Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Кафедра сетей и устройств телекоммуникаций

КРИПТОГРАФИЧЕСКОЕ КОДИРОВАНИЕ ИНФОРМАЦИИ

Методические указания
к лабораторной работе
по дисциплинам «Основы защиты информации»
и «Криптографическая защита информации в телекоммуникациях»
для студентов специальности «Сети телекоммуникаций»
дневной, вечерней и заочной форм обучения

В 3-х частях

Часть 2

ШИФРОВАНИЕ АЛГОРИТМАМИ DES и RSA

Минск 2003

УДК 621.391.2(075.8)
ББК 32.811 я 73
К 82

Составители:
В.Ф. Голиков, А.В. Курилович

Криптографическое кодирование информации: Метод. указания к К 82 лабораторной работе по дисциплинам «Основы защиты информации» и «Криптографическая защита информации в телекоммуникациях» для студентов специальности «Сети телекоммуникаций» дневной, вечерней и заочной форм обучения. В 3 ч. Ч. 2: Шифрование алгоритмами DES и RSA / Сост. В.Ф. Голиков, А.В. Курилович. — Мн.: БГУИР, 2003. — 24 с.: ил.

Часть 2 методических указаний посвящена изучению алгоритма DES шифрования информации в криптосистемах симметричного типа и алгоритма RSA шифрования информации в криптосистемах асимметричного типа.

УДК 621.391.2(075.8)
ББК 32.811 я 73

Часть 1 настоящих методических указаний издана в БГУИР в 2003 г.

1. ШИФРОВАНИЕ АЛГОРИТМОМ DES

1.1. ЦЕЛЬ РАБОТЫ

Закрепление теоретических знаний по алгоритму DES шифрования информации в криптосистемах симметричного типа.

1.2. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

1.2.1. Обобщенная схема алгоритма DES

Алгоритм DES использует комбинацию подстановок и перестановок. DES осуществляет шифрование 64-битовых блоков данных с помощью 64-битового ключа, в котором значащими являются 56 бит (остальные 8 бит — проверочные биты для контроля на четность). Дешифрование в DES является операцией, обратной шифрованию, и выполняется путем повторения операций шифрования в обратной последовательности. Обобщенная схема процесса

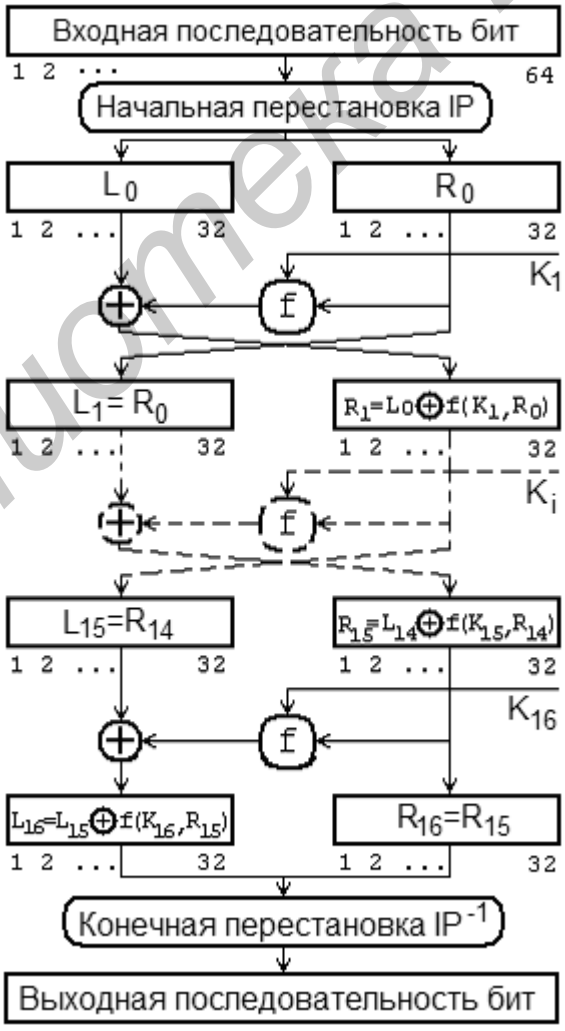

Рис. 1.1. Обобщенная схема шифрования в алгоритме DES

шифрования в алгоритме DES (рис. 1.1) заключается в начальной перестановке бит 64-битового блока, шестнадцати циклах шифрования и, наконец, в конечной перестановке бит.

Следует отметить, что все приводимые таблицы являются стандартными и должны включаться в реализацию алгоритма DES в неизменном виде.

Все перестановки и коды в таблицах подобраны разработчиками таким образом, чтобы максимально затруднить процесс взлома шифра.

Пусть из файла исходного текста считан очередной 64-битовый блок $\mathrm{T_O}$. Этот блок преобразуется с помощью матрицы начальной перестановки IP (табл. П1).

Биты входного блока T (64 бита) переставляются в соответствии с матрицей IP: бит 58 входного блока T становится битом 1, бит 50 — битом 2 и т.д. Эту перестановку можно описать выражением $\mathrm{T_O} = \mathrm{IP}(\mathrm{T})$. Полученная последовательность бит $\mathrm{T_O}$ разделяется на две последовательности: $\mathrm{L_0}$ — левые, или старшие, биты, $\mathrm{R_0}$ — правые, или младшие, биты — каждая из которых содержит 32 бита.

Затем выполняется итеративный процесс шифрования, состоящий из 16 шагов (циклов). Пусть T_i — результат i-й итерации: $\mathrm{T}_i = \mathrm{L}_i\mathrm{R}_i$, где $\mathrm{L}_i = t_1 t_2 \ldots t_{32}$ (первые 32 бита); $\mathrm{R}_i = t_{33} t_{34} \ldots t_{64}$ (последние 32 бита). Тогда результат i-й итерации описывается следующими формулами:

$$\mathrm{L}_i = \mathrm{R}_{i-1},\ i = 1, 2, \ldots, 16;$$

$$\mathrm{R}_i = \mathrm{L}_{i-1} \oplus f\left(\mathrm{R}_{i-1}, \mathrm{K}_i\right),\ i = 1, 2, \ldots, 16.$$

Функция f называется функцией шифрования. Ее аргументами являются последовательность R_{i-1}, получаемая на предыдущем шаге итерации, и 48-битовый ключ K_i, который является результатом преобразования 64-битового ключа шифра K. (Подробнее функция шифрования f и алгоритм получения ключа K описаны ниже.)

На последнем шаге итерации получают последовательности $\mathrm{R_{16}}$ и $\mathrm{L_{16}}$ (без перестановки местами), которые конкатенируются в 64-битовую последовательность $\mathrm{R_{16}L_{16}}$.

По окончании шифрования осуществляется восстановление позиций бит с помощью матрицы обратной перестановки IP^{-1} (табл. П2).

Процесс расшифровывания данных является инверсным по отношению к процессу шифрования. Все действия должны быть выполнены в обратном порядке. Это означает, что расшифровываемые данные сначала переставляются в соответствии с матрицей IP^{-1}, а затем над последовательностью бит $\mathrm{R_{16}L_{16}}$ выполняются те же действия, что и в процессе шифрования, но в обратном порядке.

