

**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ
ФГБОУ ВПО «МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ
ОТКРЫТЫЙ УНИВЕРСИТЕТ им. В.С.Черномырдина»
Институт (филиал) в г. Махачкале**

КАФЕДРА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

для выполнения лабораторных работ по дисциплине «Методы и средства защиты
компьютерной информации »

Махачкала 2013

Методические указания для выполнения лабораторных работ по дисциплине «Методы и средства защиты компьютерной информации» – Махачкала: МГОУ, 2013г.– с.

Методические указания предназначены для студентов специальности **230105.65 ПОАСиВТ**, а также для студентов других специальностей, изучающих дисциплину «Методы и средства защиты компьютерной информации».

Методические указания содержат описания лабораторных работ 1, 2, 3, 4 и 5. К каждой лабораторной работе прилагаются краткие теоретические сведения , индивидуальные задания и контрольные вопросы.

Составитель: доцент кафедры ИТ Саидахмедова М.Б.

Рецензенты:

Печатается согласно постановлению

от «_____» 2013г.

Лабораторная работа №1
Изучение простейшего сом-вируса
1. цель работы

Изучение принципов функционирования простейшего вируса

СТРУКТУРА И ПРОЦЕСС ЗАГРУЗКИ СОМ - ПРОГРАММЫ

Прежде чем рассматривать СОМ-вирусы приведем основные сведения по структуре и процессу загрузки СОМ-программ. Что же представляет собой СОМ-программа, как она загружается в память и запускается? Структура СОМ-программы предельно проста - она содержит только код и данные программы, не имея даже заголовка. Размер СОМ-программы ограничен размером одного сегмента (64Кбайт).

И еще два понятия, которые часто будут встречаться:

Program Segment Prefix (PSP) - область памяти размером 256 (0100h) байт, предшествующая программе при ее загрузке. PSP содержит данные командной строки и относящиеся к программе переменные.

Disk Transfer Address (DTA) - блок данных, содержащий адреса обмена данными с файлом (чтение или запись). Область DTA для работы с файлом используют многие функции, в том числе и не производящие чтение или запись в файл. Примером может служить функция 4Eh (найти первый файл по шаблону), которая будет неоднократно встречаться в листингах программ.

Загрузка СОМ-программы в память и ее запуск происходят так:

1. Определяется сегментный адрес свободного участка памяти достаточного для размещения программы размера.
2. Создается и заполняется блок памяти для переменных среды.
3. Создается блок памяти для PSP и программы (сегмент: 0000h - PSP; сегмент: 0100h - программа). В поля PSP заносятся соответствующие значения.
4. Устанавливается адрес DTA равным PSP:0080h.
5. Загружается СОМ-файл с адреса PSP:0100h.
6. Значение регистра AX устанавливается в соответствии с параметрами командной строки.

7. Регистры DS, ES и SS устанавливаются на сегмент PSP и программы (PSP:0000h).

8. Регистр SP устанавливается на конец сегмента, после чего в стек записывается 0000h.

9. Происходит запуск программы с адреса PSP:0100h.

COM-программа всегда состоит из одного сегмента и запускается со смещения 0100h.

ПРОСТЕЙШИЙ COM - ВИРУС

В начале COM-файла обычно находится команда безусловного перехода JMP, состоящая из трех байт. Первый байт содержит код команды 0E9h, следующие два

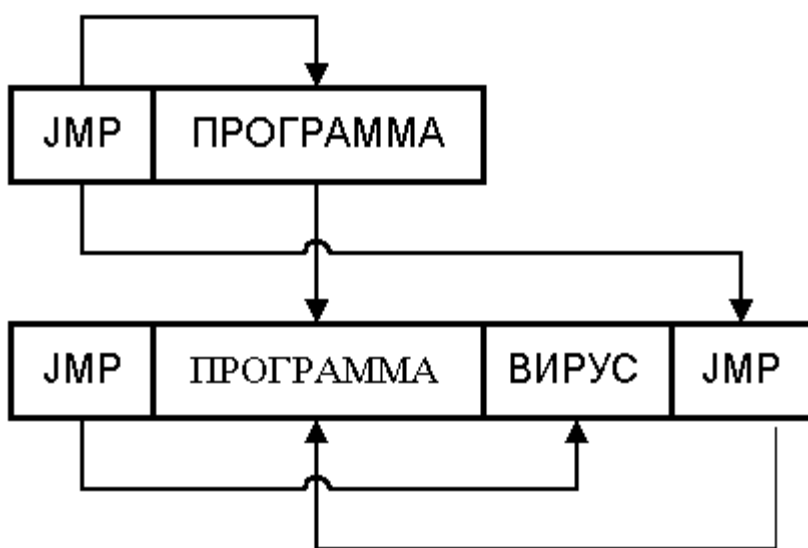


Рис. 1

— адрес перехода. Поскольку рассматриваемый ниже вирус учебный, он будет заражать только COM-файлы, начинающиеся с команды JMP. Благодаря простому строению COM-файла в него очень просто добавить тело вируса и затем указать его адрес в команде JMP. На рис.1. показано заражение файла таким способом.

