами:

1.*Замкнутость*: произведение подстановок $\pi_1\pi_2$ является подстановкой:

$$\pi: t \rightarrow \pi_1(\pi_2(t)).$$

2.*Ассоциативность*: результат произведения $\pi_1\pi_2\pi_3$ не зависит от порядка расстановки скобок:

$$(\pi_1\pi_2)\pi_3=\pi_1(\pi_2\pi_3)$$

3.*Существование нейтрального элемента*: постановка i, определяемая как $i(t)=t$, $0\le t<m$, является нейтральным элементом $SYM(Z_m)$ по операции умножения: $i\pi=\pi i$ для $\forall\pi\in SYM(Z_m)$.

4.*Существование обратного*: для любой подстановки π существует единственная обратная подстановка π^{-1}, удовлетворяющая условию

$$\pi\pi^{-1}=\pi^{-1}\pi=i.$$

Число возможных подстановок в симметрической группе Z_m называется *порядком* $SYM(Z_m)$ и равно $m!$.

Определение. *Ключом* подстановки k для Z_m называется последовательность элементов симметрической группы Z_m:

$$k=(p_0,p_1,...,p_{n-1},...),\ p_n\in SYM(Z_m),\ 0\le n<\infty$$

Подстановка, определяемая ключом k, является криптографическим преобразованием T_k, при помощи которого осуществляется преобразование n-граммы исходного текста $(x_0, x_1, .., x_{n-1})$ в n-грамму шифрованного текста $(y_0, y_1, ..., y_{n-1})$:

$$y_i=p(x_i),\quad 0\le i<n$$

где n – произвольное ($n=1,2,..$). T_k называется моноалфавитной подстановкой, если p неизменно при любом i, $i=0,1,...$, в противном случае T_k называется многоалфавитной подстановкой.

Примечание. К наиболее существенным особенностям подстановки T_k относятся следующие:

1. Исходный текст шифруется посимвольно. Шифрования n-граммы $(x_0, x_1, .., x_{n-1})$ и ее префикса $(x_0, x_1, .., x_{s-1})$ связаны соотношениями

$$T_k(x_0, x_1, .., x_{n-1})=(y_0, y_1, ..., y_{n-1})$$

$$T_k(x_0, x_1, .., x_{s-1})=(y_0, y_1, ..., y_{s-1})$$

2. *Буква шифрованного текста y_i является функцией только i-й компоненты ключа p_i и i-й буквы исходного текста x_i.*

Подстановка Цезаря

Подстановка Цезаря является самым простым вариантом подстановки. Она относится к группе *моноалфавитных подстановок*.

Определение. Подмножество $C_m=\{C_k: 0\le k<m\}$ симметрической группы $SYM(Z_m)$, содержащее m подстановок

$$C_k: j\to(j+k) \pmod m,\ 0\le k < m,$$

называется подстановкой Цезаря.

Умножение коммутативно, $C_kC_j=C_jC_k=C_{j+k}$, C_0 – идентичная подстановка, а обратной к $C_к$ является $C_k^{-1}=C_{m-k}$, где $0<k<m$. Семейство подстановок Цезаря названо по имени римского императора Гая Юлия Цезаря, который поручал Марку Туллию Цицерону составлять послания с использованием 50-буквенного алфавита и подстановки C_3.

Подстановка определяется по таблице замещения, содержащей пары соответствующих букв «исходный текст – шифрованный текст». Для C_3 подстановки приведены в Табл. 1. Стрелка (→) означает, что буква исходного текста (слева) шифруется при помощи C_3 в букву шифрованного текста (справа).

Определение. *Системой Цезаря* называется моноалфавитная подстановка, преобразующая n-грамму исходного текста $(x_0, x_1, ..., x_{n-1})$ в n-грамму шифрованного текста $(y_0, y_1, ..., y_{n-1})$ в соответствии с правилом

$$y_i=C_k(x_i),\ 0\le i<n.$$

Например, ВЫШЛИТЕ_НОВЫЕ_УКАЗАНИЯ посредством подстановки C_3 преобразуется в еюыолхиврсеюивцнгкгрлб.

Таблица 1.

А→г	Й→м	Т→х	Ы→ю
Б→д	К→н	У→ц	Ь→я
В→е	Л→о	Ф→ч	Э→_
Г→ж	М→п	Х→ш	Ю→а
Д→з	Н→р	Ц→щ	Я→б
Е→и	О→с	Ч→ъ	_→в
Ж→й	П→т	Ш→ы	
З→к	Р→у	Щ→ь	
И→л	С→ф	Ъ→э	

При своей несложности система легко уязвима. Если злоумышленник имеет

1) шифрованный и соответствующий исходный текст или

2) шифрованный текст выбранного злоумышленником исходного текста,

то определение ключа и дешифрование исходного текста тривиальны.

Более эффективны обобщения подстановки Цезаря - *шифр Хилла* и *шифр Плэйфера*. Они основаны на подстановке не отдельных символов, а 2-грамм (шифр Плэйфера) или *n*-грамм[3] (шифр Хилла). При более высокой криптостойкости они значительно сложнее для реализации и требуют достаточно большого количества ключевой информации.

Многоалфавитные системы. Системы одноразового использования.

Слабая криптостойкость моноалфавитных подстановок преодолевается с применением подстановок многоалфавитных.

Многоалфавитная подстановка определяется ключом $\pi=(\pi_1, \pi_2, ...)$, содержащим не менее двух различных подстановок. В начале рассмотрим многоалфавитные системы подстановок с нулевым начальным смещением.

Пусть $\{K_i: 0\le i<n\}$ - независимые случайные переменные с одинаковым распределением вероятностей, принимающие значения на множестве Z_m

$$P_{кл}\{(K_0, K_1, ..., K_{n-1})=(k_0, k_1, ..., k_{n-1})\}=(1/m)^n$$

Система одноразового использования преобразует исходный текст

$$X=(X_0, x_1, ..., x_{n-1})$$

в шифрованный текст

$$Y=(Y_0, y_1, ..., y_{n-1})$$

при помощи подстановки Цезаря

$$Y_i=C_{Ki}(x_i)=(K_i+X_i) \pmod m \quad i=0...n-1 \qquad (1)$$

Для такой системы подстановки используют также термин «одноразовая лента» и «одноразовый блокнот». Пространство ключей К системы одноразовой подстановки является вектором рангов $(K_0, K_1, ..., K_{n-1})$ и содержит m^n точек.

Рассмотрим небольшой пример шифрования с бесконечным ключом. В качестве ключа примем текст

«БЕСКОНЕЧНЫЙ_КЛЮЧ....».

Зашифруем с его помощью текст «ШИФР_НЕРАСКРЫВАЕМ». Шифрование оформим в таблицу:

ШИФРУЕМЫЙ_ТЕКСТ	24	8	20	16	19	5	12	27	9	32	18	5	10	17	18
БЕСКОНЕЧНЫЙ_КЛЮЧ	1	5	17	10	14	13	5	23	13	27	9	32	10	11	30
ЩРДЪАТТССЦЪЫДФЫП	25	13	4	26	0	18	17	17	22	26	27	4	20	28	15

Исходный текст невозможно восстановить без ключа.

Наложение белого шума в виде бесконечного ключа на исходный текст меняет статистические характеристики языка источника. Системы

[3] *n*-граммой называется последовательность из *n* символов алфавита.

одноразового использования *теоретически не расшифруемы*[4], так как не содержат достаточной информации для восстановления текста.

Почему же эти системы неприменимы для обеспечения секретности при обработке информации? Ответ простой - они непрактичны, так как требуют независимого выбора значения ключа для каждой буквы исходного текста. Хотя такое требование может быть и не слишком трудным при передаче по прямому кабелю Москва - Нью-Йорк, но для информационных оно непосильно, поскольку там придется шифровать многие миллионы знаков.

Посмотрим, что получится, если ослабить требование шифровать каждую букву исходного текста отдельным значением ключа.

Системы шифрования Вижинера

Начнем с конечной последовательности ключа

$$k = (k_0, k_1, ..., k_n),$$

которая называется *ключом пользователя*, и продлим ее до бесконечной последовательности, повторяя цепочку. Таким образом, получим *рабочий ключ*

$$k = (k_0, k_1, ..., k_n),\ k_j = k_{(j \bmod r)},\ 0 \le j < \infty .$$

Например, при $r = \infty$ и ключе пользователя 15 8 2 10 11 4 18 рабочий ключ будет периодической последовательностью:

15 8 2 10 11 4 18 15 8 2 10 11 4 18 15 8 2 10 11 4 18 ...

Определение. *Подстановка Вижинера* $\mathrm{VI}G_k$ определяется как

$$\mathrm{VI}G_k : (x_0, x_1, ..., x_{n-1}) \to (y_0, y_1, ..., y_{n-1}) = (x_0+k, x_1+k, .., x_{n-1}+k).$$

Таким образом:

1) исходный текст x делится на r *фрагментов*

$$x_i = (x_i , x_{i+r} , ..., x_{i+r(n-1)}),\ 0 \le i < r;$$

2) i-й фрагмент исходного текста x_i шифруется при помощи подстановки Цезаря C_k :

$$(x_i , x_{i+r} , ..., x_{i+r(n-1)}) \to (y_i , y_{i+r} , ..., y_{i+r(n-1)}),$$

Вариант системы подстановок Вижинера при m=2 называется *системой Вернама (1917 г)*.

В то время ключ k=(k_0 ,k_1 ,...,$k_{к-1}$) записывался на бумажной ленте. Каждая буква исходного текста в алфавите, расширенном некоторыми дополнительными знаками, сначала переводилась с использованием *кода Бодо* в пятибитовый символ. К исходному тексту Бодо добавлялся ключ (по модулю 2). Старинный телетайп фирмы AT&T со считывающим устройством Вернама и оборудованием для шифрования, использовался корпусом связи армии США.