Итеративный процесс расшифровывания может быть описан следующими формулами:

$$R_{i-1} = L_i, i = 1, 2, \ldots, 16;$$

$$L_{i-1} = R_i \oplus f(L_i, K_i), i = 1, 2, \ldots, 16.$$

Таким образом, для процесса расшифровывания с переставленным входным блоком $R_{16}L_{16}$ на первой итерации используется ключ K_{16}, на второй итерации — K_{15} и т.д. На 16-й итерации используется ключ K_1. На последнем шаге итерации будут получены последовательности L_0 и R_0, которые конкатенируются в 64-битовую последовательность L_0R_0. Затем в этой последовательности 64 бита переставляются в соответствии с матрицей IP. Результат такого преобразования — исходная последовательность бит (расшифрованное 64-битовое значение).

1.2.2. Реализация функции шифрования

Схема вычисления функции шифрования $f(R_{i-1}, K_i)$ показана на рис. 1.2.

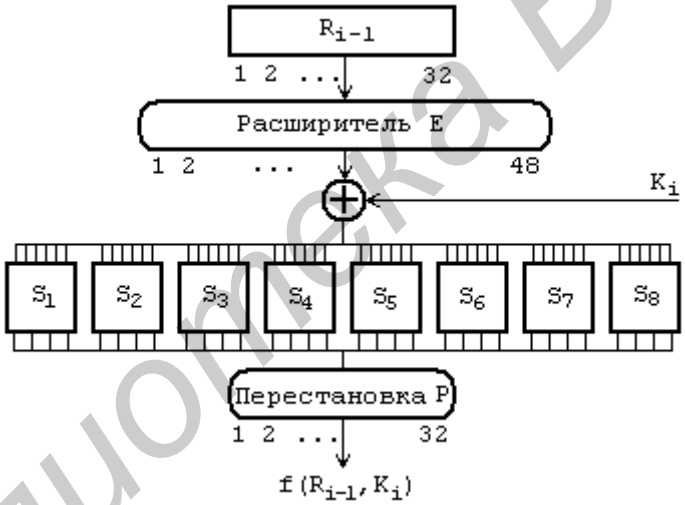

Рис. 1.2. Схема вычисления функции шифрования f

Для вычисления значения функции f используются:

- функция E (расширение 32 бит до 48);
- функция $S_1, S_2, \ldots, S_8$ (преобразование 6-битового числа в 4-битовое);
- функция P (перестановка бит в 32-битовой последовательности).

Приведем определения этих функций.

Аргументами функции шифрования f являются R_{i-1} (32 бита) и K_i (48 бит). Результат функции $E(R_{i-1})$ есть 48-битовое число. Функция расширения E, выполняющая расширение 32 бит до 48 (принимает блок из 32 бит и порождает блок из 48 бит), определяется табл. П3.

В соответствии с табл. П3 первые три бита $E(R_{i-1})$ — это биты 32, 1 и 2, а последние — 31, 32 и 1. Полученный результат (обозначим его $E(R_{i-1})$)

складывается по модулю 2 с текущим значением ключа K_i и затем разбивается на восемь 6-битовых блоков $B_1, B_2, \ldots, B_8 = E(R_{i-1}) \oplus K_i$. Далее каждый из этих блоков используется как номер элемента в функциях — матрицах $S_1, S_2, \ldots, S_8$, содержащих 4-битовые значения (табл. П8).

Следует отметить, что выбор элемента в матрице S осуществляется достаточно оригинальным образом. Пусть на вход матрицы S поступает 6-битовый блок $B_j = b_1b_2b_3b_4b_5b_6$, тогда 2-битовое число b_1b_6 указывает номер строки матрицы, а 4-битовое число $b_2b_3b_4b_5$ — номер столбца. Например, если на вход матрицы S_1 поступает 6-битовый блок $B_1 = b_1b_2b_3b_4b_5b_6 = 100110_{(2)}$, то 2-битовое число $b_1b_6 = 10_{(2)} = 2_{(2)}$ указывает строку с номером 2 матрицы S_1, а 4-битовое число $b_2b_3b_4b_5 = 0011_{(2)} = 3_{(10)}$ указывает столбец с номером 3 матрицы S_1. Это означает, что в матрице S_1 блок $B_1 = 100110$ выбирает элемент на пересечении строки с номером 2 и столбца с номером 3, т.е. элемент $8_{(10)} = 1000_{(2)}$. Совокупность 6-битовых блоков $B_1, B_2, \ldots, B_8$ обеспечивает выбор 4-битового элемента в каждой из матриц $S_1, S_2, \ldots, S_8$.

В результате получаем $S_1(B_1), S_2(B_1), \ldots, S_8(B_1)$, т.е. 32-битовый блок (поскольку матрицы S содержат 4-битовые элементы). Этот 32-битовый блок преобразуется с помощью функции перестановки бит P (табл. П4).

Таким образом, функция шифрования

$$f(R_{i-1}, K_i) = P(S_1(B_1), \ldots, S_8(B_1)).$$

1.2.3. Алгоритм вычисления ключей

Как нетрудно заметить, на каждой итерации используется новое значение ключа K_i (длиной 48 бит). Новое значение ключа K_i вычисляется из начального ключа K (рис. 1.3).

Ключ K представляет собой 64-битовый блок с 8 битами контроля по четности, расположенными в позициях 8, 16, 24, 32, 40, 48, 56, 64. Для удаления контрольных бит и подготовки ключа к работе используется функция G первоначальной подготовки ключа (табл. П5).

Табл. П5 разделена на две части. Результат преобразования $G(K)$ разбивается на две половины C_0 и D_0, по 28 бит каждая. Первые четыре строки матрицы G определяют, как выбираются биты последовательности C (первым битом C_0 будет бит 57 ключа шифра, затем бит 49 и т.д., а последними битами — биты 44 и 36 ключа).

Следующие четыре строки матрицы G определяют, как выбираются биты последовательности D_0 (т.е. последовательность D_0 будет состоять из бит 63, 55, 47, ..., 12, 4 ключа шифра).

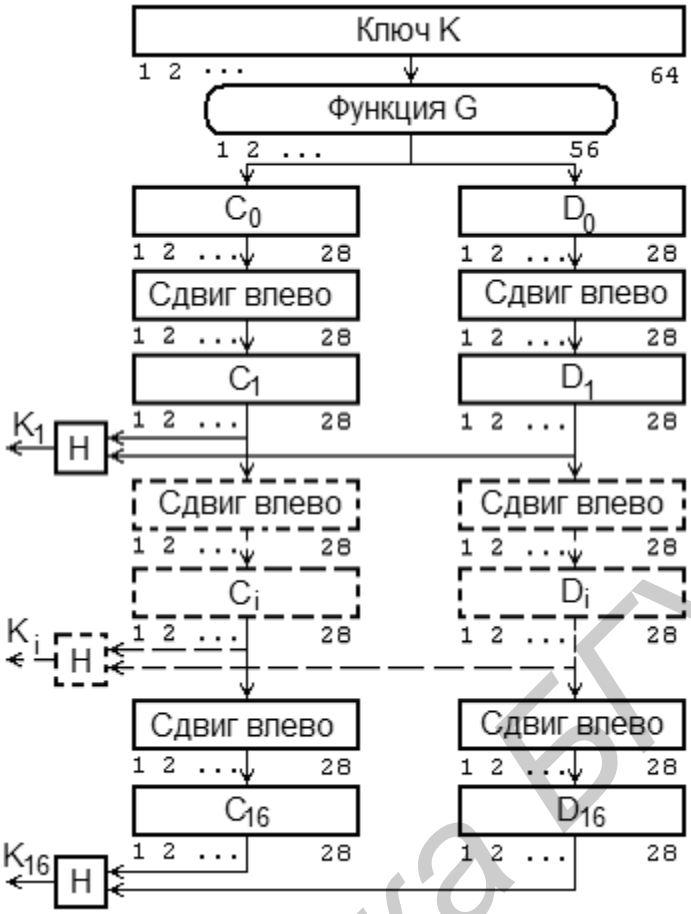

Рис. 1.3. Схема алгоритма вычисления ключей K_i

Как видно из табл. П5, для генерации последовательностей C_0 и D_0 не используются биты 8, 16, 24, 32, 40, 48, 56 и 64 ключа шифра. Эти биты не влияют на шифрование и могут служить для других целей (например, для контроля по четности). Таким образом, в действительности ключ шифра является 56-битовым.