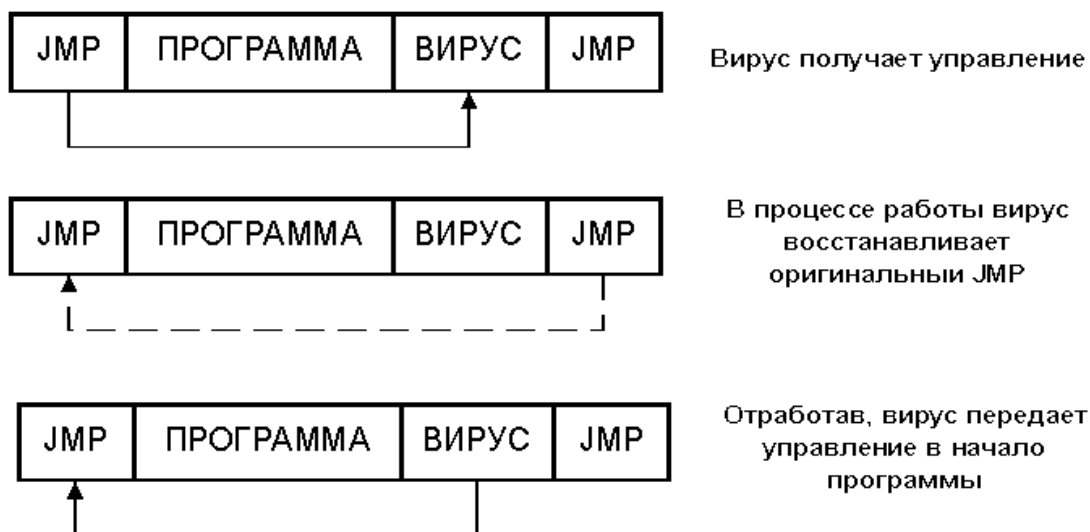


Рис 2

После загрузки зараженного файла управление получает вирус. Закончив работу, вирус восстанавливает оригинальный JMP и передает управление программе, как показано на рис. 2.

Что же делает рассматриваемый вирус? После старта он ищет в текущем каталоге СОМ-программы. Для этого используется функция 4Eh (найти первый файл).

СПОСОБЫ ВНЕДРЕНИЯ СОМ - ВИРУСОВ

Рассмотренный вирус дописывался в конец файла, а в начало файла вписывал переход на себя. Существуют и другие способы внедрения вирусов.

Рассмотрим два варианта внедрения СОМ-вируса в начало файла. Вариант первый. Вирус переписывает начало программы в конец файла, чтобы освободить место для себя. После этого тело вируса записывается в начало файла, а небольшая его часть,

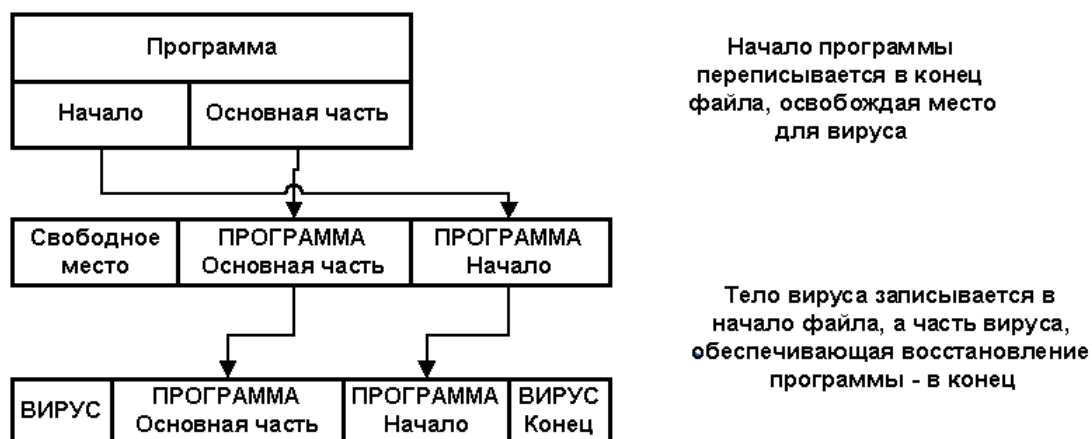


Рис. 3.

