

А.Н. Молчанов

**КРИПТОГРАФИЧЕСКИЕ МЕТОДЫ
ЗАЩИТЫ ИНФОРМАЦИИ**

*Методические указания по выполнению лабораторных работ
по дисциплине «Криптографические методы защиты информации»*

Калуга

УДК 681.3-7
ББК 32.973

Данные методические указания разработаны в соответствии с учебным планом специальности 090303.65 «Информационная безопасность автоматизированных систем».

Указания рассмотрены и одобрены:

кафедрой ЭИУ6-КФ «Информационная безопасность автоматизированных систем»

протокол № 1 от 03 сентября 2015 г.

Зав. кафедрой ЭИУ6-КФ [подпись] к.т.н., доц. Мазин А.В.

методической комиссией факультета ЭИУК

протокол № _____ от _____ 20__ г.

Председатель методической комиссии

факультета ЭИУК

[подпись] к.т.н., доц. Адкин М.Ю.

методической комиссией Калужского филиала МГТУ им. Н.Э. Баумана

протокол № 2 от 03. 11 2015 г.

Председатель методической комиссии

КФ МГТУ им. Н.Э. Баумана

Зам.директора по учебной

работе [подпись] Перерва О.Л.

Рецензент:

к.т.н., доцент кафедры ЭИУ2-КФ

[подпись] Донецков А. М.

Автор

[подпись] ст. преподаватель кафедры ЭИУ6-КФ

Молчанов Алексей Николаевич

В методических указаниях изложены основные методы защиты информации с помощью криптографических алгоритмов, способы их реализации, а так же программные продукты, использующие самые современные криптографические алгоритмы. Предназначено для студентов специальности 090303.65 «Информационная безопасность автоматизированных систем» и может быть рекомендовано к применению при проведении лабораторных работ по курсу «Криптографические методы защиты информации».

©Калужский филиал МГТУ им. Н.Э. Баумана, 2015 г.

© Кафедра ЭИУ6-КФ, 2015 г.

© Молчанов А.Н., 2015 г.

ВВЕДЕНИЕ

По мере развития и усложнения средств, методов и форм автоматизации процессов обработки информации повышается зависимость общества от степени безопасности используемых им информационных технологий, которая определяется степенью защищенности и устойчивости как компьютерных систем в целом, так и отдельных программ.

Для обеспечения защиты информации в настоящее время не существует какого-то одного технического приема или средства, однако общим в решении многих проблем безопасности является использование криптографии и криптоподобных преобразований информации. Большинство средств защиты информации базируется на использовании криптографических шифров и процедур шифрования - расшифрования.

При выполнении предлагаемых заданий студент должен ознакомиться с основными методами криптографической защиты и получить практические навыки криптографических преобразований информации.

Цель лабораторных работ:

выработать практические навыки применения методов симметричного и асимметричного шифрования, анализа систем шифрования, а так же выбора и эксплуатации современных средств криптографической защиты информации

После выполнения лабораторных работ студенты смогут:

- составлять алгоритмы для решения прямых и обратных задач шифрования;
- ориентироваться в современных и перспективных методах криптографической защиты информации;
- эффективно использовать криптографические методы и средства защиты информации в автоматизированных системах;

Требования к оборудованию:

- Компьютер уровня не ниже, чем Pentium CoreDuo/RAM 1024 MB/HDD 1 GB;
- Операционная система Windows версии XP и выше с типовым вариантом установки;
- любая современная среда программирования (например: Visual Studio 8 или выше, RAD Studio 2009 или выше, и т.п.);
- MS Word и MS Excel.

ЛАБОРАТОРНАЯ РАБОТА №1

ИССЛЕДОВАНИЕ ТРАДИЦИОННЫХ МЕТОДОВ СИММЕТРИЧНОГО ШИФРОВАНИЯ

Цель работы: сформировать практические навыки применения традиционных методов симметричного шифрования.

Задачи: Изучить классические алгоритмы симметричного шифрования. Разработать алгоритмы симметричного шифрования и расшифровывания. Реализовать на их основе программы для защиты текстовых данных.

Требования к отчету.

Отчет по лабораторной работе должен содержать:

1. Титульный лист установленного образца.
2. Постановку задачи.
3. Описание хода выполнения работы
4. Вывод

Длительность работы: 8 академических часов.

Защита работы: собеседование с преподавателем по контрольным вопросам.

Задание: разработать программу, осуществляющую криптографическую защиту информации, содержащейся в файле данных, с помощью алгоритма шифрования, указанного в варианте.

Дополнительное задание: написать программу/модуль генерации случайных ключей шифра, оценить размерность ключевого пространства.

Варианты заданий:

1. Шифрующие таблицы с перестановкой по ключу – размеру таблицы
2. Шифрующие таблицы с числовым ключом

3. Шифрующие таблицы с ключевым словом
4. Шифрующие таблицы с двойной перестановкой по числовому ключу
5. Полибианский квадрат для русского алфавита
6. Полибианский квадрат для английского алфавита
7. Шифрующие таблицы Трисемуса
8. Шифр Уинстона

Вариант повышенной сложности:

9. Магические квадраты

Теоретические сведения

Симметричными называют алгоритмы, в которых шифрование и дешифрование ведется на одном и том же ключе. И этот ключ является секретным. Сам алгоритм зашифровывания, как правило, считается известным всем.

Рассмотрим традиционные (классические) методы симметричного шифрования, отличающиеся простотой и наглядностью.

Табличные шифры перестановки

Табличные шифры появились в эпоху Возрождения (конец XIV столетия). Разработанные в то время шифрующие таблицы по существу задают правила перестановки букв в сообщении. Они относятся к шифрам перестановки и являются блочными шифрами, где длина блока определяется размером таблицы.

Шифрующие таблицы с перестановкой по ключу – размеру таблицы.

Одним из самых примитивных табличных шифров является простая перестановка, для которой ключом служит размер таблицы. Например, сообщение записывается в таблицу поочередно по столбцам. После заполнения таблицы текстом сообщения по столбцам для формирования шифротекста считывают содержимое таблицы по строкам. При расшифровывании действия выполняют в обратном порядке. Естественно, отправитель и получатель сообщения должны заранее условиться об общем ключе в виде размера таблицы.

Шифрующие таблицы с перестановкой по числовым или буквенным ключам.

Несколько большей стойкостью к раскрытию обладает метод шифрования, называемый перестановкой по ключу. Этот метод отличается от предыдущего тем, что столбцы таблицы переставляются по ключевому слову или набору чисел длиной в строку таблицы. В верхней (ключевой) строке таблицы до перестановки записывается ключ, затем столбцы таблицы переставляются в соответствии с алфавитным порядком букв ключа в алфавите или по возрастанию или убыванию цифр ключа. Затем буквы считываются по строкам, получается блок шифротекста.

Пример. Зашифруем сообщение: «ТЕРМИНАТОР ПРИБЫВАЕТ СЕДЬМОГО В ПОЛНОЧЬ» с помощью таблицы 5х7 и ключевого слова «ПЕЛИКАН»:

П	Е	Л	И	К	А	Н
7	2	5	3	4	1	6
Т	Н	П	В	Е	Г	Л
Е	А	Р	А	Д	О	Н
Р	Т	И	Е	Ь	В	О
М	О	Б	Т	М	П	Ч
И	Р	Ы	С	О	О	Ь

А	Е	И	К	Л	Н	П
1	2	3	4	5	6	7
Г	Н	В	Е	П	Л	Т
О	А	А	Д	Р	Н	Е
В	Т	Е	Ь	И	О	Р
П	О	Т	М	Б	Ч	М
О	Р	С	О	Ы	Ь	И

При считывании содержимого правой таблицы по строкам и записи шифротекста группами по пять букв получим шифрованное сообщение:

«ГНВЕП ЛТООА ДРНЕВ ТЕЬИО РПОТМ БЧМОР СОЫЬИ».

Магические квадраты

Применялись в средние века. В те времена считалось, что созданные с помощью магических квадратов шифротексты охраняет не только ключ, но и магическая сила. В качестве ключевой информации используются особенности структуры таблицы.

Магическими квадратами называют квадратные таблицы с вписанными в их клетки последовательными натуральными

числами, начиная от 1, которые дают в сумме по каждому столбцу, каждой строке и каждой диагонали одно и то же число.

Шифруемый текст вписывали в магические квадраты в соответствии с нумерацией их клеток. Если затем выписать содержимое такой таблицы по строкам, то получится шифротекст, сформированный благодаря перестановке букв исходного сообщения.

Пример магического квадрата и его заполнения сообщением ПРИЛЕТАЮ ВОСЬМОГО:

16	3	2	13	О	И	Р	М
5	10	11	8	Е	О	С	Ю
9	6	7	12	В	Т	А	Б
4	15	14	1	Л	Г	О	П

Шифротекст, получаемый при считывании содержим правый таблицы по строкам, имеет вполне загадочный вид:

ОИРМ ЕОСЮ ВТАБ ЛГОП

Число магических квадратов быстро возрастает с увеличением размера квадрата. Существует только один магический квадрат размером 3x3 (если не учитывать его повороты). Количество магических квадратов 4x4 составляет уже 880, а количество магических квадратов 5x5 - около 250000.

Магические квадраты средних и больших размеров могли служить хорошей базой для обеспечения нужд шифрования того времени, поскольку практически нереально выполнить вручную перебор всех вариантов для такого шифра.

Шифрующие таблицы с двойной перестановкой.

Для обеспечения дополнительной стойкости можно повторно зашифровать сообщение, которое уже прошло шифрование. Такой метод шифрования называется двойной перестановкой. В случае двойной перестановки столбцов и строк таблицы перестановки определяются отдельно для столбцов и отдельно для строк. Сначала в таблицу записывается текст сообщения, а потом поочередно переставляются столбцы, а затем строки. При расшифровывании порядок перестановок должен быть обратным.