После определения C_0 и D_0 рекурсивно определяются C_i и D_i, $i = 1, 2, \ldots, 16$. Для этого применяются операции циклического сдвига влево на один или два бита в зависимости от номера шага итерации, как показано в табл. П7.

Операции сдвига выполняются для последовательностей C_i и D_i независимо. Например, последовательность C_3 получается посредством циклического сдвига влево на две позиции последовательности C_2, а последовательность D_3 — посредством сдвига влево на две позиции последовательности D_2, C_{16} и D_{16} получаются из C_{15} и D_{15} посредством сдвига влево на одну позицию.

Ключ K_i, определяемый на каждом шаге итерации, есть результат выбора конкретных бит из 56-битовой последовательности C_iD_i и их перестановки.

Другими словами, ключ $K_i = H(C_i D_i)$, где функция H определяется матрицей, завершающей обработку ключа (табл. П6).

Как следует из табл. П6, первым битом ключа K_i будет 14-й бит последовательности $C_i D_i$, вторым — 17-й бит, 47-м битом ключа K_i будет 29-й бит $C_i D_i$, а 48-м битом — 32-й бит $C_i D_i$.

1.2.4. Основные режимы работы алгоритма DES

Чтобы воспользоваться алгоритмом DES для решения разнообразных криптографических задач, разработаны четыре рабочих режима:

- электронная кодовая книга ECB (Electronic Code Book);
- сцепление блоков шифра CBC (Cipher Block Chaining);
- обратная связь по шифротексту CPB (Cipher FeedBack);
- обратная связь по выходу OFB (Output FeedBack).

1.2.4.1. Режим «Электронная кодовая книга»

Длинный файл разбивают на 64-битовые отрезки (блоки) по 8 байт. Каждый из этих блоков шифруют независимо с использованием одного и того же ключа шифрования (рис. 1.4).

Основное достоинство — простота реализации. Недостаток — относительно слабая устойчивость против квалифицированных криптоаналитиков. Из-за фиксированного характера шифрования при ограниченной длине блока 64 бита возможно проведение криптоанализа «со словарем». Блок такого размера может повториться в сообщении вследствие большой избыточности в тексте на естественном языке. Это приводит к тому, что идентичные блоки открытого текста в сообщении будут представлены идентичными блоками шифротекста, что дает криптоаналитику некоторую информацию о содержании сообщения.

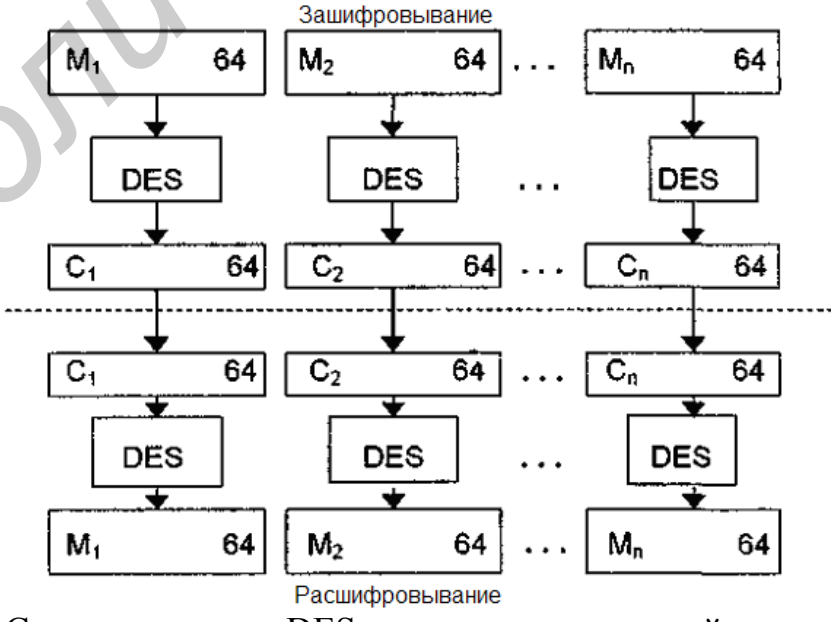

Рис. 1.4. Схема алгоритма DES в режиме электронной кодовой книги

1.2.4.2. Режим «Сцепление блоков шифра»

В этом режиме исходный файл M разбивается на 64-битовые блоки: $M=M_1M_2\ldots M_n$. Первый блок M_1 складывается по модулю 2 с 64-битовым начальным вектором IV, который меняется ежедневно и держится в секрете (рис. 1.5). Полученная сумма затем шифруется с использованием ключа DES, известного и отправителю, и получателю информации. Полученный 64-битовый шифр C_1 складывается по модулю 2 со вторым блоком текста, результат шифруется и получается второй 64-битовый шифр C_2 и т.д. Процедура повторяется до тех пор, пока не будут обработаны все блоки текста.

Таким образом, для всех $i=1\ldots n$ (n — число блоков) результат шифрования C определяется следующим образом: $C_i=\mathrm{DES}(M_i\oplus C_{i-1})$, где $C_0=\mathrm{IV}$ — начальное значение шифра, равное начальному вектору (вектору инициализации).

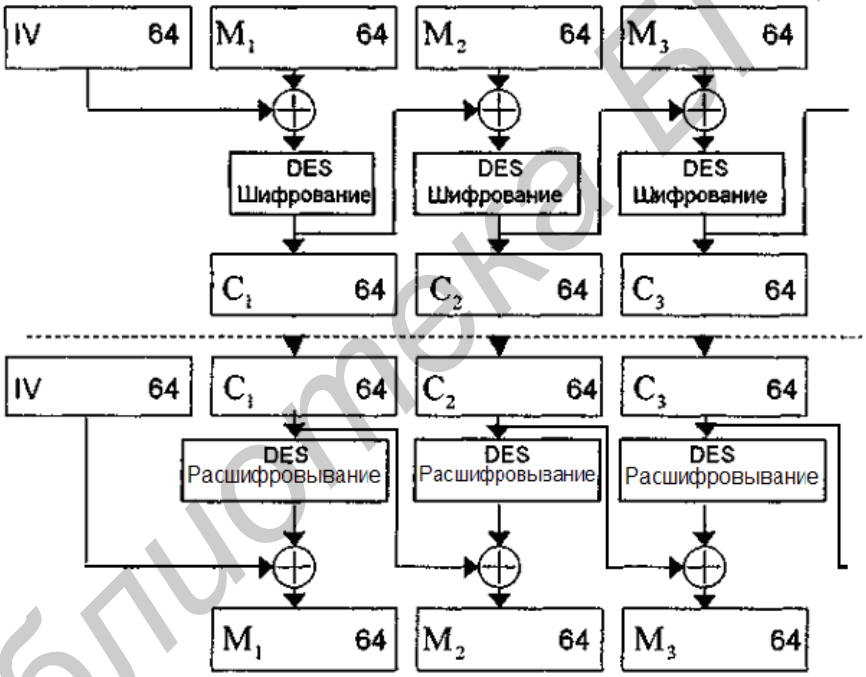

Рис. 1.5. Схема алгоритма DES в режиме сцепления блоков шифра

Очевидно, что последний 64-битовый блок шифротекста является функцией секретного ключа, начального вектора и каждого бита открытого текста независимо от его длины. Этот блок шифротекста называют кодом аутентификации сообщения (КАС).