обеспечивающая перенос вытесненного фрагмента программы, на прежнее место — в конец. При восстановлении первоначального вида программы тело вируса будет затерто, поэтому код вируса, восстанавливающий программу, должен находиться в безопасном месте, отдельно от основного тела вируса. Этот способ внедрения изображен на рис. 3.

При загрузке зараженного таким способом файла управление получит вирус (так как он находится в начале файла и будет загружен с адреса 0100h). После окончания работы вирус передает управление коду, переносящему вытесненную часть программы на прежнее место. После восстановления (в памяти, не в файле) первоначального вида программы, она запускается. Схема работы вируса изображена на рис. 4.

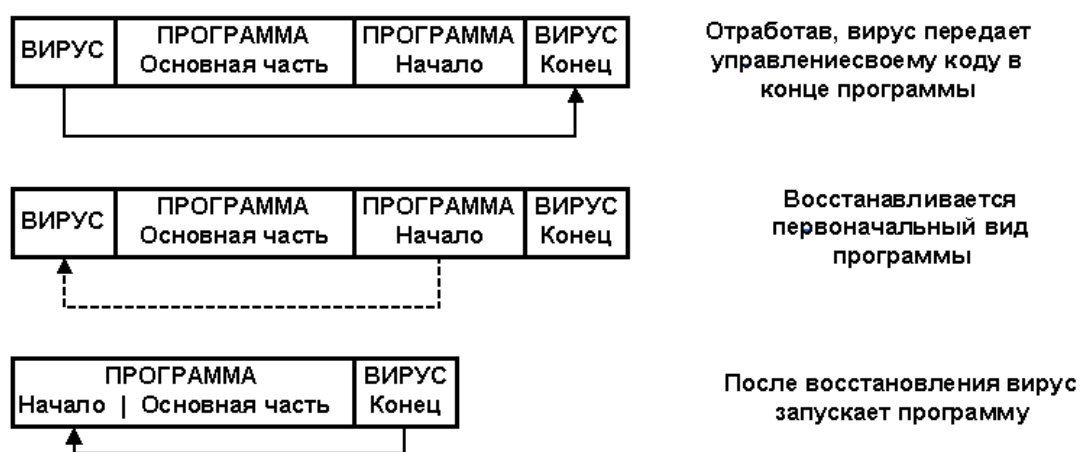


Рис. 4.

Второй вариант отличается от первого тем, что вирус, освобождая для себя место, сдвигает все тело программы, а не переносит ее часть в конец файла. Этот способ внедрения изображен на рис. 5.

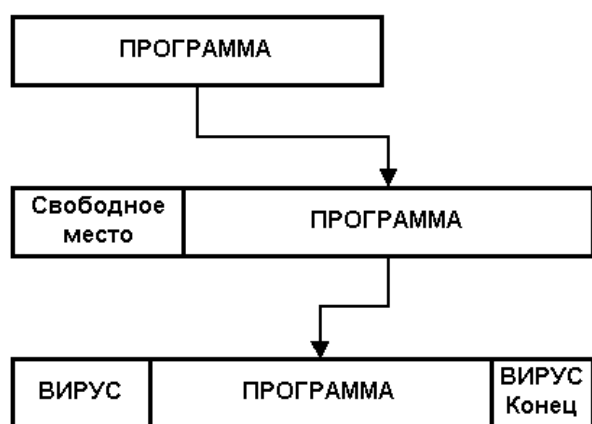


Рис. 5.

Тело программы сдвигается ближе к концу файла, освобождая место для вируса

Тело вируса записывается в начало файла, а часть вируса, обеспечивающая восстановление программы - в конец

После восстановления вирус запускает программу

После запуска зараженной программы, как и в предыдущем случае, управление получает вирус. Дальнейшая работа вируса отличается только тем, что часть вируса, восстанавливающая первоначальный вид программы, переносит к адресу 0100h все тело программы, а не только вытесненную часть. Схема работы вируса, заражающего файл таким образом, приведена на рис. 6.

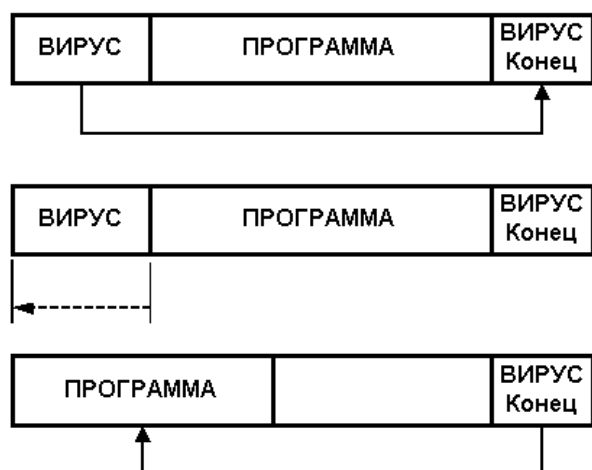


Рис. 6.