Табличные шифры замены.

Полибианский квадрат.

Относится к шифрам простой замены, в которых буквы исходного текста заменяются по определенному правилам другими буквами того же алфавита. Одним из первых шифров простой замены считается так называемый полибианский квадрат. За два века до нашей эры греческий полководец и историк Полибий изобрел для целей шифрования квадратную таблицу размером 5x5, заполненную буквами алфавита в случайном порядке.

При шифровании в этом полибианском квадрате находили очередную букву открытого текста и записывали в шифротекст букву, расположенную ниже ее в том же столбце. Если буква текста оказывалась в нижней строке таблицы, то для шифротекста брали самую верхнюю букву из того же столбца. Концепция полибианского квадрата оказалась плодотворной и нашла применение в криптосистемах последующего времени.

Шифрующие таблицы Трисемуса.

В 1508 г. аббат из Германии Иоганн Трисемус написал печатную работу по криптологии под названием "Полиграфия". В этой книге он впервые систематически описал применение шифрующих таблиц, заполненных алфавитом в случайном порядке. Для получения такого шифра замены обычно использовались таблица для записи букв алфавита и ключевое слово. В таблицу сначала вписывалось по строкам ключевое слово, причем повторяющиеся буквы отбрасывались. Затем эта таблица дополнялась не вошедшими в нее буквами алфавита по порядку.

При шифровании находят в этой таблице очередную букву открытого текста и записывают в шифротекст букву, расположенную ниже ее, в том же столбце. Если буква текста оказывается в нижней строке таблицы, тогда для шифротекста берут самую верхнюю букву из того же столбца.

Пример. Для русского алфавита шифрующая таблица может иметь размер 4x8. Выберем в качестве ключа слово БАНДЕРОЛЬ. Шифрующая таблица примет вид:

Б	А	Н	Д	Е	Р	О	Л
Ь	В	Г	Ж	З	И	И	К
М	П	С	Т	У	Ф	Х	Ц

Ч	Ш	Щ	Ы	Ъ	Э	Ю	Я
---	---	---	---	---	---	---	---

При шифровании с помощью этой таблицы сообщения В Ы Л Е Т А Е М П Я Т О Г О получаем шифротекст П Д К З Ы В З Ч Ш Л Ы Й С Й

Шифр Уинстона.

В 1854 г. англичанин Чарльз Уитстон разработал новый метод шифрования биграммами, который называют "двойным квадратом". Свое название этот шифр получил по аналогии с полибианским квадратом. В отличие от полибианского шифр "двойной квадрат" использует сразу две таблицы, размещенные по одной горизонтали, а шифрование идет биграммами (парами). Эти не столь сложные модификации привели к появлению на свет качественно новой криптографической системы ручного шифрования. Шифр "двойной квадрат" оказался очень надежным и удобным и применялся Германией даже в годы второй мировой войны.

Перед шифрованием исходное сообщение разбивают на биграммы. Каждая биграмма шифруется отдельно. Первую букву биграммы находят в левой таблице, а вторую букву - в правой таблице. Затем мысленно строят прямоугольник так, чтобы буквы биграммы лежали в его противоположных вершинах. Другие две вершины этого прямоугольника дают буквы биграммы шифротекста.

Если обе буквы биграммы сообщения лежат в одной строке, то и буквы шифротекста берут из этой же строки. Первую букву биграммы шифротекста берут из левой таблицы в столбце, соответствующем второй букве биграммы сообщения. Вторая же буква биграммы шифротекста берется из правой таблицы в столбце, соответствующем первой букве биграммы сообщения.

Пример. Пусть имеются две таблицы размером со случайно расположенными в них русскими алфавитами.

Ж	Щ	Н	Ю	Р
И	Т	Ь	Ц	Б
Я	М	Е	.	С
В	Ы	П	Ч	

И	Ч	Г	Я	Т
1	Ж	Ь	М	О
3	Ю	Р	В	Щ
Ц	:	П	Е	Л

:	Д	У	О	К	Ъ	А	Н	.	Х
З	Э	Ф	Г	Ш	Э	К	С	Ш	Д
Х	А	1	Л	Ъ	Б	Ф	У	Ы	

Рис. Две таблицы со случайно расположенными символами русского алфавита для шифра "двойной квадрат"

Предположим, что шифруется биграмма исходного текста ИЛ. Буква И находится в столбце 1 и строке 2 левой таблицы. Буква Л находится в столбце 5 и строке 4 правой таблицы. Это означает, что прямоугольник образован строками 2 и 4, а также столбцами 1 левой таблицы и 5 правой таблицы. Следовательно, в биграмму шифротекста входят буква О, расположенная в столбце 5 и строке 2 правой таблицы, и буква В, расположенная в столбце 1 и строке 4 левой таблицы, т.е. получаем биграмму шифротекста ОВ.

Если обе буквы биграммы сообщения лежат в одной строке, например ТО, то биграмма сообщения ТО превращается в биграмму шифротекста ЖБ. Аналогичным образом шифруются все биграммы сообщения:

Сообщение ПР ИЛ ЕТ АЮ _Ш ЕС ТО ГО

Шифротекст ПЕ ОВ ЩН ФМ ЕШ РФ БЖ ДЦ

Шифрование методом "двойного квадрата" дает весьма устойчивый к вскрытию и простой в применении шифр. Взламывание шифротекста "двойного квадрата" требует больших усилий, при этом длина сообщения должна быть не менее тридцати строк.

Порядок выполнения работы:

1. Разработать алгоритмы шифрования и дешифрования блока (потока) открытого текста заданной длины из алфавита Z на заданном ключе с помощью метода, указанного в варианте.
2. Определить алфавит криптосистемы (открытого текста и шифротекста). Если алфавит не задан в варианте, выбрать его самостоятельно, так, чтобы он включал в себя символы используемого в примере открытого текста. Например, русский, английский, ASCII.
3. Написать программу/модуль, реализующую шифрование на заданном ключе открытого текста,

состоящего из символов заданного алфавита. Открытый текст, ключ и шифротекст должны быть представлены отдельными файлами.

4. Написать программу/модуль для реализации алгоритма дешифрования полученного файла шифротекста при известном ключе.
5. Провести тестирование программы на коротких и длинных тестовых примерах

Контрольные вопросы:

1. Что такое ключ?
2. Объясните понятие алфавит?
3. Нарисуйте общую схему системы, использующей симметричное шифрование;
4. Перечислите основные методы шифрования с закрытым ключом;
5. В чём заключается суть режима работы блочного шифра «сцепление блоков шифра»?

ЛАБОРАТОРНАЯ РАБОТА №2

ИССЛЕДОВАНИЕ МЕТОДОВ КРИПТОАНАЛИЗА СИММЕТРИЧНЫХ СИСТЕМ ШИФРОВАНИЯ

Цель работы: сформировать практические навыки применения методов криптоанализа симметричных систем шифрования.

Задачи: Изучить способы оценки практической стойкости алгоритма. Разработать алгоритмы и реализовать программы, использующие алгоритмы криптоанализа для определения параметров алгоритма шифрования, необходимые для оценки практической криптостойкости.

Требования к отчету.

Отчет по лабораторной работе должен содержать:

1. Титульный лист установленного образца.
2. Постановку задачи.
3. Описание хода выполнения работы
4. Вывод

Длительность работы: 8 академических часов.

Защита работы: собеседование с преподавателем по контрольным вопросам.

Задание: Разработать и реализовать алгоритмы криптоанализа для оценки практической стойкости алгоритма, разработанного в лабораторной работе №1.

Теоретические сведения

Способность шифра противостоять всевозможным атакам на него называют стойкостью шифра. Понятие стойкости шифра является центральным для криптографии. Важнейшим для развития

криптографии был результат К. Шеннона о существовании и единственности абсолютно стойкого шифра. Но абсолютно стойкий шифр на практике, как правило, неприменим, поэтому для защиты своей информации законные пользователи вынуждены применять не абсолютно стойкие шифры. Такие шифры, по крайней мере теоретически, могут быть вскрыты, то есть противник с неограниченными ресурсами может вскрыть любой не абсолютно стойкий шифр. Вопрос только в том, хватит ли у него сил, средств и времени для разработки и реализации соответствующих алгоритмов.

В этой ситуации желательно, получить теоретическую оценку стойкости, то есть, например, доказать, что никакой противник не сможет вскрыть выбранный шифр, скажем, за 24 часа. К сожалению, математическая теория ещё не даёт нужных теорем — они относятся к нерешённой проблеме нижних оценок вычислительной сложности задач.

Поэтому у пользователя остаётся единственный путь — получение практических оценок стойкости. Этот путь состоит в том, чтобы определить от какого противника мы собираемся защищать информацию, а также какие силы и средства он сможет применить для его вскрытия. Затем, мысленно стать в положение противника и пытаться с его позиций атаковать шифр, т.е. разрабатывать различные алгоритмы вскрытия шифра, а наилучший из этих алгоритмов использовать для практической оценки стойкости шифра. В общем случае все оценки стойкости системы должны вестись в предположении, что криптоаналитику известен алгоритм шифрования, но он не знает ключа, на котором зашифровано конкретное сообщение. Противник также может знать некоторые характеристики открытых текстов, например, общую тематику сообщений, некоторые стандарты, форматы и т. д.

Итак, стойкость конкретного шифра оценивается только путём всевозможных попыток его вскрытия и зависит от квалификации криптоаналитиков, атакующих шифр. Такую процедуру иногда называют проверкой стойкости. Имеется несколько показателей криптостойкости, среди которых основные:

- количество всех возможных ключей;

- среднее время, необходимое для криптоанализа.