Код КАС может быть легко проверен получателем, владеющим секретным ключом и начальным вектором, путем повторения процедуры, выполненной отправителем. Посторонний, однако, не может осуществить генерацию КАС, который воспринялся бы получателем как подлинный, чтобы добавить его к ложному сообщению, либо отделить КАС от истинного сообщения для использования его с измененным или ложным сообщением.

Достоинство данного режима в том, что он не позволяет накапливаться ошибкам при передаче.

Блок M_i является функцией только C_{i-1} и C_i. Поэтому ошибка при передаче приведет к потере только двух блоков исходного текста.

1.2.4.3. Режим «Обратная связь по шифротексту»

В этом режиме размер блока может отличаться от 64 бит (рис 1.6). Файл, подлежащий шифрованию (расшифровыванию), считывается последовательными блоками длиной k бит ($k = 1 \ldots 64$).

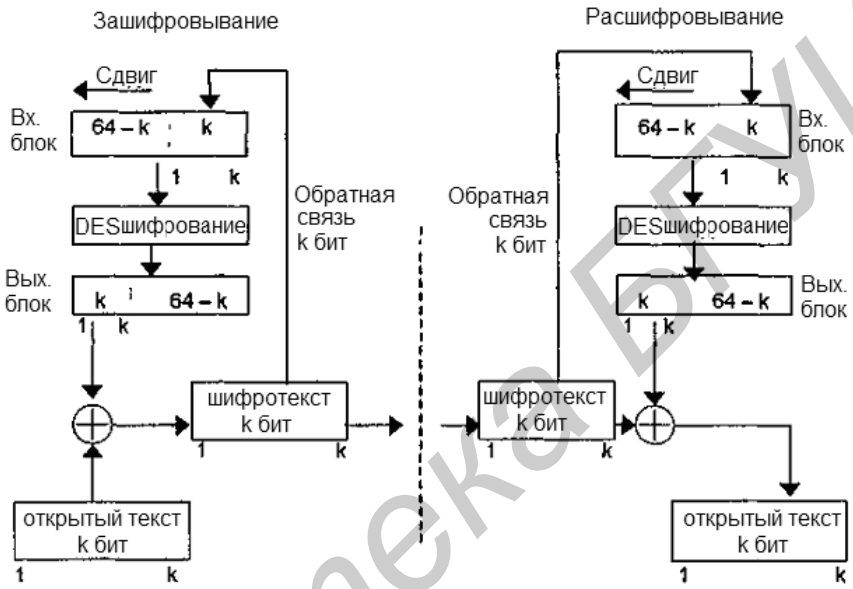

Рис. 1.6. Схема алгоритма DES в режиме обратной связи по шифротексту

Входной блок (64-битовый регистр сдвига) вначале содержит вектор инициализации, выровненный по правому краю. Предположим, что в результате разбиения на блоки мы получили n блоков длиной k бит каждый (остаток дописывается нулями или пробелами). Тогда для любого $i = 1 \ldots n$ блок шифротекста $C_i = M_i \oplus P_{i-1}$, где P_{i-1} обозначает k старших бит предыдущего зашифрованного блока.

Обновление сдвигового регистра осуществляется путем удаления его старших k бит и записи C_i в регистр. Восстановление зашифрованных данных также выполняется относительно просто: P_{i-1} и C_i вычисляются аналогичным образом и $M_i = C_i \oplus P_{i-1}$.

1.2.4.4. Режим «Обратная связь по выходу»

Этот режим тоже использует переменный размер блока и сдвиговый регистр, инициализируемый так же, как в режиме СРВ, а именно — входной блок вначале содержит вектор инициализации IV, выровненный по правому краю (рис. 1.7). При этом для каждого сеанса шифрования данных необходимо

использовать новое начальное состояние регистра, которое должно пересылаться по каналу открытым текстом.

Положим $M=M_1M_2\ldots M_n$ для всех $i=1\ldots n$ $C_i=M_i\oplus P_i$, где P_i — старшие k бит операции $DES(C_{i-1})$. Отличие от режима обратной связи по шифротексту состоит в методе обновления сдвигового регистра.

Это осуществляется путем отбрасывания старших k бит и дописывания справа P_i.

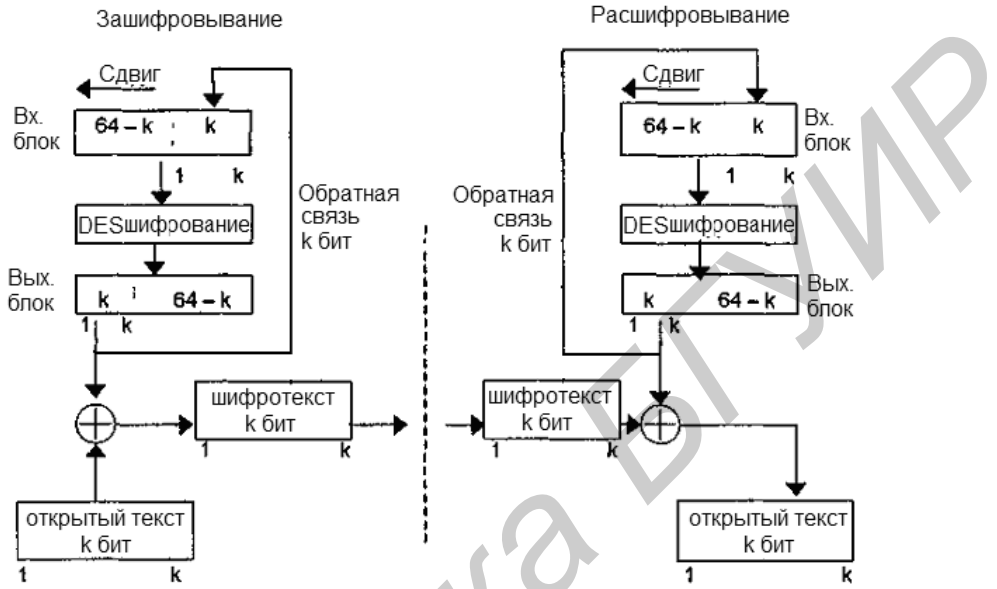

Рис. 1.7. Схема алгоритма DES в режиме обратной связи по выходу

1.2.4.5. Комбинирование блочных алгоритмов

В настоящее время блочный алгоритм DES считается относительно безопасным алгоритмом шифрования. Он подвергался тщательному криптоанализу в течение 20 лет, и самым практичным способом его взламывания является метод перебора всех возможных вариантов ключа. Ключ DES имеет длину 56 бит, поэтому существует 2^{56} возможных вариантов такого ключа. Однако, с учетом вычислительных мощностей современных компьютеров, недалеко то время, когда поиск ключа DES методом полного перебора станет возможным для мощных в финансовом отношении государственных и коммерческих организаций.