Отработав, вирус передает управление своему коду в конце программы

Восстанавливается первоначальный вид программы - тело программы сдвигается к адресу 0100h

После восстановления вирус запускает программу

Существуют разновидности вирусов, не дописывающие часть своего тела в конец файла. К примеру, вирус может внедряться в середину файла. В этом случае алгоритм работы вируса является смесью алгоритмов одного из двух только что описанных вирусов и вируса, описанного в разделе “Простейший COM-вирус”.

Алгоритм работы COM вируса

1. Ищем первый файл по шаблону.

2. Открываем файл.
3. Читаем первый байт файла.
4. Сравниваем если первый байт не E9h , то переходим к поиску следующего файла.
5. Получаем длину файла.
6. Если файл больше 64000 байт ,ищем другой файл.
7. Проверяем не заражен ли файл , если заражен ищем другой файл.
8. Копируем вирус в файл.
9. Записываем в начало файла переход на тело вируса.
10. Передаем управление программе носителю.

3. ЛАБОРАТОРНОЕ ЗАДАНИЕ.

1. Изучить работу вируса, заражающего СОМ файлы
2. Проверить работу вируса на некотором файле с расширением СОМ
3. Составить программу защиты от вируса - ревизора изменений.

4. СОДЕРЖАНИЕ ОТЧЕТА.

1. Алгоритм программы
2. Блок-схема программы
3. ИСКСТ программы
4. Выводы об эффективности защиты.

4. ЛИТЕРАТУРА.

1. Хофман А. Д Современные методы защиты информации - М. : Сов. Радио, 1980 г
2. Гульев И. Компьютерные вирусы - М. : ДМК, 1999 Г
3. Таили Э. Безопасность компьютера - Минск: Попурри, 1997 г

ЛАБОРАТОРНАЯ РАБОТА №2

Методы шифрования

Наиболее известными и часто используемыми шифрами являются шифры замены. Они характеризуются тем, что отдельные части сообщения (буквы, слова) заменяются на какие-либо другие буквы, числа, символы. При этом замена осуществляется так, чтобы потом по шифрованному сообщению можно было однозначно восстановить исходное.

Таблица.

а	б	в	...	я
Ма	Мб	Мв	...	Мя

является ключом шифра замены. Зная ее можно осуществить как зашифрование, так и расшифрование.

Шифр Цезаря.

Пример.

Вторая строка является последовательностью букв первой строки.

а	б	в	...	я
г	д	е	...	в

Число сдвига равно трем. ВАГОН ->

ЕГЖСР

Во время 2-ой Мировой войны часто использовались так называемые книжные шифры.

Множество шифрообозначений для каждой буквы определяется всеми пятизначными наборами цифр, в каждом из которых первые две цифры указывают номер страницы, третья цифра номер строки, четвертая и пятая номер места данной буквы в указанной строке.

При поимке разведчика всегда пытались найти книгу, которая могла быть использована им в качестве ключа.

Шифр, преобразования их которого изменяют только порядок следования символов исходного текста, но не изменяют их самих, называют шифром перестановки.

Всего перестановок n -мерной последовательности может быть $n!$.

Рассмотрим пример шифра вертикальной перестановки.

В нем используется прямоугольник, в который сообщения вписываются обычным способом по строкам слева

направо. Выписываются буквы по вертикали, а столбцы при этом берутся в порядке, определяемом ключом.

Пусть, например, ключ текста: (5, 1, 4, 7, 2, 6, 3), и с его помощью надо зашифровать сообщение:

ВОТ ПРИМЕР ВЕРТИКАЛЬНОЙ ПЕРЕСТАНОВКИ

Впишем сообщение в прямоугольник, столбцы которого пронумерованы в соответствии с ключом:

5	1	4	7	2	6	3
В	О	Т	П	Р	и	м
Е	Р	В	Е	Р	т	и
К	А	л	Ь	Н	О	и
П	Е	Р	Е	С	т	А
Н	О	в	К	и	-	-

Теперь выбирая столбцы в порядке, заданном ключом и выписывая последовательно буквы каждого из них сверху вниз, получаем такую криптограмму:

ОРАЕО-РРНСИ-МИЙА-ТВЛРВ-ВЕКПН-ИТОТ-ПЕЬЕК

3. Лабораторное задание.