В данной работе мы будем защищаться от пассивной атаки с известной шифрограммой, когда противник может перехватывать

все зашифрованные сообщения, но не имеет соответствующих им открытых текстов. Задача криптоаналитика заключается в том, чтобы раскрыть исходные тексты и вычислить ключ.

Для этого рекомендуется использовать один из двух простейших методов вскрытия шифра

1. Статистическую атаку.
2. Силковую атаку.

Силковая атака

Силковая атака заключается в подборе ключа шифрования путем перебора подряд всех возможных ключей вплоть до нахождения истинного. В отличие от всех других методов эта атака применима для всех типов шифров, но она имеет максимальную сложность реализации и, соответственно, требует максимальных затрат времени и средств.

Алгоритм криптоанализа заключается в следующем: определяется множество всех возможных ключей шифрования для данной криптосистемы и для каждого ключа выполняются шаги:

1. Дешифрование известного шифротекста на этом ключе;
2. Анализ «осмысленности» полученного предполагаемого открытого текста, например, путем поиска соответствующих слов в словаре возможных сообщений;
3. Принятие решения об истинности найденного ключа на основе результатов предыдущих шагов дешифрования.

Данный метод работает тем медленнее, чем больше длина шифротекста, используемого для криптоанализа.

Криптоаналитическая статистическая атака

Криптоаналитическая статистическая атака применима только против систем одноалфавитной замены, главным недостатком которых является возможность взлома шифротекста на основе анализа частот появления букв. Такая атака начинается с подсчета частот появления символов:

1. определяется число появлений каждой буквы в шифротексте.

2. полученное распределение частот букв в шифротексте сравнивается с распределением частот букв в алфавите исходных сообщений, например в английском.
3. буква с наивысшей частотой появления в шифротексте заменяется на букву с наивысшей частотой появления в английском языке и т.д.

По закону больших чисел вероятность успешного вскрытия системы шифрования повышается с увеличением длины шифротекста.

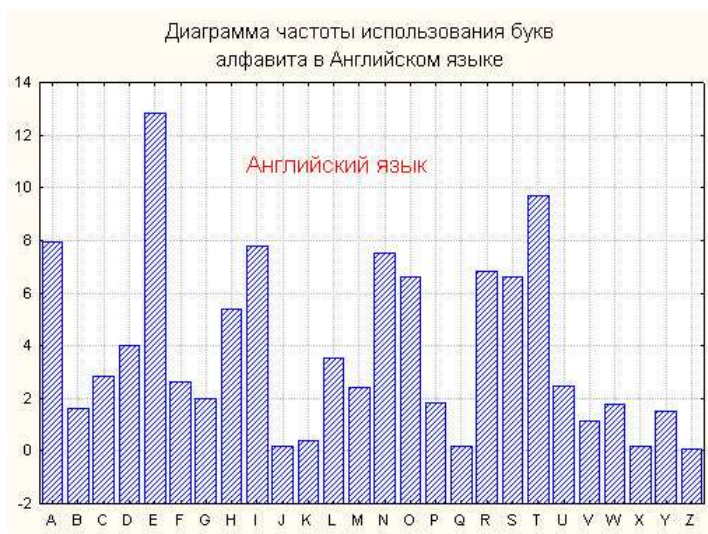
Таблица вероятностей букв в русских текстах.

букв а	пробе л	о	е или ё	а	и	н	т	с	р	в	л
вер- ть	0,175	0,09 0	0,07 2	0,06 2	0,06 2	0,05 3	0,05 3	0,04 5	0,04 0	0,03 8	0,03 5
букв а	к	м	д	п	у	я	з	ы	б	ь или ъ	г
вер- ть	0,028	0,02 6	0,02 5	0,02 3	0,02 1	0,01 8	0,01 6	0,01 6	0,01 4	0,01 4	0,01 3
букв а	ч	й	х	ж	ш	ю	ц	щ	э	ф	
вер- ть	0,012	0,01 0	0,00 9	0,00 7	0,00 6	0,00 6	0,00 4	0,00 3	0,00 3	0,00 2	

Таблица вероятностей букв в английских текстах.

буква	пр-л	e	t	a	o	n	i	s	r
вер- ть	0,185	0,097	0,076	0,064	0,062	0,057	0,056	0,052	0,047
буква	h	l	d	c	u	p	f	m	w

вер- ть	0,04	0,031	0,028	0,025	0,018	0,018	0,018	0,017	0,016
буква	у	в	g	v	к	q	x	j	z
вер- ть	0,015	0,013	0,013	0,007	0,039	0,002	0,002	0,001	0,001

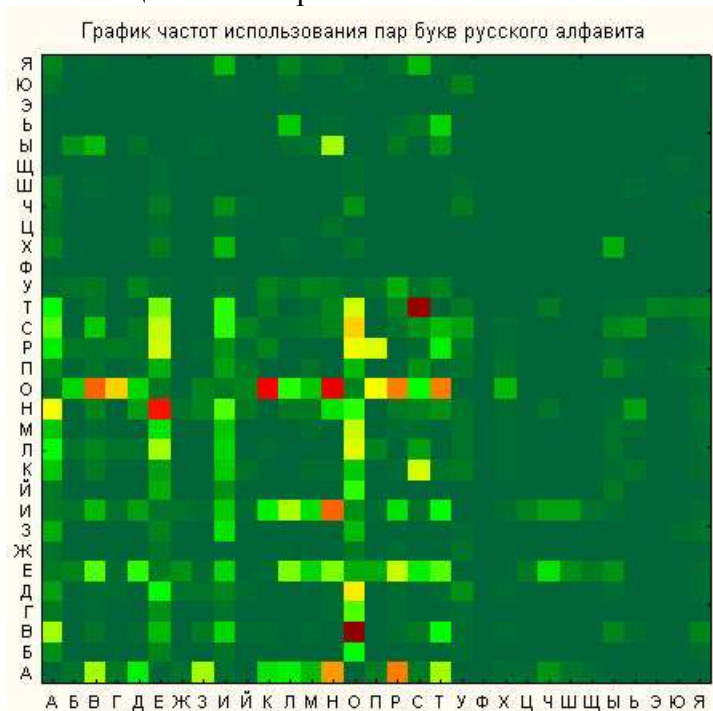


Устойчивыми являются также частотные характеристики биграмм, триграмм и четырехграмм осмысленных текстов.

Для получения более точных сведений об открытых текстах можно строить и анализировать таблицы k-грамм при $k > 2$, однако для учебных целей вполне достаточно ограничиться биграммами. Неравномерность k-грамм (и даже слов) тесно связана с характерной особенностью открытого текста – наличием в нем большого числа повторений отдельных фрагментов текста: корней, окончаний, суффиксов, слов и фраз. Так, для русского языка такими привычными фрагментами являются наиболее частые биграммы и триграммы:

СТ, НО, ЕН, ТО, НА, ОВ, НИ, РА, ВО, КО
СТО, ЕНО, НОВ, ТОВ, ОВО, ОВА

Полезной является информация о сочетаемости букв, то есть о предпочтительных связях букв друг с другом, которую легко извлечь из таблиц частот биграмм.



Имеется в виду таблица, в которой слева и справа от каждой буквы расположены наиболее предпочтительные "соседи" (в

порядке убывания частоты соответствующих биграмм). В таких таблицах обычно указывается также доля гласных и согласных букв (в процентах), предшествующих (или следующих за) данной букве.

Порядок выполнения работы:

Считать, что противнику известен алгоритм шифрования. Выбрать наилучший с его точки зрения алгоритм подбора ключа и обосновать свой выбор. Использовать методы:

1. анализа статистических свойств шифротекста (частот появления букв).
2. силовую атаку (полный перебор ключей).
3. другие (если есть более эффективные)

С помощью программы, реализующей выбранный алгоритм криптоанализа провести эксперимент по вскрытию шифротекстов различного размера.

При использовании статистического криптоанализа использовать таблицы, приведенные выше или подсчитать частоты появления букв используемого алфавита в тексте, частью которого является текст примера.

Для проверки на «осмысленность» полученного текста создать мини-словарь из части слов, встречающихся в тексте примера.

В результате эксперимента определить параметры алгоритма шифрования (размер передаваемого текста, размер и характеристики ключа, объем ключевого пространства и другие параметры алгоритма шифрования), необходимые для оценки практической криптостойкости разработанного в лабораторной работе №1 алгоритма шифрования.

Построить графики зависимости времени криптоанализа от параметров алгоритма шифрования (длины или других параметров ключа, размера шифротекста или др., в зависимости от алгоритмов шифрования и криптоанализа).

Практической криптостойкостью в данной работе будем считать невозможность взлома шифра противником, имеющим в распоряжении один ПК мощности, равной мощности компьютера, на котором делалась работа и 30 минут времени.

Контрольные вопросы:

1. Перечислите виды криптографических атак?
2. Объясните понятие активная криптографическая атака?
3. Какие существуют методы вскрытия шифров?
4. Что называется криптостойкостью?
5. В каких случаях наиболее эффективной является статистическая атака?

ЛАБОРАТОРНАЯ РАБОТА №3

ИССЛЕДОВАНИЕ СОВРЕМЕННЫХ СРЕДСТВ ЗАЩИТЫ ИНФОРМАЦИИ ИСПОЛЬЗУЮЩИЕ АЛГОРИТМЫ СИММЕТРИЧНОГО ШИФРОВАНИЯ

Цель работы: сформировать практические навыки выбора и эксплуатации современных средств криптографической защиты информации.

Задачи: Изучить современные средства криптографической защиты информации. Выбрать СКЗИ для использования в конкретной ситуации, установить, произвести настройку и применить для решения поставленной задачи.

Требования к отчету.

Отчет по лабораторной работе должен содержать:

1. Титульный лист установленного образца.
2. Постановку задачи.
3. Описание хода выполнения работы
4. Вывод

Длительность работы: 8 академических часов.