Возникает естественный вопрос: нельзя ли использовать DES в качестве строительного блока для создания другого алгоритма с более длинным ключом?

В принципе, существует много способов комбинирования блочных алгоритмов для получения новых алгоритмов. Одним из таких способов комбинирования является многократное шифрование, т.е. использование блочного алгоритма несколько раз с разными ключами для шифрования одного и того же блока открытого текста. Двукратное шифрование блока открытого текста одним и тем же ключом не приводит к положительному результату. При

использовании одного и того же алгоритма такое шифрование не влияет на сложность криптоаналитической атаки полного перебора.

Рассмотрим эффективность двукратного шифрования блока открытого текста с помощью двух разных ключей. Сначала шифруют блок P ключом K_1, а затем получившийся шифротекст $E_{K_1}(P)$ шифруют ключом K_2. В результате двукратного шифрования получают криптограмму $C=E_{K_2}\left(E_{K_1}(P)\right)$.

Расшифровывание является обратным процессом: $P=D_{K_1}\left(D_{K_2}(C)\right)$.

Если блочный алгоритм обладает свойствами группы, то всегда найдется такой ключ K_3, что $C=E_{K_2}\left(E_{K_1}(P)\right)=E_{K_3}(P)$.

Если же блочный алгоритм не является группой, то результирующий двукратно шифрованный блок текста окажется намного сложнее для взламывания методом полного перебора вариантов. Вместо 2^n попыток, где n — длина ключа в битах, потребуется 2^{2n} попыток. В частности, если $n=64$, то двукратно зашифрованный блок текста потребует 2^{128} попыток для нахождения ключа.

Однако Р. Меркль и М. Хеллман показали на примере DES, что, используя метод «обмена времени на память» и криптоаналитическую атаку «встреча посередине», можно взломать такую схему двукратного шифрования за 2^n попыток. Правда, эта атака потребует очень большого объема памяти (для алгоритма с 56-битовым ключом потребуется 2^{56} 64-битовых блоков или 10^{17} бит памяти).

Более привлекательную идею предложил У. Тачмен. Суть этой идеи состоит в том, чтобы шифровать блок открытого текста P три раза с помощью двух ключей K_1 и K_2 (рис. 1.8). Процедура шифрования: $C=E_{K_1}\left(D_{K_2}\left(E_{K_1}(P)\right)\right)$, т.е. блок открытого текста P сначала шифруется ключом K_1, затем расшифровывается ключом K_2 и окончательно зашифровывается ключом K_1.

Этот режим иногда называют режимом EDE (encrypt — decrypt — encrypt). Введение в данную схему операции расшифровывания D_{K_2} позволяет обеспечить совместимость этой схемы со схемой однократного использования алгоритма DES. Если в схеме трехкратного использования DES выбрать все ключи одинаковыми, то эта схема превращается в схему однократного использования DES. Процедура расшифровывания выполняется в обратном порядке: $P=D_{K_1}\left(E_{K_2}\left(D_{K_1}(C)\right)\right)$, т.е. блок шифротекста C сначала расшифровывается ключом K_1, затем зашифровывается ключом K_2 и окончательно расшифровывается ключом K_1.

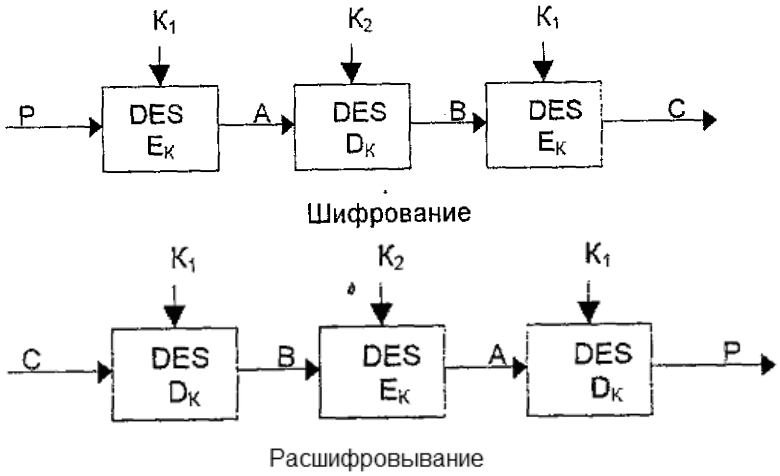

Рис.1.8. Схемы трехкратного применения алгоритма DES с двумя разными ключами

Если исходный блочный алгоритм имеет n-битовый ключ, то схема трехкратного шифрования имеет $2n$-битовый ключ. Чередование ключей K_1 и K_2 позволяет предотвратить криптоаналитическую атаку «встреча посредине».

При трехкратном шифровании можно применить три различных ключа. При этом возрастает общая длина результирующего ключа. Процедуры шифрования и расшифровывания описываются выражениями: $C=E_{K_3}\left(D_{K_2}\left(E_{K_1}(P)\right)\right)$, $P=D_{K_1}\left(E_{K_2}\left(D_{K_3}(C)\right)\right)$.

Трехключевой вариант имеет большую стойкость.

1.3. ПРЕДВАРИТЕЛЬНОЕ ЗАДАНИЕ

1.3.1. Изучите теоретическую часть.

1.3.2. Переведите число 3^{43} в двоичную систему счисления.

1.3.3. Пусть каждая из 16 первых букв русского алфавита (абвгдежзийклмноп) имеет четырехразрядный двоичный код, соответствующий ее номеру от 0 до 15, т.е. а — 0000_2, б — 0001_2, …, п — 1111_2. Составьте из этих букв произвольное сообщение, состоящее из 32 символов, затем разбейте полученное сообщение на блоки длиной 64 бита. Значения полученных блоков запишите в десятичной системе счисления.

1.3.4. Найдите состояние 28-разрядного двоичного регистра сдвига после циклического сдвига влево на 5 числа 179317333_{10}, предварительно записанного в регистр.

1.3.5. Найдите сумму по модулю 2 двух чисел 2244899301_{10} и 28973675_{10}.

1.4. ЛАБОРАТОРНОЕ ЗАДАНИЕ

1.4.1. Включите ПЭВМ.

1.4.2. Запустите программу des.exe на выполнение. Данная программа реализует алгоритм зашифровывания и расшифровывания данных по стандарту DES, DES в режиме обратной связи по шифротексту, комбинированный алгоритм DES. Входные и выходные данные работы этого алгоритма представляют собой 64-разрядные двоичные числа. Поэтому если данные, подлежащие обработке, представлены в другом виде, их предварительно переводят в двоичный вид и разбивают на блоки длиной по 64 бита. Для упрощения работы с 64-разрядными двоичными числами блоки представляются в десятичной системе счисления.