1. Составить программу кодирования, декодирования по шифру Цезаря.
2. Составить программу кодирования, декодирования по шифру вертикальной перестановки.

4. Содержание отчета

1. Алгоритм программы.
2. Блок-схема программы.
3. Текст программы.
4. Оценка эффективности шифров

Лабораторная работа №3

Защита данных при помощи ключа

1.Цель работы

Изучение схемы аутентификации Шнорра и шифрования при помощи ключа

2.Краткие теоретические сведения

Большинство программных защит используют шифровку своего кода в целях затруднения анализа и модификации кода.

Наиболее популярными являются криптосистемы на основе логической операции хог. Одним из её свойств является зеркальность. Повторное шифрование результата восстанавливает исходный текст. Шифровщик и дешифровщик устроены одинаково, что упрощает и сокращает код. Докажем, что $a \text{ хог } b \text{ хог } = b$.

Для этого перечислим всевозможные значения a и b в следующей таблице:

$a \backslash b$	0	1
0	0 хог исключаяющее или $0 \ 0 == 0$	$0 \text{ хог } 1 \text{ хог } 0 == 1$
1	1 хог исключаяющее или $0 \ 1 == 0$	$1 \text{ хог } 1 \text{ хог } 1 == 1$

Заметим, что хог – битовая операция. Аргументы a и b могут иметь только 2 значения: 0 и 1. Однако, эту же операцию можно проводить для последовательности битов.

Из таблицы видно, что $a \text{ хог } 0 == a$; $a \text{ хог } 1 == a$. Т.е. значащими в маске шифрования являются только единичные биты. Поэтому рекомендуется выбирать только такую маску, в которой единичные и нулевые биты равномерно перемешаны, например 01010011.

В протоколе аутентификации имеется 2 участника пользователи А и В. Пользователь А должен доказать свою аутентичность. В - должен её проверить. У А имеется два ключа – общедоступный, открытый К1 и секретный К2. А нужно доказать, что он знает К2, и сделать это таким образом, чтобы это доказательство можно было проверить, зная только К1.

Схема аутентификации Шнорра.

1. Пользователь А выбирает случайное число k из множества $\{1, \dots, q-1\}$, вычисляет $r = g^k \bmod p$ и посылает В.
2. В выбирает случайный запрос e из множества $\{0, \dots, 2^t - 1\}$, где t – некоторый параметр, и посылает e пользователю А.
3. А вычисляет $s = k + xe \bmod q$ и посылает s В.
4. В проверяет соотношение $r = g^s y^e \bmod p$ и, если оно выполняется, принимает доказательство, в противном случае – отвергает.

Здесь p и q – простые числа такие, что q делит $p-1$. g принадлежит множеству целых чисел до p таково, что $g^q = 1 \bmod p$, $q < p$.

3. Лабораторное задание

1. Составить программу шифрования и дешифрования на основе логической операции хог.
2. Составить программу аутентификации по методу Шнорра.

4. Содержание отчёта

1. Алгоритм программ.
2. Блок-схема программ.
3. Тексты программ.

5. Литература

1. Хофман А.Д. Современные методы защиты информации – Б.:Сов.Радио, 1980 г.
2. Яценко В.В. Введение в криптографию – М.: МЦНМО, 1999 г.

Лабораторная работа №4

Алгоритм шифрования RSA

Алгоритм RSA -- асимметричный, т. е. для шифрования и расшифрования используются разные ключи. Любой из ключей может быть открытым, т. е. кто угодно сможет зашифровать текст, но расшифровать -- только тот, кто знает секретный ключ (хорошие системы парольной идентификации -- пароли хранятся в зашифрованном виде, ввод шифруется и сравнивается с паролем -- т. е. даже полностью разобравшись в логике шифратора нельзя узнать исходный пароль, т. к. ключ для расшифрования будет физически отсутствовать) или, наоборот, прочесть текст может каждый, но зашифровать -- только один человек (системы подтверждения подлинности; программы, зашифрованный код которых нельзя изменить, даже зная алгоритм шифрования).

Вот краткое изложение алгоритма:

1. Выбираются два простых числа -- P и Q ;
2. $N := P * Q$;
3. $M := (P-1) * (Q-1)$;
4. Выбирается число D , взаимно простое с M , т. е. $\text{НОД}(D, M) = 1$;
5. Выбирается число E , так чтобы $(E * D) = 1 \pmod{M}$;
6. Теперь пара (E, N) -- открытый ключ, (D, N) -- закрытый.

Шифрование происходит так:

1. Данные разбиваются на блоки (C) , каждый из которых может быть представлен числом от 0 до $N-1$;
2. Для шифрования полагаем $C := (B^E) \pmod{N}$;
3. Для дешифрования полагаем $B := (C^D) \pmod{N}$.