Защита работы: собеседование с преподавателем по контрольным вопросам.

Задание: Изучить СКЗИ, указанное в варианте, провести тестирование демонстрационной версии программы, дать оценку функционала и практического использования данного СКЗИ.

Варианты заданий:

1. SecretDisk
2. BitLocker
3. Dr.Web Бастион
4. Kaspersky Endpoint Security

5. BestCrypt
6. TrueCrypt
7. RohosDisk

Теоретические сведения

Dr.Web Бастион

Программный продукт Atlansys Bastion Ultimate разработан для широкого круга пользователей персональных компьютеров и ноутбуков, которым необходима надежная и гибкая организация безопасного хранения персональных данных. Atlantis bastion ultimate является частью антивирусного ПО Dr.Web Бастион Pro, а так же может выступать в роли самостоятельного продукта. В Atlansys Bastion Ultimate объединены три ключевые технологии: возможность непосредственного шифрования логических дисков операционной системы, флеш-накопителей и сменных носителей информации, криптоархивов.

Технические характеристики

Поддерживаемые операционные системы	Windows XP, Windows Vista, Windows 7
Поддержка файловых систем	FAT, FAT32, NTFS
Создание криптоархивов	Есть
Создание криптоконтейнеров	Есть
Создание криптодисков с сохранением существующих данных	Есть, для файловых систем FAT, FAT32, NTFS, в том числе на съёмных носителях
Поддерживаемые алгоритмы шифрования ¹	AES, Blowfish
Алгоритмы гарантированного уничтожения файлов	ГОСТ P50739-95, DoD 5220.22-M, NAVSO P-5239-26
Максимальный размер защищаемых дисков	Ограничен файловой системой диска

¹В зависимости от типа поставки продукта набор алгоритмов шифрования может отличаться от указанных

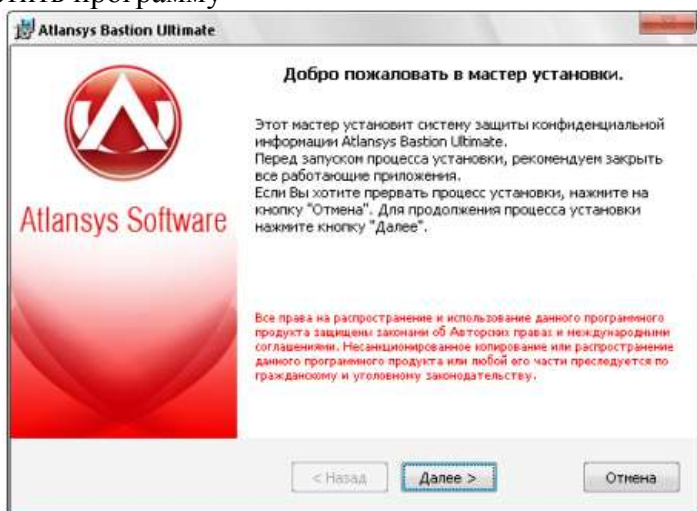
Средняя стоимость Atlantis bastion ultimate как отдельного продукта составляет 1700р

Системные требования

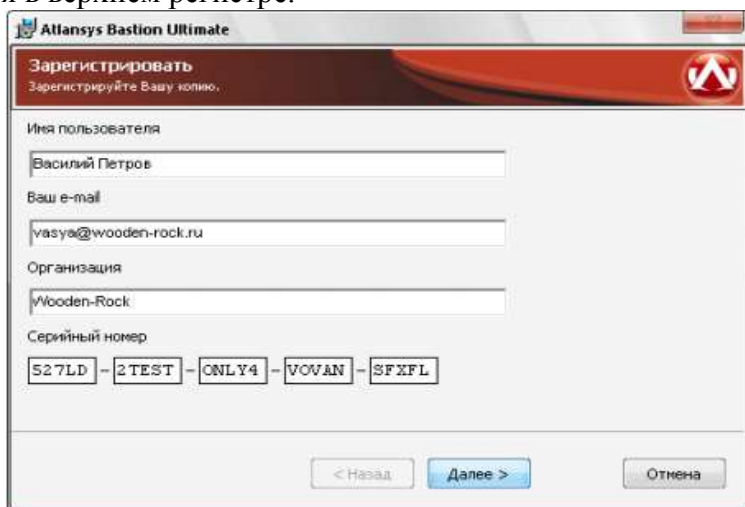
Процессор	Intel Pentium III, AMD Athlon или выше
Операционная система	Windows XP, Windows Vista, Windows 7
Объём оперативной памяти	не менее 128 Мбайт
Свободное место на диске	50 Мбайт
Разрешение экрана	минимум 800х600 пикселей
Привод CD-ROM	при установке с компакт-диска
Подключение к Internet	для регистрации продукта

Установка программного обеспечения

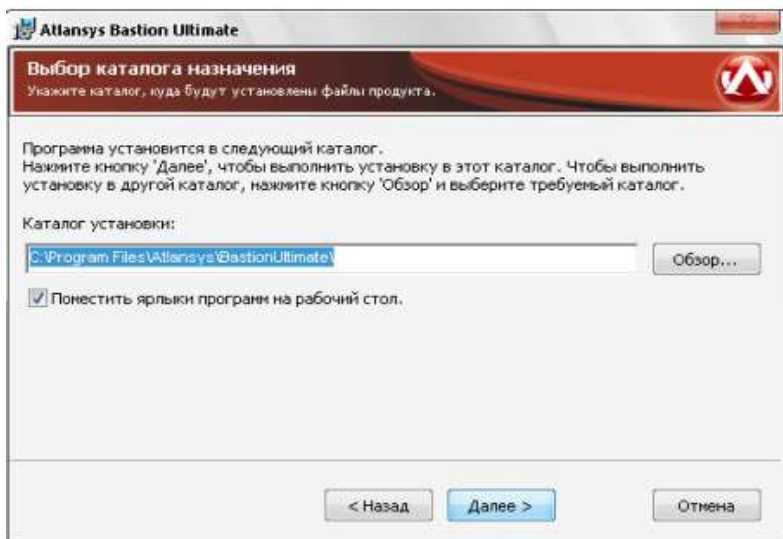
Запустить программу



Ввести имя пользователя, e-mail, название организации, серийный номер, который поставляется с продуктом. Серийный номер содержит пять полей по пять символов, все буквы должны вводиться в верхнем регистре.



Указать имя каталога для установки программы (рекомендуется оставить значение по умолчанию).



Настройки продукта

1. "Язык интерфейса" - выбор языка пользовательского интерфейса.

2. "Настройки криптоархива" - задаёт алгоритм защиты создаваемых криптоархивов. В списке криптосхем выбирается вид текущий криптосхемы. При отключении чекбокса "Использовать только криптостойкие пароли" отключается проверка качества пароля при создании криптоархива. В целях обеспечения высокого уровня безопасности рекомендуется не отключать эту проверку.



3. "Регистрация событий" - задаёт параметры регистрации сообщений в локальной базе и отправки лог-сообщений на внешний syslog-сервер.

"База данных лог-сообщений" - задаётся путь к файлу базы данных лог-сообщений. Рекомендуется оставить этот параметр по умолчанию. База данных состоит из одного файла, который создаётся автоматически при старте системы, если по указанному пути он не был найден.

"Syslog-сервер" - задаётся IP-адрес сервера для передачи на него лог-сообщений по протоколу syslog. Лог-сообщения передаются по протоколу UDP на порт 514, при использовании в сети межсетевых экранов их необходимо настроить для пропуска данных пакетов от рабочих станций к syslog-серверу.



4. "Красная кнопка" - настройки действий при активации механизма "Красной кнопки", который вляется пользователем в критической ситуации, когда необходимо быстро закрыть доступ к открытым криптообъектам. По умолчанию красная кнопка не активирована и никаких действий не производится. При активации красной кнопки она может включаться либо через главное меню "Файл / Активировать красную кнопку", либо с помощью горячих клавиш, комбинация которых задается в настройках. Необходимо так же задать действия при активации красной кнопки. По умолчанию - это закрытие всех открытых криптообъектов, дополнительно можно перезагрузить или выключить компьютер, чтобы полностью блокировать доступ к криптообъектам.



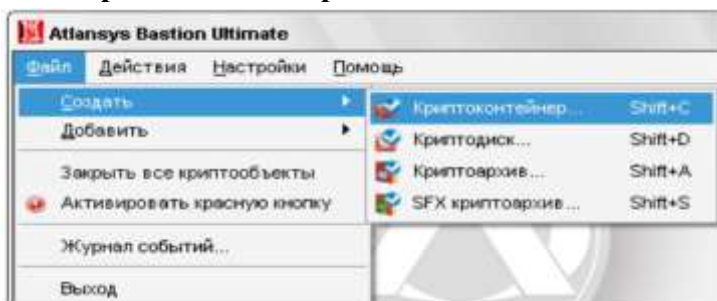
5. "Уничтожение данных" - задаёт алгоритм уничтожения данных по умолчанию, который применяется для уничтожения файлов через контекстное меню Проводника Windows.