[4] К вопросу о том, существует ил не существует абсолютно надежная криптосистема.

Очень распространена плохая с точки зрения секретности *практика использовать слово или фразу в качестве ключа* для того, чтобы $k=(k_0, k_1, ..., k_{к-1})$ было легко запомнить. В ИС для обеспечения безопасности информации это недопустимо. Для получения ключей должны использоваться программные или аппаратные средства случайной генерации ключей.

Пример. Преобразование текста с помощью подстановки Вижинера (r=4)

Исходный текст (ИТ1):

НЕ_СЛЕДУЕТ_ВЫБИРАТЬ_НЕСЛУЧАЙНЫЙ_КЛЮЧ

Ключ: КЛЮЧ

Разобьем исходный текст на блоки по 4 символа:

НЕ_С ЛЕДУ ЕТ_В ЫБИР АТЬ_ НЕСЛ УЧАЙ НЫЙ_ КЛЮЧ

и наложим на них ключ (используя таблицу Вижинера):
Н+К=Ч, Е+Л=Р и т.д.

Получаем зашифрованный (ЗТ1) текст:

ЧРЭЗ ХРБЙ ПЭЭЩ ДМЕЖ КЭЩЦ ЧРОБ ЭБЮ_ ЧЕЖЦ ФЦЫН

Можно выдвинуть и обобщенную систему Вижинера. ЕЕ можно сформулировать не только при помощи подстановки Цезаря.

Пусть x - подмножество симметрической группы $SYM(Z_m)$.

Определение. *r-многоалфавитный ключ* шифрования есть r-набор $\pi = (\pi_0, \pi_1, ..., \pi_{r-1})$ с элементами в x.

Обобщенная система Вижинера преобразует исходный текст $(x_0, x_1, ..., x_{n-1})$ в шифрованный текст $(y_0, y_1, ..., y_{n-1})$ при помощи ключа $\pi = (\pi_0, \pi_1, ..., \pi_{r-1})$ по правилу

$$VIG_k : (x_0, x_1, ..., x_{n-1}) \rightarrow (y_0, y_1, ..., y_{n-1}) = (\pi_0(x_0), \pi_1(x_1), ..., \pi_{n-1}(x_{n-1})),$$

где используется условие $\pi_i = \pi_{i \bmod r}$.

Следует признать, что и многоалфавитные подстановки в принципе доступны криптоаналитическому исследованию. Криптостойкость многоалфавитных систем резко убывает с уменьшением длины ключа.

Тем не менее такая система как шифр Вижинера допускает несложную аппаратную или программную реализацию и при достаточно большой длине ключа может быть использован в современных ИС.

Гаммирование

Гаммирование является также широко применяемым криптографическим преобразованием. На самом деле граница между гаммированием и использованием бесконечных ключей и шифров Вижинера, о которых речь шла выше, весьма условная.

Принцип *шифрования* гаммированием заключается в генерации гаммы шифра с помощью датчика псевдослучайных чисел и наложении полученной гаммы на открытые данные обратимым образом (например, используя сложение по модулю 2).

Процесс *дешифрования* данных сводится к повторной генерации гаммы шифра при известном ключе и наложении такой гаммы на зашифрованные данные.

Полученный зашифрованный текст является достаточно трудным для раскрытия в том случае, если гамма шифра не содержит повторяющихся битовых последовательностей. По сути дела гамма шифра должна изменяться случайным образом для каждого шифруемого слова. Фактически же, если период гаммы превышает длину всего зашифрованного текста и неизвестна никакая часть исходного текста, то шифр можно раскрыть только прямым перебором (пробой на ключ). Криптостойкость в этом случае определяется размером ключа.

Метод гаммирования становится бессильным, если злоумышленнику становится известен фрагмент исходного текста и соответствующая ему шифрограмма. Простым вычитанием по модулю получается отрезок ПСП и по нему восстанавливается вся последовательность. Злоумышленники может сделать это на основе догадок о содержании исходного текста. Так, если большинство посылаемых сообщений начинается со слов «СОВ.СЕКРЕТНО», то криптоанализ всего текста значительно облегчается. Это следует учитывать при создании реальных систем информационной безопасности.

Ниже рассматриваются наиболее распространенные методы генерации гамм, которые могут быть использованы на практике.

Датчики ПСЧ

Чтобы получить линейные последовательности элементов гаммы, длина которых превышает размер шифруемых данных, используются *датчики ПСЧ*. На основе теории групп было разработано несколько типов таких датчиков.

Конгруэнтные датчики

В настоящее время наиболее доступными и эффективными являются *конгруэнтные* генераторы ПСП. Для этого класса генераторов можно сделать математически строгое заключение о том, какими свойствами обладают выходные сигналы этих генераторов с точки зрения периодичности и случайности.

Одним из хороших конгруэнтных генераторов является линейный конгруэнтный датчик ПСЧ. Он вырабатывает последовательности псевдослучайных чисел T(i), описываемые соотношением

$$T(i+1) = (A*T(i)+C) \bmod m,$$

где A и C - константы, T(0) - исходная величина, выбранная в качестве порождающего числа. Очевидно, что эти три величины и образуют ключ.

Такой датчик ПСЧ генерирует псевдослучайные числа с определенным периодом повторения, зависящим от выбранных значений A и C. Значение *m* обычно устанавливается равным 2n , где *n* - длина машинного слова в битах. Датчик имеет максимальный период M до того, как генерируемая последовательность начнет повторяться. По причине, отмеченной ранее, необходимо выбирать числа A и C такие, чтобы период M был максимальным. Как показано Д. Кнутом, линейный конгруэнтный датчик ПСЧ имеет максимальную длину M тогда и только тогда, когда C - нечетное, и A mod 4 = 1.

Для шифрования данных с помощью датчика ПСЧ может быть выбран ключ любого размера. Например, пусть ключ состоит из набора чисел *x*(j) размерностью b, где j=1, 2, ..., *n*. Тогда создаваемую гамму шифра *G* можно представить как объединение непересекающихся множеств H(j).

Датчики М-последовательностей[5]

М-последовательности также популярны, благодаря относительной легкости их реализации.

М-последовательности представляют собой линейные рекуррентные последовательности максимального периода, формируемые *k*-разрядными генераторами на основе регистров сдвига. На каждом такте поступивший бит сдвигает k предыдущих и к нему добавляется их сумма по модулю 2. Вытесняемый бит добавляется к гамме.

Строго это можно представить в виде следующих отношений:

$$r_1:=r_0 \quad r_2:=r_1 \quad \ldots \quad r_{k-1}:=r_{k-2}$$

$$r_0:=a_0\, r_1 \oplus a_1\, r_2 \oplus \ldots \oplus a_{k-2}\, r_{k-1}$$

$$\Gamma_i:= r_{k-}$$

Здесь $r_0\ r_1\ \ldots\ r_{k-1}$ - k однобитных регистров, $a_0\ a_1\ \ldots\ a_{k-1}$ - коэффициенты неприводимого двоичного полинома степени k-1. Γ_i - i-е значение выходной гаммы.

Период М-последовательности исходя из ее свойств равен 2^k-1.

Другим важным свойством М-последовательности является *объем ансамбля*, т.е. количество различных М-последовательностей для заданного *k*. Эта характеристика приведена в таблице:

k	Объем ансамбля
5	6
6	8

[5] Материал предоставлен Ю. Г. Писаревым

7	18
8	16
9	48
10	60
16	2048

Очевидно, что такие объемы ансамблей последовательности неприемлемы.

Поэтому на практике часто используют последовательности Голда, образующиеся суммированием нескольких М-последовательностей. Объем ансамблей этих последовательностей на несколько порядков превосходят объемы ансамблей порождающих М-последовательностей. Так при k=10 ансамбль увеличивается от 1023 (М-последовательности) до 388000.

Также перспективными представляются нелинейные датчики ПСП (например сдвиговые регистры с элементом И в цепи обратной связи), однако их свойства еще недостаточно изучены.

Возможны и другие, более сложные варианты выбора порождающих чисел для гаммы шифра.

Шифрование с помощью датчика ПСЧ является довольно распространенным криптографическим методом. Во многом качество шифра, построенного на основе датчика ПСЧ, определяется не только и не столько характеристиками датчика, сколько алгоритмом получения гаммы. Один из фундаментальных принципов криптологической практики гласит, даже сложные шифры могут быть очень чувствительны к простым воздействиям.

Стандарт шифрования данных ГОСТ 28147-89[6]

Важной задачей в обеспечении гарантированной безопасности информации в ИС является разработка и использования стандартных алгоритмов шифрования данных. Первым среди подобных стандартов стал американский DES, представляющий собой последовательное использование замен и перестановок. В настоящее время все чаще говорят о неоправданной сложности и невысокой криптостойкости. На практике приходится использовать его модификации.

Более эффективным является отечественный стандарт шифрования данных.

Он рекомендован к использованию для защиты любых данных, представленных в виде двоичного кода, хотя не исключаются и другие методы шифрования. Данный стандарт формировался с учетом мирового опыта, и в частности, были приняты во внимание недостатки и нереализованные возможности

[6] ГОСТ 28147-89 закрыт грифом ДСП поэтому дальнейшее изложение сделано по изданию Спесивцев А.В. и др. «Защита информации в персональных ЭВМ», М., Радио и связь, 1992.

алгоритма DES, поэтому использование стандарта ГОСТ предпочтительнее. Алгоритм достаточно сложен и ниже будет описана в основном его концепция.

Введем ассоциативную операцию конкатенации, используя для нее мультипликативную запись. Кроме того будем использовать следующие операции сложения:

- A⊕B - побитовое сложение по модулю 2;
- A[+]B - сложение по модулю 2^{32};
- A{+}B - сложение по модулю 2^{32}-1;.