Криптостойкость алгоритма основана на предположении, что не существует эффективного способа разложения числа на простые множители. С другой стороны, пока нельзя гарантировать, что таких способов не появится в будущем.

Так как криптостойкость основана сложности разложения на множители, рекомендуется использовать в качестве ключей достаточно большие числа -- 256--512 двоичных разрядов (для достижения надежности DES необходимо применять около 700 двоичных разрядов).

Для достижения хорошей секретности, кроме того, необходимо, чтобы числа $(P-1)$ и $(Q-1)$ имели большие простые делители (т. е. число 2^{X+1} является "плохим"). Кроме того, при регулярном использовании алгоритма, лучше, если у разных ключей будут отличаться оба числа в паре (т. е. нежелательно использовать ключи $(E1, N)$ и $(E2, N)$ или $(E, N1)$ и $(E, N2)$).

На сегодняшний день RSA является самым распространенным криптографическим алгоритмом с открытым ключом.

Алгоритм шифрования RSA

Алгоритм RSA предполагает, что посланное закодированное сообщение может быть прочитано адресатом и только им. В этом алгоритме используется два ключа - открытый и секретный. Данный алгоритм привлекателен также в случае, когда большое число субъектов (N) должно общаться по схеме все-со-всеми. В случае симметричной схемы шифрования каждый из субъектов каким-то образом должен доставить свои ключи всем остальным участникам обмена, при этом суммарное число используемых ключей будет достаточно велико при большом значении N . Применение асимметричного алгоритма требует лишь рассылки открытых ключей всеми участниками, суммарное число ключей равно N .

Сообщение представляется в виде числа M . Шифрование осуществляется с помощью общедоступной функции $f(M)$, и только адресату известно, как выполнить операцию f^{-1} . Адресат выбирает два больших простых (prime) числа p и q , которые делает секретными. Он объявляет $n=pq$ и число d , с $(d, p-1)=(d, q-1)=1$ (один из

возможных способов выполнить это условие, выбрать d больше чем $p/2$ и $q/2$). Шифрование производится по формуле:

$$f(M) \equiv M^d \bmod n,$$

где M и $f(M)$ оба $\leq n-1$. Как было показано, может быть вычислено за разумное время, даже если M , d и n содержит весьма большое число знаков. Адресат вычисляет M на основе M^d , используя свое знание p и q . В соответствии со следствием 6, если

$$dc \equiv_{(p-1)} 1, \text{ тогда } (M^d)^e \equiv_p 1.$$

Исходный текст M получается адресатом из зашифрованного $F(M)$ путем преобразования: $M = (F(M))^e \pmod{pq}$. Здесь как исходный текст, так и зашифрованный рассматриваются как длинные двоичные числа.

Аналогично $(M^d)^e \equiv_q M$, если $dc \equiv_{(q-1)} 1$. e удовлетворяет этим двум условиям, если $cd \equiv_{(p-1)(q-1)} 1$. Теорема 1 гласит, что мы можем позволить $e=x$, когда x является решением уравнения $dx + (p-1)(q-1)y = 1$.

Так как $(M^d)^e - M$ делимо на p и q , оно делимо и на pq , следовательно, мы можем определить M , зная M^d , вычислив его значение в степени e и определив остаток от деления на pq . Для соблюдения секретности важно, чтобы, зная n , было нельзя вычислить p и q . Если n содержит 100 цифр, подбор шифра связан с перебором $\sim 10^{50}$ комбинаций. Данная проблема изучается уже около 100 лет. RSA-алгоритм запатентован (20 сентября 1983, действует до 2000 года).

Теоретически можно предположить, что возможно выполнение операции f^{-1} , не вычисляя p и q . Но в любом случае задача эта не проста и разработчики считают ее трудно факторизуемой.

Предположим, что мы имеем зашифрованный текст $f(M)$ и исходный текст M , и мы хотим найти значения p и q . Нетрудно показать, что таких исходных данных для решения задачи недостаточно - надо знать все возможные значения M_i .

Проясним использование алгоритма RSA на конкретном примере. Выбираем два простые числа $p=7$; $q=17$ (на практике эти числа во много раз длиннее). В этом случае $n = p \cdot q$ будет равно 119. Теперь необходимо выбрать e , выбираем $e=5$. Следующий шаг связан с формированием числа d так, чтобы $d \cdot e = 1 \bmod [(p-1)(q-1)]$.