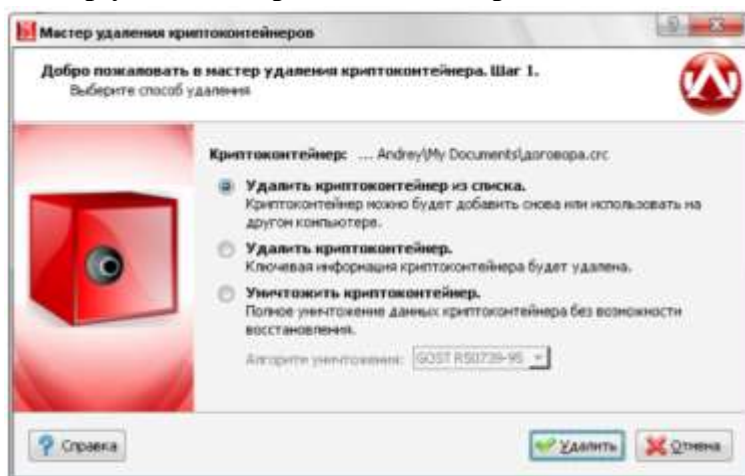
Работа с криптоконтейнерами

Криптоконтейнер представляет собой файл, содержащий полностью зашифрованный образ файловой системы раздела, который можно подключить в систему в виде диска. При этом все приложения и служебные программы Windows будут воспринимать его как полноценное дисковое устройство. Пока криптоконтейнер не открыт, его содержимое невозможно прочитать, так как оно зашифровано криптостойким алгоритмом, поэтому файл криптоконтейнера можно безопасно копировать на различные носители, передавать по сети, использовать для создания архивных копий конфиденциальной информации.

Создание криптоконтейнера



Мастер удаления криптоконтейнеров



Перед удалением криптоконтейнера необходимо закрыть все приложения, которые с ним работают и закрыть криптоконтейнер. После нажатия кнопки «удалить» в контекстном меню контейнера появится окно Мастера удаления криптоконтейнеров, в котором необходимо выбрать один из способов удаления:

- Удалить криптоконтейнера из списка. Удаляет криптоконтейнер только из списка Навигатора. При этом файл криптоконтейнера и все данные, содержащиеся в нем, не удаляются. Данный способ может использоваться при переносе криптоконтейнера на другую рабочую станцию.
- Удалить криптоконтейнер. В файле криптоконтейнера стирается ключевая информация и заголовок, затем

файл удаляется. Так как вся информация в криптоконтейнере зашифрована, то удаление ключевой информации полностью блокирует доступ к данным, содержащимся в криптоконтейнере. Это самый быстрый способ удаления криптоконтейнера, но он не защищает от дешифрования данных с помощью прямого перебора ключей.

- Уничтожить криптоконтейнер. Для гарантированного уничтожения данных помимо удаления ключевой информации, все данные в криптоконтейнере уничтожаются одним из алгоритмов уничтожения.

Алгоритмы уничтожения:

1. Алгоритм по стандарту ГОСТ Р 50739-95 выполняет два цикла записи псевдослучайных значений.
2. Алгоритм по стандарту DoD 5220.22M выполняет два цикла записи псевдослучайных значений и один цикл записи фиксированных значений.
3. Алгоритм по стандарту NAVSO P-5239-26 выполняет два цикла записи фиксированных значений и один цикл записи псевдослучайных значений.

Контрольные вопросы:

1. Перечислите современные программы использующие симметричные методы криптографической защиты информации
2. Какие основные функции выполняет программа в вашем варианте задания?
3. Поясните принцип шифрования и дешифрования информации с помощью программы из вашего варианта задания.
4. Как можно обмениваться ключами со своими корреспондентами?
5. Назовите основные способы шифрования информации при помощи данной программы.

ЛАБОРАТОРНАЯ РАБОТА №4

ИССЛЕДОВАНИЕ МЕТОДОВ АСИММЕТРИЧНОГО ШИФРОВАНИЯ

Цель работы: сформировать практические навыки анализа и применения методов асимметричного шифрования.

Задачи: Изучить алгоритмы асимметричного шифрования. Научиться строить алгоритмы асимметричного шифрования и расшифровывания, а также создавать на их основе программы для защиты текстовых данных

Требования к отчету.

Отчет по лабораторной работе должен содержать:

1. Титульный лист установленного образца.
2. Постановку задачи.
3. Описание хода выполнения работы
4. Вывод

Длительность работы: 8 академических часов.

Защита работы: собеседование с преподавателем по контрольным вопросам.

Задание: разработать программу, осуществляющую криптографическую защиту информации, с помощью алгоритма шифрования RSA, Эль Гамала или Диффи-Хеллмана.

Теоретические сведения

Электронную (цифровую) подпись обычно реализуют на основе криптографического алгоритма с открытым ключом. Такие алгоритмы строятся на основе двух ключей (ключ шифратор (КШ), ключ дешифратор (КД)). Ключи в асимметричных криптосистемах

всегда генерируются парами и состоят из двух частей – открытого ключа (ОК) и секретного ключа (СК).

Ключевая пара	
СК	ОК

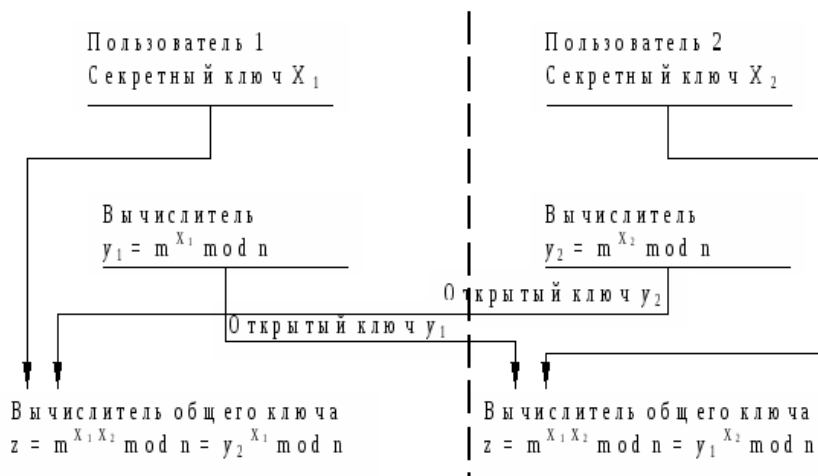
Открытый ключ используется для шифрования информации, является доступным для всех пользователей и может быть опубликован в общедоступном месте для использования всеми пользователями криптографической сети. Дешифрование информации с помощью открытого ключа невозможно.

Секретный ключ является закрытым и не может быть восстановлен злоумышленником из открытого ключа. Этот ключ используется для дешифрования информации и хранится только у одного пользователя – сгенерировавшего ключевую пару.

У. Диффи и М. Хеллман сформулировали требования, выполнение которых обеспечивает безопасность асимметричной криптосистемы :

1. Вычисление ключевой пары (ОК, СК) должно быть достаточно простым.
2. Отправитель, зная открытый ключ получателя, может легко получить шифротекст.
3. Получатель, используя свой секретный ключ, может легко из шифротекста восстановить исходное сообщение.
4. Знание открытого ключа злоумышленником не должно влиять на криптостойкость системы. При попытке вычислить злоумышленником закрытый ключ по открытому, он должен наталкиваться на непреодолимую вычислительную проблему.
5. Злоумышленник, зная шифротекст и открытый ключ, на котором осуществлялось шифрование, при попытке восстановить исходный текст должен наталкиваться на непреодолимую вычислительную проблему

На рисунке представлена система открытого распространения ключей Диффи и Хеллмана.



Система открытого распространения ключей Диффи и Хеллмана

При обмене информацией первый пользователь выбирает случайное число X_1 , из целых чисел $1, \dots, n-1$. Это он держит в секрете, а другому пользователю посылает число

$$y_1 = m^{X_1} \bmod n$$

где m — положительное целое, меньше n . Аналогично поступает и второй пользователь. В результате они могут вычислить z

$$z = m^{X_1 X_2} \bmod n$$

Для того, чтобы вычислить z первый пользователь возводит y_2 в степень X_1 , аналогичным способом поступает и второй пользователь. Не зная X_1 и X_2 , злоумышленник не может вычислить z , имея только y_1 и y_2 .

Эквивалентность этой проблемы проблеме вычисления дискретного логарифма есть главный вопрос криптографии с открытым ключом. До настоящего времени нет простого решения проблемы.

Например, если простое число длиной 1000 бит, то для вычисления y_1 из X_1 потребуется около 2000 умножений 1000

битовых чисел. С другой стороны, имеющимися алгоритмами вычисления логарифмов над полем $GF(n)$ потребуется $2^{100} \approx 10^{30}$ операций. Подобрать же z злоумышленнику, не зная m , n , X_1 или X_2 за разумное время практически невозможно (см. табл.). В таблице представлено сравнение одноключевого криптографического стандарта DES и двухключевого RSA.

Таблица

Характеристика	DES	RSA
1	2	3
вид криптографического алгоритма	Одноключевой	Двухключевой
Скорость работы	Быстрая	Медленная
Наименее затратный криптоанализ	Перебор по всему ключевому пространству	Разложение модуля
Стойкость	Теоретическая	Практическая
Временные затраты на криптоанализ	Столетия	зависят от длины ключа
Время генерации ключа	Миллисекунды	Десятки секунд
Тип ключа	Симметричный	Асимметричный
Длина ключа	56 бит	300 – 600 бит

RSA

В основу криптографической системы с открытым ключом RSA положена сложность задачи факторизации произведения двух больших простых чисел. Для шифрования используется операция возведения в степень по модулю большого числа. Для дешифрования за разумное время (обратной операции) необходимо уметь вычислять функцию Эйлера от данного большого числа, для чего необходимо знать разложения числа на простые множители.

криптографической системе RSA каждый ключ состоит из пары целых чисел. Каждый участник создаёт свой открытый и закрытый ключ самостоятельно. Закрытый ключ каждый из них держит в секрете, а открытые ключи можно сообщать кому угодно

или даже публиковать их. Открытый и закрытый ключи каждого участника обмена сообщениями в криптосистеме RSA образуют «согласованную пару» в том смысле, что они являются взаимно обратными, то есть:

\forall допустимых пар открытого и закрытого ключей (p, s)
 \exists соответствующие функции шифрования $E_p(x)$ и
 расшифрования $D_s(x)$ такие, что
 \forall сообщений $m \in M$, где M — множество допустимых сообщений,

$$m = D_s(E_p(m)) = E_p(D_s(m)).$$

Алгоритм создания открытого и секретного ключей

RSA-ключи генерируются следующим образом:

Выбираются два различных случайных простых числа p и q заданного размера (например, 1024 бита каждое).