Алгоритм криптографического преобразования предусматривает несколько режимов работы. Во всех режимах используется ключ W длиной 256 бит, представляемый в виде восьми 32-разрядных чисел *x*(i).

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

Для дешифрования используется тот же ключ, но процесс дешифрования является инверсным по отношению к исходному.

Самый простой из возможных режимов - *замена.*

Пусть открытые блоки разбиты на блоки по 64 бит в каждом, которые обозначим как T(j).

Очередная последовательность бит T(j) разделяется на две последовательности B(0) и A(0) по 32 бита (правый и левый блоки). Далее выполняется итеративный процесс шифрования описываемый следующими формулами, вид который зависит от :i:

- Для i=1, 2, ..., 24, j=(i-1) mod 8;

$$A(i) = f(A(i-1)\ [+]\ x(j)) \oplus B(i-1)$$
$$B(i) = A(i-1)$$

- Для i=25, 26, ..., 31, j=32-i;

$$A(i) = f(A(i-1)\ [+]\ x(j)) \oplus B(i-1)$$
$$B(i) = A(i-1)$$

- Для i=32

$$A(32) = A(31)$$
$$B(32) = f(A(31)\ [+]\ x(0)) \oplus B(31).$$

Здесь i обозначает номер итерации. Функция *f* – функция шифрования.

Функция шифрования включает две операции над 32-разрядным аргументом.

Первая операция является подстановкой *K*. Блок подстановки K состоит из 8 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 вектора, каждый из который преобразуется в 4-разрядный вектор соответствующим узлом замены, представляющим из себя 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 векторы последовательно объединяются в 32-разрядный выходной.

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

$$T=A(32)B(32).$$

Остальные блоки открытых данных в режиме простой замены зашифровываются аналогично.

Следует учитывать, что данный режим шифрования обладает ограниченной криптостойкостью.

Другой режим шифрования называется *режимом гаммирования*.

Открытые данные, разбитые на 64-разрядные блоки T(i) (i=1,2,...,m) *(m* определяется объемом шифруемых данных), зашифровываются в режиме гаммирования путем поразрядного сложения по модулю 2 с гаммой шифра $\Gamma_ш$, которая вырабатывается блоками по 64 бит, т.е.

$$\Gamma_ш=(\Gamma(1),\Gamma(2),....,\Gamma(m)).$$

Уравнение шифрования данных в режиме гаммирования может быть представлено в следующем виде:

$$Ш(i)=A(Y(i-1) \oplus C2, Z(i-1)) \{+\} C(1) \oplus T(i)=\Gamma(i) \oplus T(i)$$

В этом уравнении Ш(i) обозначает 64-разрядный блок зашифрованного текста, А - функцию шифрования в режиме простой замены (аргументами этой функции являются два 32-разрядных числа). С1 и С2 - константы, заданные в ГОСТ 28147-89. Величины *y*(i) и *Z*(i) определяются итерационно по мере формирования гаммы следующим образом:

(Y(0),Z(0))=A(S), S - 64-разрядная двоичная последовательность

(Y(i),Z(i))=(Y(i-1) [+] C2, Z(i-1) {+} C(1)), i=1, 2, ..., *m*.

64-разрядная последовательность, называемая синхропосылкой, не является секретным элементом шифра, но ее наличие необходимо как на передающей стороне, так и на приемной.

Режим гаммирования с обратной связью очень похож на режим гаммирования. Как и в режиме гаммирования открытые данные, разбитые на 64-разрядные блоки T(i), зашифровываются путем поразрядного сложения по модулю 2 с гаммой шифра $\Gamma_ш$, которая вырабатывается блоками по 64 бит:

$$\Gamma_ш=(\Gamma(1), \Gamma(2), ..., \Gamma(m)).$$

Уравнение шифрования данных в режиме гаммирования с обратной связью выглядят следующим образом:

$$Ш(1)=A(S)\oplus T(1)=\Gamma(1)\oplus T(1),$$

$$Ш(i)=A(Ш(i-1)\oplus T(i)=\Gamma(i)\oplus T(i),\ \ i=2, 3, ..., m.$$

В ГОСТ 28147-89 определяется процесс выработки имитовставки, который единообразен для всех режимов шифрования. Имитовставка - это блок

из p бит (имитовставка $И_р$), который вырабатывается либо перед шифрованием всего сообщения. либо параллельно с шифрованием по блокам. Параметр p выбирается в соответствии с необходимым уровнем имитозащищенности.

Для получения имитовставки открытые данные представляются также в виде блоков по 64 бит. Первый блок открытых данных T(1) подвергается преобразованию, соответствующему первым 16 циклам алгоритма режима простой замены. Причем в качестве ключа используется тот же ключ, что и для шифрования данных. Полученное 64-разрядно число суммируется с открытым блоком T(2) и сумма вновь подвергается 16 циклам шифрования для режима простой замены. Данная процедура повторятся для всех m блоков сообщения. Из полученного 64-разрядного числа выбирается отрезок $И_р$ длиной p бит.

Имитовставка передается по каналу связи после зашифрованных данных. На приемной стороне аналогичным образом из принятого сообщения выделяется? имитовставка и сравнивается с полученной откуда?. В случае несовпадения имитовставок сообщение считается ложным.

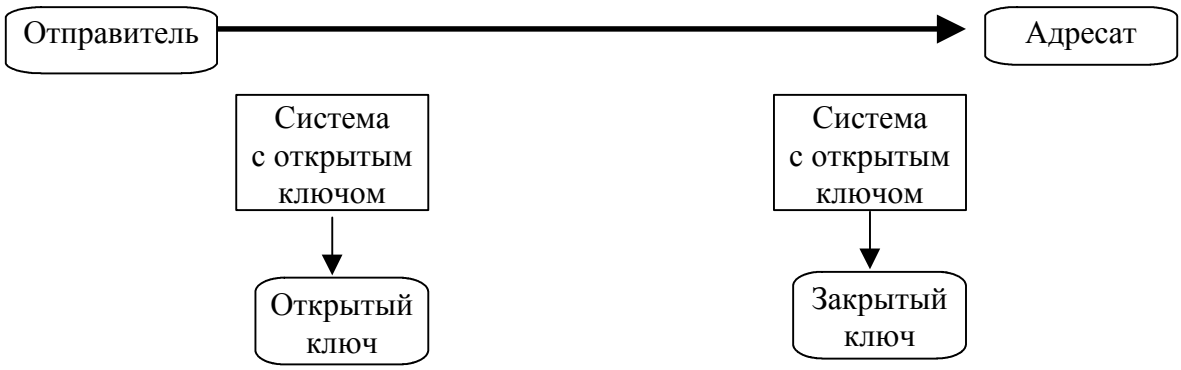
Системы с открытым ключом

Как бы ни были сложны и надежны криптографические системы - их слабое мест при практической реализации - проблема *распределения ключей*. Для того, чтобы был возможен обмен конфиденциальной информацией между двумя субъектами ИС, ключ должен быть сгенерирован одним из них, а затем каким-то образом опять же в конфиденциальном порядке передан другому. Т.е. в общем случае для передачи ключа опять же требуется использование какой-то криптосистемы.

Для решения этой проблемы на основе результатов, полученных классической и современной алгеброй, были предложены *системы с открытым ключом.*

Суть их состоит в том, что каждым адресатом ИС генерируются два ключа, связанные между собой по определенному правилу. Один ключ объявляется *открытым*, а другой *закрытым*. Открытый ключ публикуется и доступен любому, кто желает послать сообщение адресату. Секретный ключ сохраняется в тайне.

Исходный текст шифруется открытым ключом адресата и передается ему. Зашифрованный текст в принципе не может быть расшифрован тем же открытым ключом. Дешифрование сообщение возможно только с использованием закрытого ключа, который известен только самому адресату.



Криптографические системы с открытым ключом используют так называемые *необратимые или односторонние функции*, которые обладают следующим свойством: при заданном значении x относительно просто вычислить значение $f(x)$, однако если $y=f(x)$, то нет простого пути для вычисления значения x.

Множество классов необратимых функций и порождает все разнообразие систем с открытым ключом. Однако не всякая необратимая функция годится для использования в реальных ИС.

В самом определении необратимости присутствует неопределенность. Под *необратимостью* понимается не теоретическая необратимость, а практическая невозможность вычислить обратное значение используя современные вычислительные средства за обозримый интервал времени.

Поэтому чтобы гарантировать надежную защиту информации, к системам с открытым ключом (СОК) предъявляются два важных и очевидных требования:

1. Преобразование исходного текста должно быть необратимым и исключать его восстановление на основе открытого ключа.

2. Определение закрытого ключа на основе открытого также должно быть невозможным на современном технологическом уровне. При этом желательна точная нижняя оценка сложности (количества операций) раскрытия шифра.

Алгоритмы шифрования с открытым ключом получили широкое распространение в современных информационных системах. Так, алгоритм RSA стал мировым стандартом де-факто для открытых систем и рекомендован МККТТ.

Вообще же все предлагаемые сегодня криптосистемы с открытым ключом опираются на один из следующих типов необратимых преобразований:

1. Разложение больших чисел ан простые множители.

2. Вычисление логарифма в конечном поле.

3. Вычисление корней алгебраических уравнений.

Здесь же следует отметить, что алгоритмы криптосистемы с открытым ключом (СОК) можно использовать в трех назначениях.

1. Как *самостоятельные средства защиты* передаваемых и хранимых данных.

2. Как *средства для распределения ключей*. Алгоритмы СОК более трудоемки, чем традиционные криптосистемы. Поэтому часто на практике рационально с помощью СОК распределять ключи, объем которых как информации незначителен. А потом с помощью обычных алгоритмов осуществлять обмен большими информационными потоками.