1.4.3. Нажмите кнопку «Начать выполнение работы». Выполнение работы состоит в том, что вы должны вручную найти значения в контрольных точках работы алгоритма. Для этого необходимо ввести полученное значение в соответствующее поле рабочего окна программы и нажать кнопку «Проверить». Если расчет выполнен правильно, происходит переход к следующей контрольной точке.

1.4.4. Программа имеет встроенный калькулятор, который выполняет перевод 64-разрядных чисел из двоичной системы счисления в десятичную и наоборот, а также осуществляет циклические сдвиги.

1.4.5. Исследование работы криптографического алгоритма DES.

1.4.5.1. Введите (в десятичной системе счисления) значение 64-разрядного ключа K.

1.4.5.2. Введите (в десятичной системе счисления) значение 64-разрядного блока открытого текста.

1.4.5.3. Руководствуйтесь инструкциями в рабочем окне программы.

1.4.5.4. Для того чтобы скопировать в буфер обмена результат какой-либо контрольной точки, достаточно один раз кликнуть по нему. Подтверждением того, что копирование произошло, является кратковременное изменение цветового фона.

1.4.6. Исследование DES в режиме обратной связи по шифротексту.

1.4.6.1. Введите (в десятичной системе счисления) значение 64-разрядного вектора инициализации IV.

1.4.6.2. Введите (в десятичной системе счисления) значения двух 10-битовых блоков открытого текста.

1.4.6.3. Для того чтобы зашифровать или расшифровать какой-либо 64-разрядный блок открытого текста, используя заданный ключ K, необходимо нажать кнопку «DES» в рабочем окне программы. В открывшемся окне необходимо ввести начальные данные (открытый текст или шифротекст, ключ K), выбрать направление работы (зашифровывание или расшифровывание) и нажать кнопку «Выполнить».

1.4.7. Исследование комбинированного алгоритма DES.

1.4.7.1. Введите (в десятичной системе счисления) значение 64-разрядного ключа K_1.

1.4.7.2. Введите (в десятичной системе счисления) значение 64-разрядного ключа K_2.

1.4.7.3. Введите (в десятичной системе счисления) значение 64-разрядного блока открытого текста.

1.4.7.4. Руководствуйтесь инструкциями в рабочем окне программы.

1.4.8. Оформите отчет и сделайте выводы.

1.5. СОДЕРЖАНИЕ ОТЧЕТА

1.5.1. Решение задач предварительного задания.

1.5.2. Результаты выполнения работы.

1.5.3. Анализ результатов и выводы.

1.6. КОНТРОЛЬНЫЕ ВОПРОСЫ

1.6.1. Какие существуют режимы работы алгоритма?

1.6.2. К какому типу криптосистем относится алгоритм?

1.6.3. Какой разрядности ключ используется в алгоритме?

1.6.4. Поясните принцип работы блока замены.

1.6.5. Поясните принцип работы функции шифрования.

1.6.6. Перечислите основные достоинства и недостатки алгоритма.

1.6.7. Как повлияет искажение одного бита шифротекста на передаваемую информацию при разных режимах работы алгоритма?

1.6.8. Чем определяется криптостойкость алгоритма?

2. ШИФРОВАНИЕ АЛГОРИТМОМ RSA

2.1. ЦЕЛЬ РАБОТЫ

Закрепление теоретических знаний по алгоритму RSA шифрования информации в криптосистемах асимметричного типа.

2.2. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

2.2.1. Элементы теории чисел

Определения

Число a называется простым, если оно не имеет других натуральных делителей, кроме 1 и a.

Например, 17, 23.

Числа a и b называются взаимно простыми, если наибольший общий множитель этих чисел $(a,b)=1$.

Например: 8 и 9.

Модулярная арифметика

В модулярной арифметике все арифметические действия выполняются, как в обычной арифметике, с учетом того что получаемые числа не могут превышать некоторой величины, называемой модулем.

$(3+14)\bmod 12=5$;

$(3+14)\equiv 5 \bmod 12$.

В общем случае $a\equiv r \bmod n$. Читается a сравнимо с r по модулю n. Это справедливо, если $a=nk+r$, где $k=0,1,2,\ldots$. Отсюда $r=a-nk$ называется вычетом числа a по модулю n, $(0\le r<n)$. Справедливо:

$(a\pm b)\bmod n=(a \bmod n \pm b \bmod n)\bmod n$;

$(ab)\bmod n=(a \bmod n \cdot b \bmod n)\bmod n$;

$a(b\pm c)\bmod n=((ab)\bmod n \pm (ac)\bmod n)\bmod n$.

Использование модулярной арифметики позволяет оперировать с очень большими числами, например, при возведении в степень:

$$a^8 \bmod n=\left(\left(a^2 \bmod n\right)^2 \bmod n\right)^2 \bmod n.$$

Малая теорема Ферма

Если n — простое и $\text{НОД}(a,n)=1$, то

$$a^{n-1} \equiv 1 \bmod n.$$

Функция Эйлера

Количество положительных целых, меньших n, которые взаимно просты с n, определяется с помощью функции Эйлера $\varphi(n)$:

Модуль	n простое	n^2	n^m	pq (p и q простые)
$\varphi(n)$	$n-1$	$n(n-1)$	$n^{m-1}(n-1)$	$(p-1)(q-1)$

Обобщение Эйлера малой теоремы Ферма: если НОД$(a, n)=1$, то

$$a^{\varphi(n)} \equiv 1 \bmod n.$$

Нахождение обратных величин

Если задано уравнение $(ax) \bmod n = 1$, то величина $a^{-1} \equiv x \bmod n$ называется обратной величиной a по модулю n.

Обратная величина существует, если a и n — взаимно простые числа.

Способы нахождения обратных чисел

1. Перебором возможных значений.

Подставляя вместо x числа: $1, 2, \ldots, n-1$ — добиваемся выполнения исходного уравнения.

Пример. $(5x) \bmod 7 = 1$, $x = \left(5^{-1}\right) \bmod 7 = 3$, т.к. $(5 \cdot 3) \bmod 7 = 15 \bmod 7 = (15 - 7 \cdot 2) \bmod 7 = 1$.

2. С помощью функции Эйлера $\varphi(n)$.

$$\left(a^{-1}\right) \bmod n = \left(a^{\varphi(n)-1}\right) \bmod n.$$

Пример.

$$x = \left(5^{-1}\right) \bmod 7 = \left(5^{6-1}\right) \bmod 7 = \left(\left(5^2\right) \bmod 7 \left(5^3\right) \bmod 7\right) \bmod 7 = (4 \cdot 6) \bmod 7 = 3.$$

3. С помощью алгоритма Евклида.

Алгоритм Евклида применяется для нахождения НОД чисел a и b. Однако его расширенный вариант можно использовать и для вычисления обратной величины.

Основной вариант.

Даны a и b, $(a > b)$. Алгоритм имеет итерационный характер:

$a = bq_1 + r_1$, $0 < r_1 < b$;

$b = r_1 q_2 + r_2,\ 0 < r_2 < r_1;$

$r_1 = r_2 q_3 + r_3,\ 0 < r_3 < r_2;$

$\vdots$

$r_{k-2} = r_{k-1} q_k + r_k,\ 0 < r_k < r_{k-1};$

$r_{k-1} = r_k q_{k+1},\ r_{k+1} = 0;$

$(a,b) = r_k,$

где q_i, r_i — частное и остаток на i-м шаге алгоритма. На первом шаге делимое — a, делитель — b, частное — q_1, остаток — r_1. На i-м, $i > 1$ шаге алгоритма: делимое — делитель $i-1$-го шага, делитель — остаток $i-1$-го шага (r_{i-1}), частное — q_i, остаток — r_i.