$d=77$ (использован расширенный алгоритм Эвклида). d - секретный ключ, а e и n характеризуют открытый ключ. Пусть текст, который нам нужно зашифровать представляется $M=19$. $C = M^e \bmod n$. Получаем зашифрованный текст $C=66$. Этот “текст” может быть послан соответствующему адресату. Получатель дешифрует полученное сообщение, используя $M = C^d \bmod n$ и $C=66$. В результате получается $M=19$.

На практике общедоступные ключи могут помещаться в специальную базу данных. При необходимости послать партнеру зашифрованное сообщение можно сделать сначала запрос его открытого ключа. Получив его, можно запустить программу шифрации, а результат ее работы послать адресату. На использовании общедоступных ключей базируется и так называемая электронная подпись, которая позволяет однозначно идентифицировать отправителя. Сходные средства могут применяться для предотвращения внесения каких-либо корректив в сообщение на пути от отправителя к получателю. Быстродействующие аппаратные 512-битовые модули могут обеспечить скорость шифрования на уровне 64 кбит в сек. Готовятся ИС, способные выполнять такие операции со скоростью 1 Мбайт/сек. Разумный выбор параметра e позволяет заметно ускорить реализацию алгоритма.

Лабораторная работа №5

«Защита данных при помощи алгоритма DES»

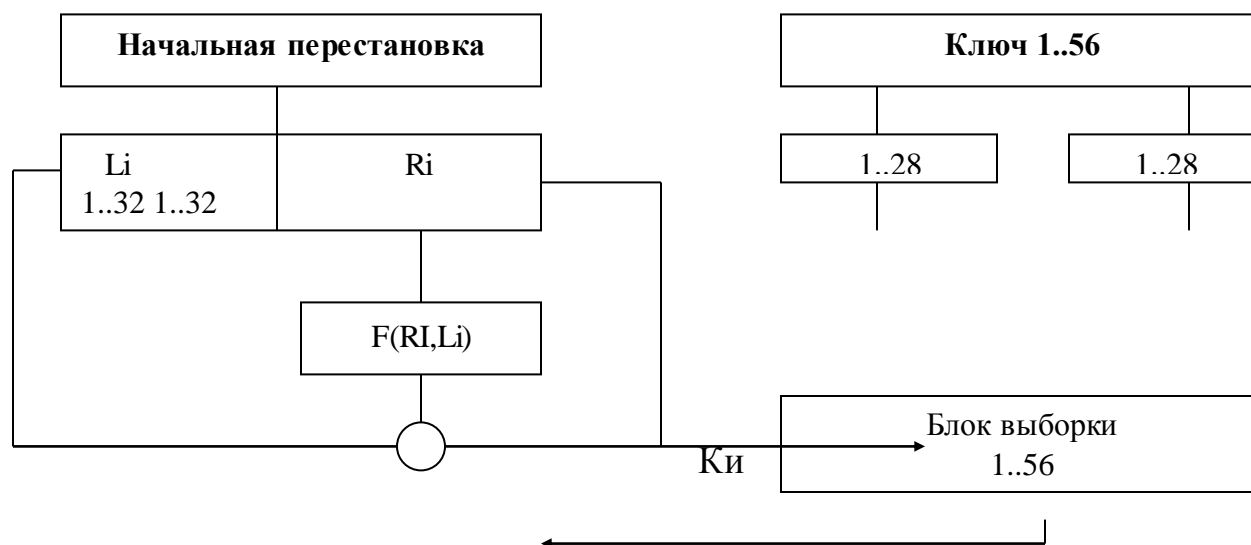
1. Цель работы

Изучение алгоритма DES

2. Краткие теоретические сведения

Одним из самых распространенных алгоритмов блочного шифрования рекомендованных Национальным бюро стандартов США совместно с АНБ в качестве основного средства криптографической защиты информации как в государственных так и в коммерческих структурах, является Data Encryption Standard (DES).

DES является блочным алгоритмом шифрования с длиной блока 64 бита и симметричными ключами и длиной 56 бит. На практике обычно ключ имеет длину 64 бита, где каждый восьмой бит используется для контроля четности остальных битов ключа.



В некоторых реализациях DES блоки открытого сообщения перед тем, как они будут загружены в регистр сдвига длиной две ячейки и размером ячейки 32 бита, проходят процедуру начальной перестановки, которая применяется для того, чтобы

осуществить начальное рассеивание статической структуры сообщения. Пример начальной перестановки приведен в табл. 1.1

Таблица 1.2. Начальная перестановка

58	50	42	34	26	18	10	2	60	52	44	36	28	20	12	4
62	54	46	38	36	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

В случае использования начальной перестановки после завершения 16 раундов к полученному блоку применяется обратная перестановка. Работа алгоритма заключается в следующем:

1. Входной блок разбивается на две части по 32 бита в каждой (L-левая половина, R- правая половина).
2. Правая половина преобразуется функцией f с использованием текущей ключевой последовательности длиной 48бит, снятой с выхода блока выработки ключевой последовательности.
3. Результат преобразования правой части складывается по модулю 2 с левой частью, а результат сложения записывается в исходный регистр, при этом исходная правая часть при помощи операции сдвига записывается на место исходной левой части.