3. *Средства аутентификации пользователей*. Об этом будет рассказано в главе «Электронная подпись».

Ниже рассматриваются наиболее распространенные системы с открытым ключом.

Алгоритм RSA

Несмотря на довольно большое число различных СОК, наиболее популярна - криптосистема RSA, разработанная в 1977 году и получившая название в честь ее создателей: Рона Ривеста[7], Ади Шамира и Леонарда Эйдельмана.

Они воспользовались тем фактом, что нахождение больших простых чисел в вычислительном отношении осуществляется легко, но разложение на множители произведения двух таких чисел практически невыполнимо. Доказано (теорема Рабина), что раскрытие шифра RSA эквивалентно такому разложению. Поэтому для любой длины ключа можно дать нижнюю оценку числа операций для раскрытия шифра, а с учетом производительности современных компьютеров оценить и необходимое на это время.

[7] В настоящее время он возглавляет компанию RSA Data Security

Возможность гарантированно оценить защищенность алгоритма RSA стала одной из причин популярности этой СОК на фоне десятков других схем. Поэтому алгоритм RSA используется в банковских компьютерных сетях, особенно для работы с удаленными клиентами (обслуживание кредитных карточек).

В настоящее время алгоритм RSA используется во многих стандартах, среди которых SSL, S-HHT*P*, S-MIME, S/WAN, STT и *P*CT.

Рассмотрим математические результаты, положенные в основу этого алгоритма.

Теорема 1. (Малая теорема Ферма.)

Если p - простое число, то

$$x^{p-1} = 1 \pmod{p} \qquad (1)$$

для любого x, простого относительно p, и

$$x^{p} = x \pmod{p} \qquad (2)$$

для любого x.

Доказательство. Достаточно доказать справедливость уравнений (1) и (2) для $x \in Z^p$. Проведем доказательство методом индукции.
Очевидно, что уравнение (8.2.2) выполняется при x=0 и 1. Далее

$$x^p=(x\text{-}1+1)^p= \sum_{0\le j\le p} C(p,j)(x\text{-}1)^j=(x\text{-}1)^p+1 \pmod{p},$$

так как $C(p,j)=0 \pmod{p}$ при $0<j<p$. С учетом этого неравенства и предложений метода доказательства по индукции теорема доказана.

Определение. *Функцией Эйлера* φ(n) называется число положительных целых, меньших n и простых относительно n.

n	2	3	4	5	6	7	8	9	10	11	12
φ(n)	1	2	2	3	2	6	4	6	4	10	4

Теорема 2. Если $n=pq$, (p и q - отличные друг от друга простые числа), то

$$\varphi(n)=(p\text{-}1)(q\text{-}1).$$

Теорема 3. Если $n=pq$, (p и q - отличные друг от друга простые числа) и x - простое относительно p и q, то

$$x_{\varphi}^{(n)} = 1 \pmod{n}.$$

Следствие . Если $n=pq$, (p и q - отличные друг от друга простые числа) и e простое относительно φ(n), то отображение

$$E_{e,n}: x \rightarrow x^e \pmod{n}$$

является взаимно однозначным на $\mathbf{Z}_n$.

Очевиден и тот факт, что если e - простое относительно φ(n), то существует целое d, такое, что

$$ed = 1 \pmod{\varphi(n)} \qquad (3)$$

На этих математических фактах и основан популярный алгоритм RSA.

Пусть $n=pq$, где p и q - различные простые числа. Если e и d удовлетворяют уравнению (8.2.3), то отображения $E_{e,n}$ и $E_{d,n}$ являются инверсиями на Z_n. Как $E_{e,n}$, так и $E_{d,n}$ легко рассчитываются, когда известны e, d, p, q. Если известны e и n, но p и q неизвестны, то $E_{e,n}$ представляет собой одностороннюю функцию; нахождение $E_{d,n}$ по заданному n равносильно разложению n. Если p и q - достаточно большие простые, то разложение n практически не осуществимо. Это и заложено в основу системы шифрования RSA.

Пользователь i выбирает пару различных простых p_i и q_i и рассчитывает пару целых (e_i, d_i), которые являются простыми относительно $\varphi(n_i)$, где $n_i=p_i q_i$. Справочная таблица содержит публичные ключи $\{(e_i, n_i)\}$.
Предположим, что исходный текст

$$x =(x_0, x_1, ..., x_{n-1}), x \in Z_n , 0 \le i < n,$$

сначала представлен по основанию n_i :

$$N = c_0+c_{i\,ni}+....$$

Пользователь i зашифровывает текст при передаче его пользователю j, применяя к n отображение E_{d_i,n_i} :

$$N \rightarrow E_{d_i,n_i}\, n = n'.$$

Пользователь j производит дешифрование n', применяя E_{e_i,n_i} :

$$N' \rightarrow E_{e_i,n_i}\, n' = E_{e_i,n_i}\, E_{d_i,n_i}\, n = n .$$

Очевидно, для того чтобы найти инверсию E_{d_i,n_i} по отношению к E_{e_i,n_i}, требуется знание множителей $n=p_i\, q_i$. Время выполнения наилучших из известных алгоритмов разложения при $n=10^{100}$ на сегодняшний день выходит за пределы современных технологических возможностей.

Рассмотрим небольшой пример, иллюстрирующий применение алгоритма RSA.

Пример Зашифруем сообщение «CAB». Для простоты будем использовать маленькие числа (на практике применяются гораздо большие).

1. Выберем p=3 и q=11.
2. Определим n=3*11=33.
3. Найдем (p-1)(q-1)=20. Следовательно, в качестве d, взаимно простое с 20, например, d=3.
4. Выберем число e. В качестве такого числа может быть взято любое число, для которого удовлетворяется соотношение (e*3) (mod 20) = 1, например 7.
5. Представим шифруемое сообщение как последовательность целых чисел с помощью отображения: A→1, B→2, C→3. Тогда сообщение принимает вид (3,1,2). Зашифруем сообщение с помощью ключа {7,33}.

$$ШТ1 = (3^7) \pmod{33} = 2187 \pmod{33} = 9,$$

$ШТ2 = (1^7) \pmod{33} = 1 \pmod{33} = 1,$

$ШТ3 = (2^7) \pmod{33} = 128 \pmod{33} = 29.$

6. Расшифруем полученное зашифрованное сообщение (9,1,29) на основе закрытого ключа {3,33}:

$ИТ1 = (9^3) \pmod{33} = 729 \pmod{33} = 3,$

$ИТ2 = (1^3) \pmod{33} = 1 \pmod{33} = 1,$

$ИТ3 = (29^3) \pmod{33} = 24389 \pmod{33} = 2.$

Итак, в реальных системах алгоритм RSA реализуется следующим образом: каждый пользователь выбирает два больших простых числа, и в соответствии с описанным выше алгоритмом выбирает два простых числа *e* и *d*. Как результат умножения первых двух чисел (*p* и *q*) устанавливается *n*.

{e,n} образует открытый ключ, а *{d,n}* - закрытый (хотя можно взять и наоборот).

Открытый ключ публикуется и доступен каждому, кто желает послать владельцу ключа сообщение, которое зашифровывается указанным алгоритмом. После шифрования, сообщение невозможно раскрыть с помощью открытого ключа. Владелец же закрытого ключа без труда может расшифровать принятое сообщение.

Практическая реализация RSA

В настоящее время алгоритм RSA активно реализуется как в виде самостоятельных криптографических продуктов[8], так и в качестве встроенных средств в популярных приложениях[9].

Важная проблема практической реализации - *генерация больших простых чисел*. Решение задачи «в лоб» - генерация случайного большого числа *n* (нечетного) и проверка его делимости на множители от 3 вплоть до $n^{0.5}$. В случае неуспеха следует взять n+2 и так далее.[10]

В принципе в качестве p и q можно использовать «почти» простые числа, то есть числа для которых вероятность того, что они простые, стремится к 1. Но в случае, если использовано составное число, а не простое, криптостойкость RSA падает. Имеются неплохие алгоритмы, которые позволяют генерировать «почти» простые числа с уровнем доверия 2^{-100}.

Другая проблема - *ключи какой длины следует использовать*?

Для практической реализации алгоритмов RSA полезно знать оценки трудоемкости разложения простых чисел различной длины, сделанные Шроппелем.

[8] Например, в нашумевшей программе PGP

[9] В браузерах Интернет от Microsoft и Netscape

[10] В теории чисел показано, что вероятность того, что число порядка n будет простым составляет 1/ln n

$\log_{10} n$	Число операций	Примечания
50	$1.4*10^{10}$	Раскрываем на суперкомпьютерах
100	$2.3*10^{15}$	На пределе современных технологий
200	$1.2*10^{23}$	За пределами современных технологий
400	$2.7*10^{34}$	Требует существенных изменений в технологии
800	$1.3*10^{51}$	Не раскрываем

В конце 1995 года удалось практически реализовать раскрытие шифра RSA для 500-значного ключа. Для этого с помощью сети Интернет было задействовано 1600 компьютеров.

Сами авторы RSA рекомендуют использовать следующие размеры модуля n:

- 768 бит - для частных лиц;
- 1024 бит - для коммерческой информации;
- 2048 бит - для особо секретной информации.[11]

Третий немаловажный аспект реализации RSA - *вычислительный*. Ведь приходится использовать аппарат длинной арифметики. Если используется ключ длиной *k* бит, то для операций по открытому ключу требуется $O(k^2)$ операций, по закрытому ключу - $O(k^3)$ операций, а для генерации новых ключей требуется $O(k^4)$ операций.