Пример. Пусть $a = 1071$ и $b = 693$. Найти НОД(a,b).

$$1071 = 693 \cdot 1 + 378, \text{ т.е. } q_1 = 1,\ r_1 = 378;$$

$$693 = 378 \cdot 1 + 315, \text{ т.е. } q_2 = 1,\ r_2 = 315;$$

$$378 = 315 \cdot 1 + 63, \text{ т.е. } q_3 = 1,\ r_3 = 63;$$

$$315 = 63 \cdot 5, \text{ т.е. } q_4 = 5,\ r_4 = 0.$$

То есть на четвертом шаге остаток от деления $r_4 = 0$, следовательно, алгоритм останавливается и НОД$(a,b) = 63$.

Доказано, что при неотрицательных a и b можно найти такие целые числа u_1, u_2, u_3, что будет выполняться

$$a\mathrm{u}_1 + b\mathrm{u}_2 = \mathrm{u}_3 = (a,b).$$

Если выбрать $b = n$ и a, n — взаимно простые числа, т.е. $(a,n) = 1$, тогда

$$a\mathrm{u}_1 + n\mathrm{u}_2 = 1,$$

$$(a\mathrm{u}_1 + n\mathrm{u}_2) \bmod n = (a\mathrm{u}_1) \bmod n = 1,$$

$$\left(a^{-1}\right) \bmod n = \mathrm{u}_1 \bmod n.$$

То есть для нахождения обратной величины необходимо вычислить $\mathrm{u}_1 \bmod n$. Эта задача решается в ходе вычисления НОД(a,n) в соответствии с алгоритмом Евклида. Дополнительно на каждом шаге вычисляются координаты двух векторов:

$$\vec{\mathrm{u}} = (\mathrm{u}_1, \mathrm{u}_2, \mathrm{u}_3),\ \vec{v}(v_1, v_2, v_3).$$

Алгоритм вычисления u_1 имеет следующий вид:

1. Начальные установки:

$\vec{\mathrm{u}}_0 = (0,1,n)$, т.е. $\mathrm{u}_1 = 0$, $\mathrm{u}_2 = 1$, $\mathrm{u}_3 = n$. При этом $a \cdot 0 + b \cdot 1 = n$, т.е. $b = n$,

$\vec{v}_0 = (1,0,a)$, т.е. $v_1 = 1$, $v_2 = 0$, $v_3 = a$. При этом $a \cdot 1 + n \cdot 0 = a$.

2. Проверяем, выполняется ли $\mathrm{u}_3 = 1$, если да, то алгоритм заканчивается.

3. Делим n на a (u_3 на v_3) и определяем:

$q_1 = \left[\frac{u_3}{v_3}\right]$ и значения векторов: $\vec{u}_1 = \vec{v}_0$; $\vec{v}_1 = \vec{u}_0 - q_1\vec{v}_0$.

4. Вернуться к шагу 2.

На каждом шаге при расчетах используются результаты предыдущего:

$q_i = \left[\frac{u_3}{v_3}\right]_{i-1}$, $\vec{u}_i = \vec{v}_{i-1}$, $\vec{v}_i = \vec{u}_{i-1} - q_i\vec{v}_{i-1}$.

При $u_3 = 1$ вычисления заканчиваются $\left(a^{-1}\right) \bmod n = u_1 \bmod n$, где u_1 — значение u_1, полученное на последнем шаге.

Пример. Пусть $n = 23$ и $a = 5$. Найти число x, обратное числу a по модулю n, т.е. найти $5^{-1} \bmod 23$.

Используя расширенный алгоритм Евклида, выполним вычисления.

q	u_1	u_2	u_3	v_1	v_2	v_3
—	0	1	$n = 23$	1	0	$a = 5$
4	1	0	5	–4	1	3
1	–4	1	3	5	–1	2
1	5	–1	2	–9	2	1
—	–9	2	1			

При $u_1 = -9$, $u_2 = 2$, $u_3 = 1$ выполняется уравнение $au_1 + nu_2 = 1$, $a(-9) + n \cdot 2 = 5(-9) + 23 \cdot 2 = 1$ и $a^{-1} \bmod n = 5^{-1} \bmod 23 = (-9) \bmod 23 = 14$.

Итак, $x = 5^{-1} \bmod 23 = (-9) \bmod 23 = 14$.

2.2.2. Алгоритм шифрования RSA

Последовательность действий абонентов криптосистемы RSA

Действия получателя криптограммы B:

1. B генерирует два произвольных больших простых числа P и Q. Эти числа должны быть примерно одинаковыми, размерностью 100-150 десятичных разрядов. Они должны быть секретными.

2. B вычисляет значение модуля $n = PQ$ и функции Эйлера $\varphi(n) = (P-1)(Q-1)$ и выбирает значение открытого ключа K_O с соблюдением условий: $1 < K_O \le \varphi(n)$, $(K_O, \varphi(n)) = 1$, т.е. K_O и $\varphi(n)$ должны быть взаимно простыми.

3. B вычисляет значение секретного ключа K_C, используя расширенный алгоритм Евклида:

$$K_C = \left(K_O^{-1}\right) \bmod \varphi(n).$$

4. B посылает A пару чисел n, K_O по открытому каналу.

Действия отправителя криптограммы A:

1. Разбивает исходный текст M на блоки M_i, $i = 1, 2, \ldots, m$, т.е. $M = M_1, M_2, \ldots, M_m$. Величина $M_i < n$.

2. Шифрует каждое число M_i по формуле $C_i = \left(M_i^{K_O}\right) \bmod n$ и отправляет криптограмму $C = C_1, C_2, \ldots, C_m$.

Получатель B, получив криптограмму, расшифровывает каждый блок секретным ключом K_C, $M_i = \left(C_i^{K_C}\right) \bmod n$, и восстанавливает весь текст $M = M_1, M_2, \ldots, M_m$.

2.2.3. Реализуемость и безопасность RSA

Покажем, что при расшифровывании восстанавливается исходный текст. Согласно обобщению Эйлером малой теоремы Ферма: если $НОД(a, n) = 1$, то $a^{\varphi(n)} \equiv 1 \bmod n$, или $a^{\varphi(n)+1} \equiv a \bmod n$. Открытый K_O и закрытый K_C ключи в алгоритме связаны соотношением $K_O K_C \equiv 1 \bmod \varphi(n)$, или $K_O K_C = k\varphi(n) + 1$ для некоторого целого k. Таким образом, процесс шифрования, а затем расшифровывания некоторого сообщения M_i выглядит следующим образом:

$$\left(\left(M_i^{K_O}\right) \bmod n\right)^{K_C} \bmod n = \left(M_i^{K_O K_C}\right) \bmod n = \left(M_i^{k\varphi(n)+1}\right) \bmod n = M_i.$$

В процессе применения RSA злоумышленник может иметь: C_i, K_O, n — и организовать дешифрование двумя способами:

1. По C_i, K_O, n получить M_i. Для этого он решает задачу вычисления M_i из уравнения $C_i = M_i^{K_O} \bmod n$. Эта задача вычислительно трудна.