Таким образом в регистре оказывается следующая последовательность:

$$L_i = R_{i-1}$$

$$R_i = L_{i-1} + f(r_{i-1}, k_i)$$

Данная процедура повторяется 16 раз, только в последнем цикле замены местами правой и левой части не происходит. По завершении последнего цикла полученная последовательность проходит процедуру завершающей перестановки, которая задается подстановкой, таб 1.3 являющейся обратной к начальной подстановке и учитывающей, что в последнем цикле половины блока не меняются местами.

Таблица 1.3. Завершающая перестановка

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

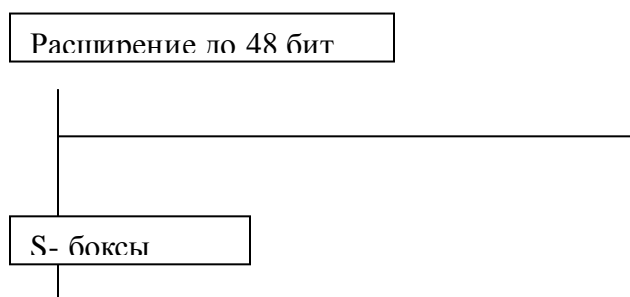
Преобразование начинается с операции расширения исходной 32-битной последовательности до 48 бит Эта операция предполагает дописывание в исходную последовательность отдельного бита в соответствии с подстановкой (см. табл 1.4)

Таблица 1.4

Подстановка расширения

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	28	29	29	30	31	32	1

Результат преобразования суммируется по модулю 2 48-битной ключевой последовательностью. Структура функции F которая вырабатывается из 56-битного ключа, записанного два 28-битных циклических регистра-сдвига, которые перемещают содержимое в каждом такте на количество битов, зависящие от номера раунда (табл. 1.5)



Подстановка

Таблица 1.5

Таблица зависимости количества сдвигаемых битов от номера раунда

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Результирующая ключевая последовательность получается путем выборки 48 бит из содержимого регистров в соответствии с подстановкой (табл. 1.6).

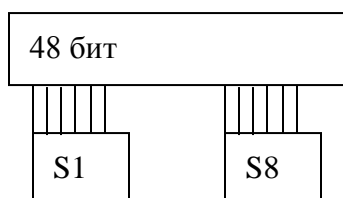
Таблица 1.6

Подстановка для выбора ключа

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
63	55	47	39	3	23	15	7	62	54	46	38	30	22
14	6	61	53	45	37	29	21	13	5	28	20	12	4

Полученный путем сложения 48 битный вектор поступает на вход S-блоков, основная задача которых заключается в сдвиге 48-битного вектора на 32 битный. Всего в DES используются 8 S-блоков 6 битными входами и 4 – битными выходами.

Подстановка в S-блоках осуществляется в соответствии с таблицей 1.7: здесь номер строки задается первым и последним входом S-блока, а номер столбца – средними 4 битами. Битовое представление числа в ячейке задано входной последовательностью и будет являться выходом S-блока.



32бит

Структура S- боксов

Таблица 1.7 Подстановки в S-блоках

S-блок1

14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	13	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

S-блок2

15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
13	8	10	1	3	15	4	2	11	6	7	2	0	5	14	9

S-блок3

10	0	9	14	6	3	15	5	1	13	12	7	1	4	2	8
13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1	10	13	0	6	9	8	7	4	15	14	3	1	5	2	12

S-блок4

7	13	14	3	0	6	9	10	1	2	8	5	1	12	4	15
13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

Таблица 1.7. Подстановки в 5-блоках (окончание)

5-блок5

2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9	14	11	2	12	4	7
13	1	5	0	15	10	3	6	8	6												
4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14	11	8	12	7	1	14
2	13	6	15	0	9	10	4	5	3												

S-блок 6

12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11	10	15	4	2	7	
12	9	5	6	1	13	14	0	11	3	8	9	14	15	5	2	8	12	3	7	0	4
10	1	13	11	64	3	2	12	9	5	15	10	11	14	1	7	6	0813				

Аналитическая сложность дешифрования DES зависит от математических свойств S-блоков, поскольку именно в них реализуются нелинейные преобразования. Все остальные операции в этом алгоритме носят линейный характер, и аналитическое вычисление подобной зависимости не представляет труда для криптоаналитика противника.