Криптографический пакет BSAFE 3.0 (RSA D.S.) на компьютере Pentium-90 осуществляет шифрование со скоростью 21.6 Кбит/с для 512-битного ключа и со скоростью 7.4 Кбит/с для 1024 битного. Самая «быстрая» аппаратная реализация обеспечивает скорости в 60 раз больше.

По сравнению с тем же алгоритмом DES, RSA требует в тысячи и десятки тысяч раз большее время.

Криптосистема Эль-Гамаля

Данная система является альтернативой RSA и при равном значении ключа обеспечивает ту же криптостойкость[12].

В отличие от RSA метод Эль-Гамаля основан на проблеме дискретного логарифма. Этим он похож на *алгоритм Диффи-Хелмана*. Если возводить число в степень в конечном поле достаточно легко, то восстановить аргумент по значению (то есть найти логарифм) довольно трудно.

Основу системы составляют параметры *p* и *g* - числа, первое из которых - простое, а второе - целое.

Александр генерирует секретный ключ *a* и вычисляет открытый ключ $y = g^a \bmod p$. Если Борис хочет послать Александру сообщение *m*, то он выбирает случайное число *k*, меньшее p и вычисляет

$$y_1 = g^k \bmod p \quad \text{и}$$
$$y_2 = m \oplus y^k,$$

[11] Данные оценки сделаны с учетом развития вычислительной техники вплоть до 2004 года.
[12] Однако общего мнения по поводу предпочтительности того или иного метода нет.

где ⊕ означает побитовое сложение по модулю 2. Затем Борис посылает (y_1,y_2) Александру.

Александр, получив зашифрованное сообщение, восстанавливает его:

$$m = (y_1^a \bmod p) \oplus y_2.$$

Алгоритм цифровой подписи DSA, разработанный NIST (National Institute of Standard and Technology) и являющийся частью стандарта DSS частично опирается на рассмотренный метод.

Криптосистемы на основе эллиптических уравнений

Эллиптические кривые - математический объект, который может определен над любым полем (конечным, действительным, рациональным или комплексным). В криптографии обычно используются конечные поля. Эллиптическая кривая есть множество точек (x,y), удовлетворяющее следующему уравнению:

$$y^2 = x^3 + ax + b,$$

а также бесконечно удаленная точка. Для точек на кривой довольно легко вводится операция сложения, которая играет ту же роль, что и операция умножения в криптосистемах RSA и Эль-Гамаля.

В реальных криптосистемах на базе эллиптических уравнений используется уравнение

$$y^2 = x^3 + ax + b \bmod p,$$

где p - простое.

Проблема дискретного логарифма на эллиптической кривой состоит в следующем: дана точка G на эллиптической кривой порядка r (количество точек на кривой) и другая точка Y на этой же кривой. Нужно найти единственную точку x такую, что $Y = xG$, то есть Y есть x-я степень G.

Электронная подпись

В чем состоит проблема аутентификации данных?

В конце обычного письма или документа исполнитель или ответственное лицо обычно ставит свою подпись. Подобное действие обычно преследует две цели. Во-первых, получатель имеет возможность убедиться в истинности письма, сличив подпись с имеющимся у него образцом. Во-вторых, личная подпись является юридическим гарантом авторства документа. Последний аспект особенно важен при заключении разного рода торговых сделок, составлении доверенностей, обязательств и т.д.

Если подделать подпись человека на бумаге весьма непросто, а установить авторство подписи современными криминалистическими методами - техническая деталь, то с подписью электронной дело обстоит иначе. Подделать цепочку битов, просто ее скопировав, или незаметно внести нелегальные исправления в документ сможет любой пользователь.

С широким распространением в современном мире электронных форм документов (в том числе и конфиденциальных) и средств их обработки особо актуальной стала проблема установления подлинности и авторства безбумажной документации.

В разделе криптографических систем с открытым ключом было показано, что при всех преимуществах современных систем шифрования они не позволяют обеспечить аутентификацию данных. Поэтому средства аутентификации должны использоваться в комплексе и криптографическими алгоритмами.

Итак, пусть имеются два пользователя Александр и Борис. От каких нарушений и действий злоумышленника должна защищать система аутентификации.

Отказ (ренегатство).

Александр заявляет, что он не посылал сообщение Борису, хотя на самом деле он все-таки посылал.

Для исключения этого нарушения используется *электронная* (или *цифровая*) *подпись*.

Модификация (переделка).

Борис изменяет сообщение и утверждает, что данное (измененное) сообщение послал ему Александр.

Подделка.

Борис формирует сообщение и утверждает, что данное (измененное) сообщение послал ему Александр.

Активный перехват.

Владимир перехватывает сообщения между Александром и Борисом с целью их скрытой модификации.

Для защиты от модификации, подделки и маскировки используются *цифровые сигнатуры*.

Маскировка (имитация).

Владимир посылает Борису сообщение от имени Александра .

В этом случае для защиты также используется электронная подпись.

Повтор.

Владимир повторяет ранее переданное сообщение, которое Александра посылал ранее Борису . Несмотря на то, что принимаются всевозможные меры защиты от повторов, именно на этот метод приходится большинство случаев незаконного снятия и траты денег в системах электронных платежей.

Наиболее действенным методом защиты от повтора являются

- использование *имитовставок*,
- учет входящих сообщений.

Возможные нарушения защиты сообщений,. посылаемых пользователем А пользователю В.

Электронная подпись на основе алгоритма RSA

Наиболее простым и распространенным инструментом электронной подписи является уже знакомый алгоритм RSA. Ниже оно будет рассмотрена в качестве примера. Кроме этого существуют еще десятки других схем цифровой подписи.

Предположим, что

d,p,q - секретные, а e, $n=pq$ - открытые.

Замечания.

1. Разложение по n дает: $\varphi(n)=(p-1)(q-1)$; зная $\varphi(n)$ и e, можно найти d.

2. Из e и d можно найти кратность $\varphi(n)$; кратность $\varphi(n)$ позволяет определить делители n.

Пусть DATA - передаваемое Александром Борису сообщение.

Александр подписывает DATA для Бориса при передаче :

$$E_{e_B,n_B}\{E_{d_A,n_A}\{DATA\}\}.$$

При этом он использует:

- закрытый ключ E_{d_A,n_A} Александра,

• открытый ключ E_{e_B,n_B} Бориса.

Борис может читать это подписанное сообщение сначала при помощи закрытого ключа E_{d_B,n_B} Бориса с целью получения

$$E_{d_A,n_A}\{DATA\} = E_{d_B,n_B}\{E_{e_B,n_B}\{E_{d_A,n_A}\{DATA\}\}\}$$

и затем - открытого ключа E_{e_A,n_A} Александра для получения

$$DATA = E_{e_A,n_A}\{E_{d_A,n_A}\{DATA\}\}.$$

Таким образом, у Бориса появляется сообщение DATA, посланное ему Александром.

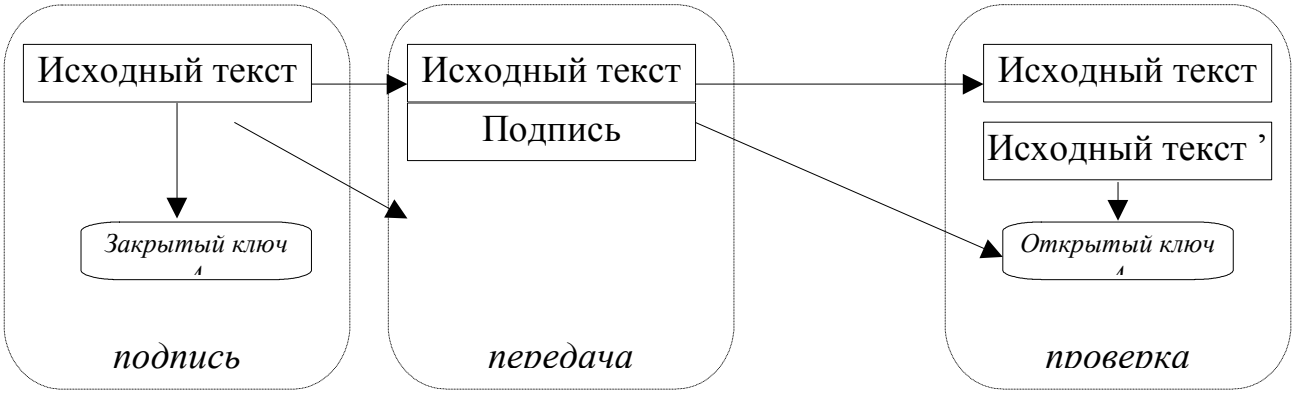
Очевидно, что данная схема позволяет защититься от нескольких видов нарушений.

Александр не может отказаться от своего сообщения, если он признает, что секретный ключ известен только ему.

Нарушитель без знания секретного ключа не может ни сформировать, ни сделать осмысленное изменение сообщения, передаваемого по линии связи.

Данная схема позволяет при решении многих конфликтных ситуаций обходиться без посредников.

Иногда нет необходимости зашифровывать передаваемое сообщение, но нужно его скрепить электронной подписью. В этом случае текст шифруется закрытым ключом отправителя и полученная цепочка символов прикрепляется к документу. Получатель с помощью открытого ключа отправителя расшифровывает подпись и сверяет ее с текстом.