Вычисляется их произведение $n = p \cdot q$, которое называется модулем.

Вычисляется значение функции Эйлера от числа n :

$$\varphi(n) = (p - 1) \cdot (q - 1).$$

Выбирается целое число e ($1 < e < \varphi(n)$), взаимно простое со значением функции $\varphi(n)$. Обычно в качестве e берут простые числа, содержащие небольшое количество единичных бит в двоичной записи, например, простые числа Ферма 17, 257 или 65537.

Число e называется открытой экспонентой (англ. public exponent)

Время, необходимое для шифрования с использованием быстрого возведения в степень, пропорционально числу единичных бит в e .

Слишком малые значения e , например 3, потенциально могут ослабить безопасность схемы RSA.

Вычисляется число d , мультипликативно обратное к числу e по модулю $\varphi(n)$, то есть число, удовлетворяющее сравнению:

$$d \cdot e \equiv 1 \pmod{\varphi(n)}.$$

Число d называется секретной экспонентой. Обычно, оно вычисляется при помощи расширенного алгоритма Евклида.

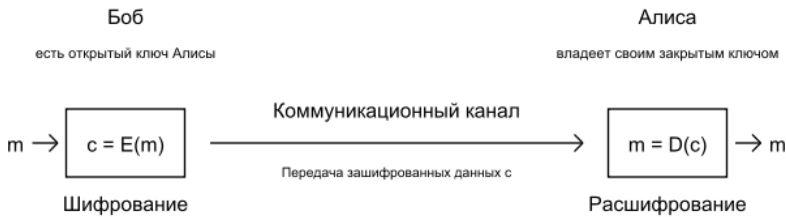
Пара $\{e, n\}$ публикуется в качестве открытого ключа RSA (англ. RSA public key).

Пара $\{d, n\}$ играет роль закрытого ключа RSA (англ. RSA private key) и держится в секрете.

Шифрование и расшифрование

Предположим, Боб хочет послать Алисе сообщение m .

Сообщениями являются целые числа в интервале от 0 до $n - 1$, т.е. $m \in \mathbb{Z}_n$.



Алгоритм:

- Взять *открытый ключ* (e, n) Алисы
- Взять *открытый текст* m
- Зашифровать сообщение c использованием открытого ключа Алисы:

$$c = E(m) = m^e \mod n \quad (1)$$

Алгоритм:

- Принять зашифрованное сообщение c
- Взять свой *закрытый ключ* (d, n)
- Применить закрытый ключ для расшифрования сообщения:

$$m = D(c) = c^d \mod n \quad (2)$$

Данная схема на практике не используется по причине того, что она не является *практически надёжной* (semantically secured). Действительно, односторонняя функция $E(m)$ является *детерминированной* — при одних и тех же значениях входных параметров (ключа и сообщения) выдаёт одинаковый результат —, а это значит, что не выполняется необходимое условие практической (семантической) надёжности шифра.

Алгоритм шифрования сеансового ключа

Наиболее используемым в настоящее время является смешанный алгоритм шифрования, в котором сначала шифруется сеансовый ключ, а потом уже с его помощью участники шифруют свои сообщения симметричными системами. После завершения сеанса сеансовый ключ как правило уничтожается.

Алгоритм шифрования сеансового ключа выглядит следующим образом:



Алгоритм:

- Взять *открытый ключ* (e, n) Алисы
- Создать случайный *сеансовый ключ* m
- Зашифровать сеансовый ключ с использованием открытого ключа Алисы:

$$c = E(m) = m^e \mod n$$

- Расшифровать сообщение C помощью сеансового ключа симметричным алгоритмом:

$$M_A = D_m(C)$$

Алгоритм:

- Принять зашифрованный сеансовый ключ Боба C
- Взять свой *закрытый ключ* (d, n)
- Применить закрытый ключ для расшифрования сеансового ключа:

$$m = D(c) = c^d \mod n$$

- Зашифровать сообщение M_{Ac} помощью сеансового ключа симметричным алгоритмом:

$$C = E_m(M_A)$$

В случае, когда сеансовый ключ больше, чем модуль n , сеансовый ключ разбивают на блоки нужной длины (в случае необходимости дополняют нулями) и шифруют каждый блок.

Контрольные вопросы:

1. Какие одноключевые криптографические преобразования Вам известны?
2. В каких случаях находят применение двухключевые криптографические методы, а в каких – одноключевые?
3. Какие методы побитовой обработки операндов реализованы в VisualStudio?
4. Какие методы побитовой обработки операндов реализованы в RAD Studio?
5. Какой вариант задания отличается максимальной сложностью алгоритма?
6. Каким образом программно измеряется время формирования общего ключа Диффи и Хеллмана?
7. Приведите основные характеристики стандарта RSA.
8. Какую минимальную разрядность общего ключа Диффи и Хеллмана целесообразно использовать?

ЛАБОРАТОРНАЯ РАБОТА №5

ИССЛЕДОВАНИЕ МЕТОДОВ КРИПТОАНАЛИЗА АСИММЕТРИЧНЫХ СИСТЕМ ШИФРОВАНИЯ

Цель работы: сформировать практические навыки анализа и применения асимметричных систем шифрования.

Задачи: Изучить способы оценки практической стойкости алгоритма. Разработать алгоритмы и реализовать программы, использующие приемы криптоанализа для определения параметров алгоритма шифрования, необходимые для оценки практической криптостойкости.

Требования к отчету.

Отчет по лабораторной работе должен содержать:

1. Титульный лист установленного образца.
2. Постановку задачи.
3. Описание хода выполнения работы
4. Вывод

Длительность работы: 8 академических часов.

Защита работы: собеседование с преподавателем по контрольным вопросам.

Задание: Разработать алгоритмы криптоанализа для оценки практической стойкости алгоритма, разработанного в лабораторной работе №4.

Теоретические сведения

В алгоритмах Диффи-Хеллмана и RSA нужно вычислять $R = y^x \bmod n$, где n - известен, а y может быть найден перехватом. Цель нападающего - найти секретный ключ x . Для осуществления атаки необходимо, чтобы жертва вычисляла $y^x \bmod n$ для нескольких

значений y , где y , n и время вычислений известны атакующему. (Если секретная экспонента x будет меняться для каждой операции, атака не сработает). Необходимая информация и время вычислений могут быть получены пассивным подслушиванием, так как нападающий может записывать сообщения, посылаемые цели атаки и измерять время получения ответа для каждого y .

Атака может быть осуществлена при любой реализации алгоритма модульного возведения в степень, где время вычисления не является фиксированным, но сначала лучше рассмотреть обычный алгоритм вычисления $R = y^x \bmod n$, где x - длиной w бит:

```

Let  $s_0 = 1$ .
For  $k = 0$  upto  $w-1$ :
  If (bit  $k$  of  $x$ ) is 1 then
    Let  $R_k = (s_k \cdot y) \bmod n$ .
  Else
    Let  $R_k = s_k$ .
  Let  $s_{k+1} = R_k^2 \bmod n$ .
EndFor.
Return  $(R_{w-1})$ .

```

:

Атака позволит тому, кто знает биты $0..(b-1)$, найти бит b . Чтобы получить экспоненту целиком, надо начать с b равным 0 и повторять атаку до тех пор, пока вся экспонента не станет известна.

Т.к. первые b бит известны, нападавший может вычислить первые b итераций цикла и найти значение s_b . Следующая итерация потребует первого неизвестного бита. Если этот бит установлен в 1, то будет вычислен $R_b = (s_b \cdot y) \bmod n$. Если бит равен нулю, умножение будет пропущено.

Рассмотрим сначала атаку в следующем гипотетическом случае. Пусть атакуемая система использует очень быструю функцию модульного умножения, но которая иногда потребляет гораздо большее время, чем вся функция модульного возведения в степень. Тогда для некоторых значений s_b и y вычисления $R_b = (s_b \cdot y) \bmod n$ будут чрезвычайно медленны, и, используя знания о реализации атакуемой системы, нападающий может определить, каковы эти значения. Если время вычисления всей функции модульного возведения в степень мало, но при этом $R_b = (s_b \cdot y) \bmod n$ должно было бы вычисляться медленно, то бит b , очевидно, равен 0. Наоборот, если долгие по времени вычисления $R_b = (s_b \cdot y) \bmod n$

всегда приводят к медленному модульному возведению в степень, это бит, вероятно, установлен в 1. Как только бит b станет известен, атакующий может проверить, что полное время операций медленно всякий раз, когда $s_{b+1} = R^2 b \bmod n$ ожидается быть медленным. Эти же самые предположения могут использоваться и дальше, чтобы найти следующие биты.

Коррекция ошибок:

Если бит b угадан неправильно, значения, вычисляемые для $R_k \geq b$ будут также неверны, и далее результат будет, в общем, случайный. Время, требуемое для умножения после ошибки не будет отражаться на полном времени возведения в степень. Атака таким образом имеет свойство "обнаружения ошибки"; после неправильного предположения о значении бита не будет наблюдаться никакой значимой корреляции.

Свойство "обнаружения ошибки" может использоваться для ее коррекции. Например, нападавший может поддерживать список наиболее вероятных промежуточных экспонент наряду со значением, соответствующей правильной вероятности. Атака продолжается для единственного наиболее вероятного кандидата. Если текущее значение неправильно, оно будет иметь тенденцию опускаться в списке наиболее вероятных значений, тогда как правильное должно было бы повышаться. Методы исправления ошибок увеличивают необходимую память и процессорное время, но при этом значительно уменьшают число необходимых значений.