2. По n вычислить P, Q, затем найти $\varphi(n)$ и вычислить $K_C = \left(K_O^{-1}\right) \bmod \varphi(n)$ и дешифровать сообщение $M_i = C_i^{K_C} \bmod n$.

Однако задача разложения большого числа на простые множители вычислительно сложна.

Пользователи A и B должны быстро осуществлять все вычисления: вычислять K_O, шифровать и расшифровывать.

Вычисление K_O с использованием алгоритма Евклида — довольно быстрый процесс и не представляет трудности. Зашифровывание и

расшифровывание — возведение большого числа в большую степень — требует определенных затрат времени, но с учетом наличия быстрых алгоритмов и быстродействия современных компьютеров это приемлемая процедура.

2.3. ПРЕДВАРИТЕЛЬНОЕ ЗАДАНИЕ

2.3.1. Изучите теоретическую часть.

2.3.2. Определите число $b = a \bmod n$, где $a = 751$, $n = 39$ и $a = -819$, $n = 52$.

2.3.3. Определите наибольший общий делитель чисел $a = 26569969$ и $b = 14408761$ с помощью алгоритма Евклида.

2.3.4. Определите, являются ли числа $a = 1172827$ и $b = 1089019$ взаимно простыми.

2.3.5. Определите методом перебора число x, обратное числу a (т.е. $(xa) \bmod n = 1$) по модулю n, где $a = 21$, $n = 33$ и $a = 8$, $n = 35$.

2.3.6. Определите с помощью расширенного алгоритма Евклида число x, обратное числу $a = 3$ (т.е. $(xa) \bmod n = 1$) по модулю $n = 667$.

2.3.7. Определите $\varphi(189)$, где $\varphi(n)$ — функция Эйлера.

2.3.8. Определите число $b = a^c \bmod n$, где $a = 83089$, $c = 88711$, $n = 509$.

2.4. ЛАБОРАТОРНОЕ ЗАДАНИЕ

2.4.1. Включите ПЭВМ.

2.4.2. Запустите программу rsa.exe на выполнение. Данная программа реализует алгоритм зашифровывания и расшифровывания данных по алгоритму RSA. Входные и выходные данные работы этого алгоритма представляют собой числа, не превосходящие значение модуля.

2.4.3. Нажмите кнопку «Начать выполнение работы». Выполнение работы состоит в том, что Вы должны вручную найти значения в контрольных точках работы алгоритма. Для этого необходимо ввести полученное значение в соответствующее поле рабочего окна программы и нажать кнопку «Проверить». Если расчет выполнен правильно, происходит переход к следующей контрольной точке.

2.4.4. Руководствуйтесь инструкциями в рабочем окне программы.

2.5. СОДЕРЖАНИЕ ОТЧЕТА

2.5.1. Решение задач предварительного задания.

2.5.2. Результаты выполнения работы.

2.5.3. Анализ результатов и выводы.

2.6. КОНТРОЛЬНЫЕ ВОПРОСЫ

2.6.1. Какие числа называются простыми, взаимно простыми?

2.6.2. Какое число называется обратным к числу a по модулю n?

2.6.3. Поясните сущность функции Эйлера?

2.6.4. Что такое вычет числа a по модулю n?

2.6.5. Какие основные свойства арифметических действий в модулярной арифметике.

2.6.6. Поясните основные способы нахождения обратных величин в модулярной арифметике.

2.6.7. Поясните работу алгоритма Евклида?

2.6.8. К какому типу криптосистем относится алгоритм RSA?

2.6.9. Приведите достоинства криптосистем с открытым ключом.

2.6.10. Раскройте связь открытого K_O и секретного K_C ключей алгоритма RSA?

2.6.11. Перечислите действия отправителя криптограммы.

2.6.12. Перечислите действия получателя криптограммы.

2.6.13. Каким образом злоумышленник может расшифровать криптограмму?

ЛИТЕРАТУРА

Романец Ю.В., Тимофеев П.А., Шаньгин В.Ф. Защита информации в компьютерных системах и сетях. М.: Радио и связь, 1999.

Мельников В.В. Защита информации в компьютерных системах. М.: Финансы и статистика, 1997.

Харин Ю.С., Берник В.И., Матвеев Г.В. Математические основы криптологии. Мн.: БГУ, 1999.

Герасименко В.А., Малюк А.А. Основы защиты информации. М.: МГИФИ, 1997.

Леонов А.П., Леонов К.П., Фролов Г.В. Безопасность автоматизированных банковских и офисных технологий. Мн.: Нац. кн. палата Беларуси, 1996.

Зима В.М., Молдовян А.А., Молдовян Н.А. Компьютерные сети и защита передаваемой информации. СПб.: СПбГУ, 1998.

ПРИЛОЖЕНИЕ

Таблица П1

Начальная перестановка IP							
58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

Таблица П2

Обратная перестановка IP^{-1}							
40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

Таблица П3

Функция E					
32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

Таблица П4

Функция P			
16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25

Таблица П5

Функция G							
C_0	57	49	41	33	25	17	9
	1	58	50	42	34	26	18
	10	2	59	51	43	35	27
	19	11	3	60	52	44	36
D_0	63	55	47	39	31	23	15
	7	62	54	46	38	30	22
	14	6	61	53	45	37	29
	21	13	5	28	20	12	4

Таблица П6

Функция H					
14	17	11	24	1	5
3	28	15	6	21	10
23	19	22	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

Таблица П7

Таблица сдвигов для вычисления ключа															
Итерация	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Сдвиг влево	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Таблица П8

Функции S																
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
S_1	**0**	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
	1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
	2	4	1	4	8	13	6	2	11	15	12	9	7	3	10	5	0
	3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13
S_2	**0**	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
	1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
	2	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
	3	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
S_3	**0**	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
	1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
	2	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
	3	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
S_4	**0**	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
	1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
	2	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
	3	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14
S_5	**0**	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
	1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
	2	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
	3	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3
S_6	**0**	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
	1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
	2	9	14	15	5	2	8	12	3	7	0	4	10	1	13	1	6
	3	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13
S_7	**0**	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
	1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
	2	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
	3	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12
S_8	**0**	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
	1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
	2	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
	3	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Св. план 2003, поз. 91

Учебное издание

КРИПТОГРАФИЧЕСКОЕ КОДИРОВАНИЕ ИНФОРМАЦИИ

Методические указания
к лабораторной работе
по дисциплинам «Основы защиты информации»
и «Криптографическая защита информации в телекоммуникациях»
для студентов специальности «Сети телекоммуникаций»
дневной, вечерней и заочной форм обучения

В 3-х частях

Часть 2

ШИФРОВАНИЕ АЛГОРИТМАМИ DES и RSA

Составители:
Голиков Владимир Федорович,
Курилович Андрей Владимирович

Редактор Т.А. Лейко
Корректор Е.Н. Батурчик

Подписано в печать Формат 64×80 1/16.
Бумага офсетная. Печать ризографическая. Гарнитура «Таймс». Усл. печ. л.
Уч.-изд. л. 1,3. Тираж 100 экз. Заказ 606.

Издатель и полиграфическое исполнение:
Учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники»
Лицензия ЛП № 156 от 05.02. 2001.
Лицензия ЛВ № 509 от 03.08. 2001.
220013, Минск, П.Бровки, 6.