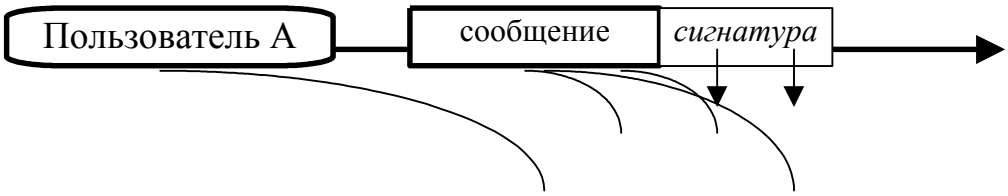
В 1991 г. Национальный институт стандартов и технологии (NIST) предложил для появившегося тогда алгоритма цифровой подписи DSA (Digital Signature Algorithm) стандарт DSS (Digital Signature Standard), в основу которого положены алгоритмы Эль-Гамаля и RSA. [13]

[13] В РФ принятые стандарты цифровой подписи Р38 и Р39, также как и ГОСТ 28147-89 имеют гриф ДСП

Цифровая сигнатура

Часто возникают ситуации, когда получатель должен уметь доказать подлинность сообщения внешнему лицу. Чтобы иметь такую возможность, к передаваемым сообщениям должны быть приписаны так называемые цифровые сигнатуры.

Цифровая сигнатура - это строка символов, зависящая как от идентификатора отправителя, так и содержания сообщения.



Цифровая сигнатура

Никто при этом кроме пользователя А не может вычислить цифровую сигнатуру А для конкретного сообщения. Никто, даже сам пользователь не может изменить посланного сообщения так, чтобы сигнатура осталась неизменной. Хотя получатель должен иметь возможность проверить является ли цифровая сигнатура сообщения подлинной. Чтобы проверить цифровую сигнатуру, пользователь В должен представить посреднику С информацию, которую он сам использовал для верификации сигнатуры.

Если помеченное сигнатурой сообщение передается непосредственно от отправителя к получателю, минуя промежуточное звено, то в этом случае идет речь об *истинной цифровой сигнатуре*.

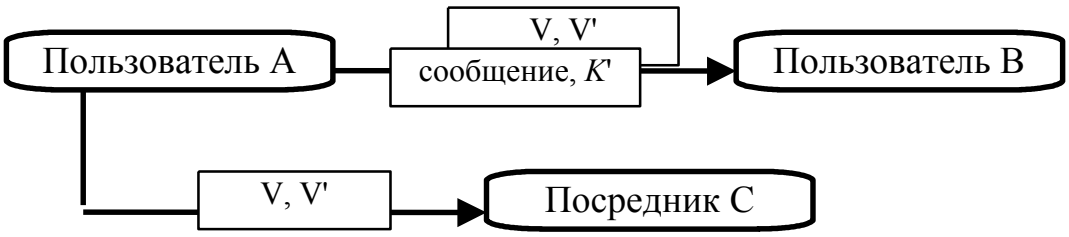
Рассмотрим типичную схему цифровой сигнатуры.

Пусть E - функция симметричного шифрования и *f* - функция отображения некоторого множества сообщений на подмножество мощности *p* из последовательности {1, ..., *n*}.

Например p=3 и *n*=9. Если *m* - сообщение , то в качестве *f* можно взять функцию *f*(*m*) = {2, 5, 7}.

Для каждого сообщения пользователь А выбирает некоторое множество ключей $K=[K_1, ..., K_n\}$ и параметров $V=\{v_1, ...,v_n\}$ для использования в качестве пометок сообщения, которое будет послано В. Множества V и $V'=\{E(v_1,K_1)$ ..., $E(v_n,K_n)\}$ посылаются пользователю В и заранее выбранному посреднику С.

Пусть *m* - сообщение и idm - объединение идентификационных номеров отправителя, получателя и номера сообщения. Если *f*({idm, *m*}), то цифровая сигнатура *m* есть множество $K'=[K_i, ..., K_j\}$. Сообщение *m*, идентификационный номер idm и цифровая сигнатура К' посылаются В.



Получатель В проверяет сигнатуру следующим образом. Он вычисляет функцию *f(*{idm, *m*}) и проверяет ее равенство К'. Затем он проверяет, что подмножество $\{v_i, ...,v_j\}$ правильно зашифровано в виде подмножества $\{E(v_i,K_i) ..., E(v_j,K_j)\}$ множества V'.

В конфликтной ситуации В посылает С сообщение *m*, идентификационный номер idm и множество ключей *K*', которое В объявляет сигнатурой *m*. Тогда посредник С так же, как и В, будет способен проверить сигнатуру. Вероятность раскрытия двух сообщений с одним и тем же значением функции *f* должна быть очень мала. Чтобы гарантировать это, число *n* должно быть достаточно большим, а число *p* должно быть больше 1, но меньше *n*.

Ряд недостатков этой модели очевиден:

- должно быть третье лицо - посредник, которому доверяют как получатель, так и отправитель;
- получатель, отправитель и посредник должны обменяться существенным объемом информации, прежде чем будет передано реальное сообщение;
- передача этой информации должна осуществляться в закрытом виде;
- эта информация используется крайне неэффективно, поскольку множества *K*, V, V' используются только один раз.

Тем не менее даже такая схема цифровой сигнатуры может использоваться в информационных системах, в которых необходимо обеспечить аутентификацию и защиту передаваемых сообщений.

Хэш-функции

Использование цифровой сигнатуры предполагает использование некоторых функций шифрования:

$$S = H(k, T),$$

где *S* - сигнатура, *k* - ключ, *T* - исходный текст.

Функция *H(k, T)* - является хэш-функцией, если она удовлетворяет следующим условиям:

1) исходный текст может быть произвольной длины;
2) само значение *H(k, T)* имеет фиксированную длину;
3) значение функции *H(k, T)* легко вычисляется для любого аргумента;
4) восстановить аргумент по значению с вычислительной точки зрения - практически невозможно;

5) функция *H(k, T)* - однозначна[14].

Из определения следует, что для любой хэш-функции есть тексты-близнецы - имеющие одинаковое значение хэш-функции, так как мощность множества аргументов неограниченно больше мощности множества значений. Такой факт получил название «эффект дня рождения».[15]

Наиболее известные из хэш-функций - MD2, MD4, MD5 и SHA.

Три алгоритма серии MD разработаны Ривестом в 1989-м, 90-м и 91-м году соответственно. Все они преобразуют текст произвольной длины в 128-битную сигнатуру.

Алгоритм MD2 предполагает:

- дополнение текста до длины, кратной 128 битам;
- вычисление 16-битной контрольной суммы (старшие разряды отбрасываются);
- добавление контрольной суммы к тексту;
- повторное вычисление контрольной суммы.

Алгоритм MD4 предусматривает:

- дополнение текста до длины, равной 448 бит по модулю 512;
- добавляется длина текста в 64-битном представлении;
- 512-битные блоки подвергаются процедуре Damgard-Merkle[16], причем каждый блок участвует в трех разных циклах.

В алгоритме MD4 довольно быстро были найдены «дыры», поэтому он был заменен алгоритмом MD5, в котором каждый блок участвует не в трех, а в четырех различных циклах.

Алгоритм SHA (Secure Hash Algorithm) разработан NIST (National Institute of Standard and Technology) и повторяет идеи серии MD. В SHA используются тексты более 2^{64} бит, которые закрываются сигнатурой длиной 160 бит. Данный алгоритм предполагается использовать в программе Capstone[17].

---

[14] При этом разделяют *слабую* и *сильную* однозначность. При слабой однозначности для заданного значения T практически невозможно отыскать другой текст T', для которого *H(k, T) = H(k, T'). При сильной однозначности для любого текста T невозможно найти другой подходящий текст, имеющий то же значение хэш-функции.*

[15] Факт теории вероятностей: в группе из 23 человек с вероятностью больше 0.5 два и более человека родились в одно и то же число.

[16] В отличие от хэш-функции - этот класс преобразований предполагает вычисление для аргументов *фиксированной длины* также фиксированных по длине значений.

[17] Государственная программа США, предполагающая централизованное хранение всех ключей, используемых организациями а частными лицами.

Управление ключами

Кроме выбора подходящей для конкретной ИС криптографической системы, важная проблема - управление ключами. Как бы ни была сложна и надежна сама криптосистема, она основана на использовании ключей. Если для обеспечения конфиденциального обмена информацией между двумя пользователями процесс обмена ключами тривиален, то в ИС, где количество пользователей составляет десятки и сотни управление ключами - серьезная проблема.

Под *ключевой информацией* понимается совокупность всех действующих в ИС ключей. Если не обеспечено достаточно надежное управление ключевой информацией, то завладев ею, злоумышленник получает неограниченный доступ ко всей информации.

Управление ключами - информационный процесс, включающий в себя три элемента:

- генерацию ключей;
- накопление ключей;
- распределение ключей.

Рассмотрим, как они должны быть реализованы для того, чтобы обеспечить безопасность ключевой информации в ИС.

Генерация ключей

В самом начале разговора о криптографических методах было сказано, что не стоит использовать неслучайные ключи с целью легкости их запоминания. В серьезных ИС используются специальные аппаратные и программные методы генерации случайных ключей. Как правило используют датчики ПСЧ. Однако степень случайности их генерации должна быть достаточно высоким. Идеальным генераторами являются устройства на основе «натуральных» случайных процессов. Например, появились серийные образцы генерации ключей на основе *белого радиошума*. Другим случайным математическим объектом являются десятичные знаки иррациональных чисел, например π или e, которые вычисляются с помощью стандартных математических методов.

В ИС со средними требованиями защищенности вполне приемлемы программные генераторы ключей, которые вычисляют ПСЧ как сложную функцию от текущего времени и (или) числа, введенного пользователем.

Накопление ключей

Под *накоплением ключей* понимается организация их хранения, учета и удаления.

Поскольку ключ является самым привлекательным для злоумышленника объектом, открывающим ему путь к конфиденциальной информации, то вопросам накопления ключей следует уделять особое внимание.

Секретные ключи никогда не должны записываться в явном виде на носителе, который может быть считан или скопирован.