Общая схема атаки:

Атаку можно трактовать как проблему распознавания сигналов. "Сигнал" состоит из вариаций измерения времени для известных бит, "шум" - из погрешностей измерения времени и вариаций измерения времени для неизвестных бит. Свойства "сигнала" и "шума" определяют количество замеров времени, требуемых для атаки. Пусть получено j сообщений y_0, y_1, \dots, y_{j-1} и им соответствующие измерения времени T_0, T_1, \dots, T_{j-1} . Вероятность, что предположение x_b для первых b бит правильно, пропорциональна

$$P(x_b) \propto \prod_{i=0}^{j-1} F(T_i - t(y_i, x_b))$$

, где

$t(y_i, x_b)$ - время, требуемое для первых b итераций цикла вычисления $y_i^x \bmod n$ с использованием бит x_b ,

F - ожидаемая функция распределения вероятности $T-t(y, x_b)$ по всем значениям y и правильному x_b . Т.к. F определена как распределение вероятности $T_i - t(y_i, x_b)$, если x_b правильно, то это - лучшая функция для предсказания $T_i - t(y_i, x_b)$. Обратите внимание, что измерения времени и промежуточные значения s могут использоваться для улучшения оценки F .

При правильном предположении для x_{b-1} имеются два возможных значения для x_b . Вероятность того, что x_b - является правильным, а x_b' - неправильным, может быть найдена как

$$\frac{\prod_{i=0}^{j-1} F(T_i - t(y_i, x_b))}{\prod_{i=0}^{j-1} F(T_i - t(y_i, x_b)) + \prod_{i=0}^{j-1} F(T_i - t(y_i, x_b'))}.$$

На практике эта формула не очень полезна, т.к. поиск F потребует слишком больших усилий.

Упрощение атаки:

К счастью, нет необходимости вычислять F . Каждое наблюдение времени состоит из $T = e + \sum_{i=0}^{w-1} t_i$, где t_i - время, требуемое для умножения и возведения в степень для бита i , а e включает ошибку измерения, время на инкремент цикла и т.п. Имея предположение x_b , нападающий может вычислить $\sum_{i=0}^{b-1} t_i$ для каждого значения y ⁴. Если x_b правильно, то вычитая это время из T , получим $e + \sum_{i=0}^{w-1} t_i - \sum_{i=0}^{b-1} t_i = e + \sum_{i=b}^{w-1} t_i$. Т.к. времена модульного умножения не зависят от друг друга и от ошибки измерения, дисперсия $e + \sum_{i=b}^{w-1} t_i$ по всем наблюдаемым значениям, ожидается, будет равна $\text{Var}(e) + (w-b)\text{Var}(t)$. Однако, если только первые $c < b$ бит угаданы правильно, ожидаемая дисперсия будет $\text{Var}(e) + (w-b+2c)\text{Var}(t)$. Итерации с правильными предположениями уменьшают ожидаемую дисперсию на $\text{Var}(t)$, каждая итерация после неправильного предположения увеличивает дисперсию на $\text{Var}(t)$. Вычислять

дисперсии легко, и это обеспечивает хороший способ идентификации правильного бита.

Теперь возможно оценить число значений, требуемых для атаки. Предположим, что нападавший имеет j точных времен измерения и два предположения относительно первых b битов w -битного значения, одно правильное и другое - неправильное с первой ошибкой в бите c . Для каждого предположения время измерения может быть сверено с $\sum_{i=0}^{b-1} t_i$. Если такая сверка имеет меньшую дисперсию, то это - правильное предположение. Возможно аппроксимировать t_i с использованием независимых стандартных нормальных переменных. Если $\text{Var}(e)$ незначительно, ожидаемая вероятность правильности предположения

$$P \left(\sum_{i=0}^{j-1} \left(\sqrt{w-b} X_i + \sqrt{2(b-c)} Y_i \right)^2 > \sum_{i=0}^{j-1} \left(\sqrt{w-b} X_i \right)^2 \right) = P \left(2\sqrt{2(b-c)(w-b)} \sum_{i=0}^{j-1} X_i Y_i + 2(b-c) \sum_{i=0}^{j-1} Y_i^2 > 0 \right), \text{ где}$$

X и Y - нормальные случайные переменные с законом $(0, 1)$.

Т.к. j относительно большое, $\sum_{i=0}^{j-1} Y_i^2 \approx j$ и $\sum_{i=0}^{j-1} X_i Y_i$

приблизительно нормальны с законом $(0, \sqrt{j})$. Отсюда

$$P \left(2\sqrt{2(b-c)(w-b)}(\sqrt{j}Z) + 2(b-c)j > 0 \right) =$$

$$P \left(Z > -\frac{\sqrt{j(b-c)}}{\sqrt{2(w-b)}} \right), \text{ где } Z - \text{стандартная нормальная}$$

случайной переменной. Интегрирование для нахождения

вероятности правильного предположения дает $\Phi \left(\sqrt{\frac{j(b-c)}{2(w-b)}} \right)$,

где $\Phi(x)$ - область под стандартной нормальной кривой от $-\infty$ до x . Требуемое число значений j таким образом пропорционально длине экспоненты w . Число измерений может быть уменьшено, если

нападающий может выбирать такие входные данные, чтобы иметь экстремумы времени в тех битах экспоненты, которые его интересуют.

На рис. показано распределение выполнения модульного умножения 10^6 раз. Для этого использовался пакет RSAREF на 120-MHz компьютере Pentium с ОС MS-DOS. Распределение было построено, замеряя время одного миллиона вычислений ($a^b \bmod n$), где a и b получались как результат фактических операций модульного возведения в степень со случайными входными данными. В качестве n использовалось первое 512-битное простое число из демонстрационной программы метода Диффи-Хеллмана пакета RSAREF. Все наиболее отклонившиеся значения измерений (которые заняли более 1300 мкс) были отвергнуты. Распределение на рис. 1 имеет мат. ожидание $\bar{H} = 1167.8$ мкс и стандартное отклонение $\sigma = 12.01$ мкс. Ошибка измерения мала; испытания проводились дважды и средняя разница между измерениями составила менее 1 мкс. RSAREF использует одну и ту же функцию для возведения в квадрат и умножения; таким образом времена возведения в квадрат и умножения имеют идентичные распределения.

RSAREF вычисляет заранее $y^2 \bmod n$ и $y^3 \bmod n$ и обрабатывает сразу два бита. Всего, 512-битное модульное возведение в степень со случайным показателем степени из 256 бит требует 128 итераций цикла модульного умножения и общее количество действий модульного умножения и возведения в квадрат приблизительно 352, т.к. каждая итерация выполняет два действия возведения в квадрат и, если любой из двух бит не 0, одно умножение. Атака может быть скорректирована для этого случая, чтобы прибавлять пару бит и оценивать четыре значения кандидата в каждой позиции экспоненты вместо двух.

Так как модульные умножения потребляют наибольшее количество общего времени, ожидается, что распределение времени будет приблизительно нормально с законом (\bar{H}, σ) , где $\bar{H} \geq 1167.8 \cdot 352 = 411065.6$ мкс и $\sigma > 12.01 \sqrt{352} = 225$ мкс. Рисунок 2 показывает измерения из 5000 фактических действий,

использующих тот же самый компьютер и модуль n , где получилось распределение с $\bar{H} = 419,901$ мкс и $\sigma = 235$ мкс.

FIGURE 1
RSAREF Modular Multiplication Times

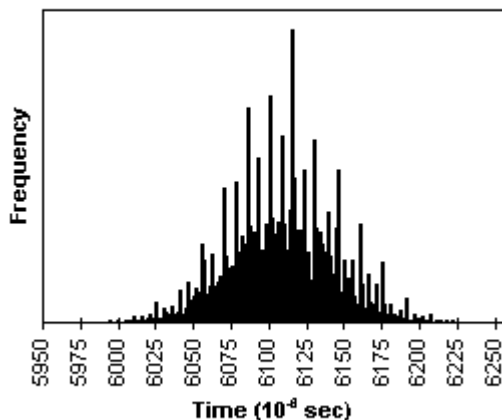
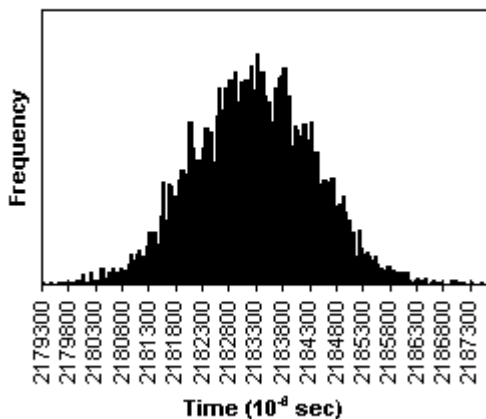


FIGURE 2
RSAREF Modular Exponentiation Times



При 250 измерениях времени вероятность того, что вычитание времени правильной итерации из каждого значения будет сокращать общую дисперсию на большую величину, чем вычитание

неправильной итерации, оценивается формулой $\Phi \left(\sqrt{\frac{j(b-c)}{2(w-b)}} \right)$, где $j=250$, $b=1$, $c=0$ и $w=127$. (Имеются 128 итераций цикла в RSAREF для экспоненты в 256 бит, но первая итерация игнорируется). Правильные предположения таким образом

ожидаются с вероятностью $\Phi \left(\sqrt{\frac{250(1-0)}{2(126)}} \right) \approx 0.84$. 5000 значений из рисунка 2 были разделены на 20 групп по 250 в каждой, и сравнивались дисперсии, получаемые при вычитании времени для неправильных и правильных итераций цикла, для каждой из 127 пар бит. Из 2450 испытаний, в 2168 дисперсия после вычитания времени неправильного цикла была больше, чем после вычитания времени правильного, составив вероятность 0.885. Первые биты наиболее трудны для анализа, но по мере увеличения b вероятность должна улучшаться. (Испытания не использовали этого свойства). Важно отметить, что использовались точные измерения времени; погрешности измерений с относительно большим отклонением по сравнению с временем вычисления модульного возведения в степень, будут увеличивать размер выборки.