В достаточно сложной ИС один пользователь может работать с большим объемом ключевой информации, и иногда даже возникает необходимость организации мини-баз данных по ключевой информации. Такие базы данных отвечают за принятие, хранение, учет и удаление используемых ключей.

Итак, каждая информация об используемых ключах должна храниться в зашифрованном виде. Ключи, зашифровывающие ключевую информацию называются *мастер-ключами*. Желательно, чтобы мастер-ключи каждый пользователь знал наизусть, и не хранил их вообще на каких-либо материальных носителях.

Очень важным условием безопасности информации является периодическое обновление ключевой информации в ИС. При этом переназначаться должны как обычные ключи, так и мастер-ключи. В особо ответственных ИС обновление ключевой информации желательно делать ежедневно.

Вопрос обновления ключевой информации связан и с третьим элементом управления ключами - распределением ключей.

Распределение ключей

Распределение ключей - самый ответственный процесс в управлении ключами. К нему предъявляются два требования:

1. Оперативность и точность распределения
2. Скрытность распределяемых ключей.

В последнее время заметен сдвиг в сторону использования криптосистем с открытым ключом, в которых проблема распределения ключей отпадает. Тем не менее распределение ключевой информации в ИС требует новых эффективных решений.

Распределение ключей между пользователями реализуются двумя разными подходами:

1. *Путем создания одного ли нескольких центров распределения ключей.* Недостаток такого подхода состоит в том, что в центре распределения известно, кому и какие ключи назначены и это позволяет читать все сообщения, циркулирующие в ИС. Возможные злоупотребления существенно влияют на защиту.

2. *Прямой обмен ключами* между пользователями информационной системы. В этом случае проблема состоит в том, чтобы надежно удостоверить подлинность субъектов.

В обоих случаях должна быть гарантирована подлинность сеанса связи. Это можно обеспечить двумя способами:

1. *Механизм запроса-ответа*, который состоит в следующем. Если пользователь А желает быть уверенным, что сообщения который он получает от В, не являются ложными, он включает в посылаемое для В сообщение непредсказуемый элемент (запрос). При ответе пользователь В должен выполнить некоторую операцию над этим элементом (например, добавить 1). Это невозможно осуществить заранее, так как не известно, какое случайное число придет в за-

просе. После получения ответа с результатами действий пользователь А может быть уверен, что сеанс является подлинным. Недостатком этого метода является возможность установления хотя и сложной закономерности между запросом и ответом.

2. *Механизм отметки времени («временной штемпель»).* Он подразумевает фиксацию времени для каждого сообщения. В этом случае каждый пользователь ИС может знать, насколько «старым» является пришедшее сообщение.

В обоих случаях следует использовать шифрование, чтобы быть уверенным, что ответ послан не злоумышленником и штемпель отметки времени не изменен.

При использовании отметок времени встает проблема допустимого временного интервала задержки для подтверждения подлинности сеанса. Ведь сообщение с «временным штемпелем» в принципе не может быть передано мгновенно. Кроме этого компьютерные часы получателя и отправителя не могут быть абсолютно синхронизированы. Какое запаздывание «штемпеля» считать подозрительным.

Поэтому в реальных ИС, например в системах оплаты кредитных карточек используется именно второй механизм установления подлинности и защиты от подделок. Используемый интервал составляет от одной до нескольких минут. Большое число известных способов кражи электронных денег, основано на «вклинивании» в этот промежуток с подложными запросами на снятии денег.

Для обмена ключами можно использовать криптосистемы с открытым ключом, используя тот же алгоритм RSA.

Но весьма эффективным оказался алгоритм Диффи-Хелмана, позволяющий двум пользователям без посредников обменяться ключом, который может быть использован затем для симметричного шифрования.

Алгоритм Диффи-Хеллмана

Диффи и Хелман предложили для создания криптографических систем с открытым ключом *функцию дискретного возведения в степень*.

Необратимость преобразования в этом случае обеспечивается тем, что достаточно легко вычислить показательную функцию в конечном поле Галуа состоящим из p элементов. (p - либо простое число, либо простое в любой степени). Вычисление же логарифмов в таких полях - значительно более трудоемкая операция.

Если $y=\alpha^x$, $1<x<p-1$, где - фиксированный элемент поля $GF(p)$, то $x=\log_{\alpha} y$ над $GF(p)$. Имея x, легко вычислить y. Для этого потребуется 2 $\ln(x+y)$ операций умножения.

Обратная задача вычисления x из y будет достаточно сложной. Если p выбрано достаточно правильно, то извлечение логарифма потребует вычислений, пропорциональных

$$L(p) = exp\ \{ (\ln p\ \ \ln \ln p)^{0.5} \}$$

Для обмена информацией первый пользователь выбирает случайное число x_1, равновероятное из целых 1...p-1. Это число он держит в секрете, а другому пользователю посылает число

$$y_1 = \alpha^x \bmod p$$

Аналогично поступает и второй пользователь, генерируя x_2 и вычислив y_2, отправляя его первому пользователю. В результате этого они могут вычислять $k_{12} = \alpha^{x_1 x_2} \bmod p$.

Для того, чтобы вычислить k_{12}, первый пользователь возводит y_2 в степень x_1. То же делает и второй пользователь. Таким образом, у обоих пользователей оказывается общий ключ k_{12}, который можно использовать для шифрования информации обычными алгоритмами. В отличие от алгоритма RSA, данный алгоритм не позволяет шифровать собственно информацию.

Не зная x_1 и x_2, злоумышленник может попытаться вычислить k_{12}, зная только перехваченные y_1 и y_2. Эквивалентность этой проблемы проблеме вычисления дискретного логарифма есть главный и открытый вопрос в системах с открытым ключом. Простого решения до настоящего времени не найдено. Так, если для прямого преобразования 1000-битных простых чисел требуется 2000 операций, то для обратного преобразования (вычисления логарифма в поле Галуа) - потребуется около 10^{30} операций.

Как видно, при всей простоте алгоритма Диффи-Хелмана, вторым его недостатком по сравнению с системой RSA является отсутствие гарантированной нижней оценки трудоемкости раскрытия ключа.

Кроме того, хотя описанный алгоритм позволяет обойти проблему скрытой передачи ключа, необходимость аутентификации остается. Без дополнительных средств, один из пользователей не может быть уверен, что он обменялся ключами именно с тем пользователем, который ему нужен. Опасность имитации в этом случае остается.

В качестве обобщения сказанного о распределении ключей следует сказать следующее. Задача управления ключами сводится к поиску такого протокола распределения ключей, который обеспечивал бы:

- возможность отказа от центра распределения ключей;
- взаимное подтверждение подлинности участников сеанса;
- подтверждение достоверности сеанса механизмом запроса-ответа, использование для этого программных или аппаратных средств;
- использование при обмене ключами минимального числа сообщений.

Проблемы и перспективы криптографических систем

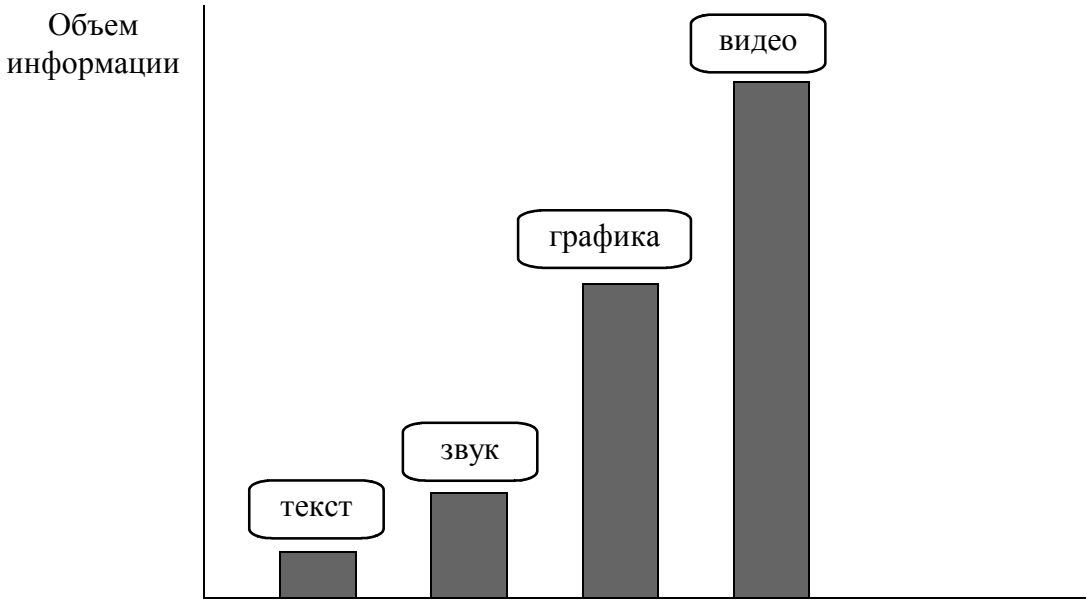
Шифрование больших сообщений и потоков данных

Эта проблема появилась сравнительно недавно с появлением средств *мультимедиа* и сетей с высокой пропускной способностью, обеспечивающих передачу мультимедийных данных.

До сих пор говорилось о защите сообщений. При этом под ними подразумевалась скорее некоторая текстовая или символическая информация. Однако в современных ИС и информационных системах начинают применяться технологии, которые требуют передачи существенно больших объемов данных. Среди таких технологий:

- факсимильная, видео и речевая связь;
- голосовая почта;
- системы видеоконференций.

Объем передаваемой информации разных типов можно представить на условной диаграмме.



Так как передача оцифрованной звуковой, графической и видеоинформации во многих случаях требует конфиденциальности, то возникает проблема шифрования огромных информационных массивов. Для интерактивных систем типа телеконференций, ведения аудио или видеосвязи, такое шифрование должно осуществляться в реальном масштабе времени и по возможности быть «прозрачным» для пользователей.

Это немыслимо без использования современных технологий шифрования.

Наиболее распространенным является *потоковое шифрование* данных. Если в описанных ранее криптосистемах предполагалось, что на входе имеется

некоторое конечное сообщение, к которому и применяется криптографический алгоритм, то в системах с потоковым шифрованием принцип другой.

Система защиты не ждет, когда закончится передаваемое сообщение, а сразу же осуществляет его шифрование и передачу.

Потоковое шифрование данных

Наиболее очевидным является побитовое сложение входящей последовательности (сообщения) с некоторым бесконечным или периодическим ключом, получаемым например от генератора ПСП[18]. Примером стандарта потокового шифрования является RC4, разработанный Ривестом. Однако, технические подробности этого алгоритма держатся в секрете[19].

Другим, иногда более эффективным методом потокового шифрования является *шифрование блоками*. Т.е. накапливается фиксированный объем информации (блок), а затем преобразованный некоторым криптографическим методом передается в канал связи.

Использование «блуждающих ключей»

Как было неоднократно отмечено, проблема распределения ключей является наиболее острой в крупных информационных системах. Отчасти эта проблема решается (а точнее снимается) за счет использования открытых ключей. Но наиболее надежные криптосистемы с открытым ключом типа RSA достаточно трудоемки, а для шифрования мультимедийных данных и вовсе не пригодны.

Оригинальные решения проблемы « блуждающих ключей» активно разрабатываются специалистами. Эти системы являются некоторым компромиссом между системами с открытыми ключами и обычными алгоритмами, для которых требуется наличие одного и того же ключа у отправителя и получателя.

Идея метода достаточно проста.

После того, как ключ использован в одном сеансе по некоторому правилу он сменяется другим. Это правило должно быть известно и отправителю, и получателю. Зная правило, после получения очередного сообщения получатель тоже меняет ключ. Если правило смены ключей аккуратно соблюдается и от-

[18] Отчасти это метод похож на гаммирование и информацию о способах генерации ПСП можно почерпнуть из соответствующей главы. Но важным отличием потокового шифрования является то, что шифрованию подвергаются не символы сообщения, а отдельные биты.

[19] Данный алгоритм является собственностью RSA Data Security, и на его экспорт правительством США наложены серьезные ограничения.

правителем и получателем, то в каждый момент времени они имеют одинаковый ключ. Постоянная смена ключа затрудняет раскрытие информации злоумышленником.

Основная задача в реализации этого метода - выбор эффективного правила смены ключей. Наиболее простой путь - генерация случайного списка ключей. Смена ключей осуществляется в порядке списка. Однако, очевидно список придется каким-то образом передавать.

Другой вариант - использование математических алгоритмов, основанных на так называемых *перебирающих последовательностях*. На множестве ключей путем одной и той же операции над элементом получается другой элемент. Последовательность этих операций позволяет переходить от одного элемента к другому, пока не будет перебрано все множество.

Наиболее доступным является использование полей Галуа. За счет возведения в степень порождающего элемента можно последовательно переходить от одного числа к другому. Эти числа принимаются в качестве ключей.

Ключевой информацией в данном случае является исходный элемент, который перед началом связи должен быть известен и отправителю и получателю.

Надежность таких методов должна быть обеспечена с учетом известности злоумышленнику используемого правила смены ключей.

Интересной и перспективной задачей является реализация метода «блуждающих ключей» не для двух абонентов, а для достаточно большой сети, когда сообщения пересылаются между всеми участниками.

Шифрование, кодирование и сжатие информации

Эти три вида преобразования информации используются в разных целях, что можно представить в таблице.

Вид преобразования	Цель	Изменение объема информации после преобразования.
Шифрование	• передача конфиденциальной информации; • обеспечение аутентификации и защиты от преднамеренных изменений;	обычно не изменяется, увеличивается лишь в цифровых сигнатурах и подписях
Помехоустойчивое кодирование	• защита от искажения помехами в каналах связи	увеличивается
Сжатие (компрессия)	• сокращение объема передаваемых или хранимых данных	уменьшается

Как видно эти три вида преобразования информации отчасти дополняют друг друга и их комплексное использование поможет эффективно использовать каналы связи для надежной защиты предаваемой информации.

Особенно интересным представляется *возможность объединения методов кодирования и шифрования*. Можно утверждать, что по сути кодирование - это элементарное шифрование, а шифрование - это элементарное помехоустойчивое кодирование.

Другая возможность - *комбинирование алгоритмов шифрования и сжатия информации*. Задача сжатия состоит в том, чтобы преобразовать сообщение в пределах одного и того же алфавита таким образом, чтобы его длина (количество букв алфавита) стала меньше, но при этом сообщение можно было восстановить без использования какой-то дополнительной информации. Наиболее популярные алгоритмы сжатия - RLE, коды Хаффмана, алгоритм Лемпеля-Зива. Для сжатия графической и видеоинформации используются алгоритмы JPEG и MPEG.

Главное достоинство алгоритмов сжатия с точки зрения криптографии состоит в том, что они изменяют статистику входного текста в сторону ее выравнивания[20]. Так, в обычном тексте, сжатом с помощью эффективного алгоритма все символы имеют одинаковые частотные характеристики и даже использование простых системы шифрования сделают текст недоступным для криптоанализа.

[20] Принципиально важно с точки зрения криптостойкости, чтобы сначала осуществлялось сжатие информации а потом шифрование, но не наоборот.

Разработка и реализация таких универсальных методов - перспектива современных информационных систем[21].

[21] Так, в криптографическом пакете PGP перед шифрованием информации происходит ее сжатие по алгоритму, лицензированному у PKWARE.

Реализация криптографических методов

Проблема реализации методов защиты информации имеет два аспекта:

- разработку средств, реализующих криптографические алгоритмы,
- методику использования этих средств.

Каждый из рассмотренных криптографических методов могут быть реализованы либо *программным*, либо *аппаратным* способом.

Возможность программной реализации обуславливается тем, что все методы криптографического преобразования формальны и могут быть представлены в виде конечной алгоритмической процедуры.

При аппаратной реализации все процедуры шифрования и дешифрования выполняются специальными электронными схемами. Наибольшее распространение получили модули, реализующие комбинированные методы.

При этом непременным компонентов всех аппаратно реализуемых методов является *гаммирование*. Это объясняется тем, что метод гаммирования удачно сочетает в себе высокую криптостойкость и простоту реализации.

Наиболее часто в качестве генератора используется широко известный регистр сдвига с обратными связями (линейными или нелинейными). Минимальный период порождаемой последовательности равен 2^N-1 бит. Для повышения качества генерируемой последовательности можно предусмотреть специальный блок управления работой регистра сдвига. Такое управление может заключаться, например, в том, что после шифрования определенного объема информации содержимое регистра сдвига циклически изменяется.

Другая возможность улучшения качества гаммирования заключается в использовании нелинейных обратных связей. При этом улучшение достигается не за счет увеличения длины гаммы, а за счет усложнения закона ее формирования, что существенно усложняет криптоанализ.

Большинство зарубежных серийных средств шифрования основано на американском стандарте DES. Отечественные же разработки, такие как, например, устройство КРИПТОН, использует отечественный стандарт шифрования.

Основным достоинством программных методов реализации защиты является их гибкость, т.е. возможность быстрого изменения алгоритмов шифрования.

Основным же недостатком программной реализации является существенно меньшее быстродействие по сравнению с аппаратными средствами (примерно в 10 раз).

В последнее время стали появляться комбинированные средства шифрования, так называемые программно-аппаратные средства. В этом случае в компьютере используется своеобразный «криптографический сопроцессор»[22] - вычислительное устройство, ориентированное на выполнение криптографических операций (сложение по модулю, сдвиг и т.д.). Меняя программное обеспе-

[22] А то и просто специализированный шифровальный микропроцессор как, например, Clipper/

чения для такого устройства, можно выбирать тот или иной метод шифрования. Такой метод объединяет в себе достоинства программных и аппаратных методов.

Таким образом, выбор типа реализации криптозащиты для конкретной ИС в существенной мере зависит от ее особенностей и должен опираться на всесторонний анализ требований, предъявляемых к системе защиты информации.

Заключение

В книге сделан обзор наиболее распространенных в настоящее время методов криптографической защиты информации.

Выбор для конкретных ИС должен быть основан на глубоком анализе слабых и сильных сторон тех или иных методов защиты. Обоснованный выбор той или иной системы защиты в общем-то должен опираться на какие-то *критерии эффективности*. К сожалению, до сих пор не разработаны подходящие методики оценки эффективности криптографических систем.

Наиболее простой критерий такой эффективности - *вероятность раскрытия ключа* или *мощность множества ключей (M)*. По сути это то же самое, что и *криптостойкость*. Для ее численной оценки можно использовать также и сложность раскрытия шифра путем перебора всех ключей.

Однако, этот критерий не учитывает других важных *требований к криптосистемам*:

- невозможность раскрытия или осмысленной модификации информации на основе анализа ее структуры,
- совершенство используемых протоколов защиты,
- минимальный объем используемой ключевой информации,
- минимальная сложность реализации (в количестве машинных операций), ее стоимость,
- высокая оперативность.

Желательно конечно использование некоторых интегральных показателей, учитывающих указанные факторы.

Для учета стоимости, трудоемкости и объема ключевой информации можно использовать удельные показатели - отношение указанных параметров к мощности множества ключей шифра.

Часто более эффективным при выборе и оценке криптографической системы является использование экспертных оценок и имитационное моделирование.

В любом случае выбранный комплекс криптографических методов должен сочетать как удобство, гибкость и оперативность использования, так и надежную защиту от злоумышленников циркулирующей в ИС информации.