Атака в вычислительном отношении весьма проста. При использовании RSAREF, нападающий должен оценить четыре варианта для каждой пары бит. Таким образом, нападающий должен сделать всего лишь в четыре раза большее число действий, выполняемых жертвой (не считая время, потраченное впустую на неправильные предположения).

Маскирование временных характеристик:

Наиболее очевидный путь предотвращения атак временным анализом состоит в том, чтобы все действия занимали точно одно и то же время. К сожалению, это часто бывает трудно. Создание программного обеспечения, особенно платформу-независимого, таким образом, чтобы оно выполнялось в фиксированное время, весьма трудно, потому что оптимизация компилятора, попадания в программный кэш, время выполнения инструкций и другие факторы могут внести неожиданные колебания во время исполнения. Если для задержки возвращаемых результатов до определенного времени используется таймер, то факторы типа "отклика системы" (*responsiveness*) или энергопотребления все еще

могут служить признаками фактического окончания операции. Также некоторые операционные системы показывают процент использования ЦПУ процессом. Более того, реализации с фиксированным временем будут медленными; большинство оптимизаций не сможет быть использовано, так как все действия должны занимать время, исполнения самой медленной операции. (Примечание: если необязательный шаг $R_i = (s_i \cdot y) \bmod n$ выполнять всегда, то такую реализацию нельзя назвать выполняющейся в фиксированное время, т.к. атакующий может использовать временные характеристики возведения в квадрат и последующих действий цикла).

Другой подход состоит в том, чтобы сделать измерения времени настолько неточными, чтобы атака стала невыполнимой. Случайные задержки, добавленные к процессу, увеличат необходимое количество шифрованного текста для нападающего, но он сможет преодолеть это увеличением числа измерений. Требуемое число значений грубо оценивается как квадрат шума. Например, если модульное возведение в степень, временные характеристики которой имеют стандартное отклонение 10 мс, может быть взломано за 1000 измерений времени, добавление случайной нормально распределенной задержки со стандартным отклонением в 1 сек. потребует приблизительно $\left(\frac{1000ms}{10ms}\right)^2 (1000) = 10^7$ значений. (Примечание: задержка должна быть несколько секунд, чтобы ее стандартное отклонение было бы в 1 секунду). Хотя 10^7 значений - больше, чем наибольшее возможное количество значений, которое можно собрать, фактор безопасности в 10^7 обычно не считается достаточно адекватным.

Предотвращение атак

К счастью, имеется и лучшее решение. Методы, используемые для скрытия подписей [1], могут быть приспособлены, чтобы предотвратить знание атакующего входных данных к модульной функции возведения в степень. Перед вычислением модульной экспоненты, выберите случайную пару (v_i, v_f) , такую, что $v_f^{-1} = v_i^x \bmod n$. Для метода Диффи-Хеллмана наиболее просто выбрать случайное v_i , затем вычислить $v_f = (v_i^{-1})^x \bmod n$. Для RSA лучше выбрать случайное v_f , взаимно простое к n , затем вычислить $v_i = (v_f$

$v_i^e \bmod n$, где e - открытая экспонента. Перед модульным возведением в степень, входные данные должны быть умножены на $v_i \bmod n$, а позже результат корректируется умножением на $v_f \bmod n$. При этом система должна отклонять сообщения, равные 0 ($\bmod n$).

Вычисление инверсий $\bmod n$ медленно, поэтому не практично выбирать новую случайную (v_i, v_f) пару для каждой новой экспоненты. Более того, само вычисление $v_f = (v_i^{-1})^x \bmod n$ может быть подвергнуто временному анализу. Однако (v_i, v_f) пары не должны быть использоваться повторно, так как они могут быть раскрыты временным анализом, что делает секретную экспоненту уязвимой. Эффективное решение этой проблемы - модернизация v_i and v_f перед каждым шагом модульного возведения в степень, вычисляя $v_i' = v_i^2$ and $v_f' = v_f^2$. Общие затраты на это будут малы (2 модульных возведений в квадрат, которые может быть вычислены заранее, плюс 2 модульных умножения). Возможно использовать и более сложную модернизацию пары, используя порядки выше 2, умножение на другую пару (v_i, v_f) и т.п., но это не принесет новых преимуществ.

Если (v_i, v_f) хранится в секрете, то нападающий не имеет никакой полезной информации относительно входных данных для функции модульного возведения в степень. Следовательно, максимум того, что атакующий может узнать, - это общее распределение времени для действий возведения в степень. Практически, эти распределения близки к нормальным и 2^w экспонент нельзя различить. Однако, специально разработанная злоумышленником функция модульного возведения в степень могла бы теоретически иметь распределение с острой пиками, соответствующими битам экспоненты, и в этом случае скрывание, *вероятно*, не предотвратит временной анализ.

Даже используя скрывание, распределение будет отражать среднее время операции, что можно использовать, чтобы найти хаммингский вес экспоненты. Если важна анонимность или если требуется дальнейшая маскировка, то экспонента может быть умножена на случайную $\varphi(n)$ перед каждым модульным возведением в степень. Если это делается, то нужно убедиться, что процесс умножения не имеет временных характеристик, которые

могут раскрыть $\varphi(n)$. Такая техника может быть полезна и в предотвращении нападений, которые используют утечку информации в течение модульного возведения в степень из-за электромагнитных излучений, колебаний в производительности системы, изменений в потреблении энергии и т.д. вплоть до изменения бит экспоненты в каждой операции.

Контрольные вопросы:

1. Какие существуют виды атак на ассиметричные криптографические алгоритмы?
2. В чем заключается суть временной атаки?
3. Какими методами необходимо воспользоваться для предотвращения подобных атак?
4. Какая длина ключа в алгоритме RSA считается надежной после публикации в 2010 году статьи об успешной дешифрации алгоритма?
5. Сравните криптостойкость основных алгоритмов асимметричного шифрования?

ЛАБОРАТОРНАЯ РАБОТА №6

ИССЛЕДОВАНИЕ СОВРЕМЕННЫХ СРЕДСТВ ЗАЩИТЫ ИНФОРМАЦИИ, ИСПОЛЬЗУЕМЫХ ДЛЯ СИСТЕМ ЭЛЕКТРОННОГО ДОКУМЕНТООБОРОТА

Цель работы: сформировать практические навыки анализа и работы с современными средствами защиты информации, применяемых в системах электронного документооборота.

Задачи: Изучить современные средства криптографической защиты информации. Научиться выбирать подходящие СКЗИ для использования в конкретной ситуации, устанавливать, производить настройку и использовать для решения поставленных задач.

Требования к отчету.

Отчет по лабораторной работе должен содержать:

1. Титульный лист установленного образца.
2. Постановку задачи.
3. Описание хода выполнения работы
4. Вывод

Длительность работы: 8 академических часов.

Защита работы: собеседование с преподавателем по контрольным вопросам.

Задание: Изучить СКЗИ, указанные в варианте, провести тестирование программ, дать оценку функционала и практического использования данных СКЗИ.

Варианты заданий:

1. GnuPG и КриптоПро CSP
2. GnuPG и VipNet CSP
3. GnuPG и SberSign
4. PGP и криптосервер ВТБ

5. PGP и Континент

Порядок выполнения работы

1. Изучите разделы методических указаний.
2. Установите программное обеспечение GnuPG/PGP
3. Создайте личный ключ шифрования.
4. Запишите свои ключи на носитель информации для дальнейшего использования.
5. Запишите свой публичный (открытый) ключ на переносной носитель (или перешлите по сети) и передайте его участникам электронного обмена информацией.
6. Получите открытые ключи от участников обмена информацией и импортируйте их на свой компьютер.
7. Создайте файл(ы) в MS Word, зашифруйте его и передайте участнику обмена информацией, ключом которого шифровался файл.
8. Получите от участников обмена информацией зашифрованные файлы и расшифруйте их.
9. Подготовьте отчёт о проделанной работе.

PGP.

Сокращение от слов (Pretty Good Privacy, достаточно хорошая секретность). Именно так называлась программа, которая была написана Филиппом Циммерманном в 1991 году. Она была предназначена для удобного шифрования данных. У этой программы довольно сложная судьба, за нее Циммермана пытались судить в США, программу перекупали друг у друга разные фирмы, в данный момент PGP принадлежит фирме Symantec, и PGP является коммерческой программой.

Изначально PGP умел только шифровать симметричным алгоритмом, но теперь PGP — это не только шифрование с открытым ключом, шифрование в ней — это заключительный этап обработки данных. До этого данные могут быть сжаты, зашифрованы алгоритмом с симметричным ключом, и затем уже происходит шифрование с открытым ключом. Причем, на каждом

этапе могут использоваться различные алгоритмы. Более того, тот алгоритм, который будет использоваться в дальнейшем, может использовать для шифрования несколько открытых ключей таким образом, что зашифрованное сообщение смогут прочитать несколько человек (как вариант, Алиса может зашифровать сообщение используя открытый ключ Боба и свой открытый ключ, тогда она сама тоже сможет расшифровать посылаемое сообщение). Другой интересной особенностью является то, что сгенерированные ключи могут иметь срок годности, после которого они считаются недействительными.

Итак, PGP — это коммерческая программа, но пока этой программой владел Циммерман и фирма PGP, Inc. (затем PGP, Inc. поглотит фирма McAfee, которая тогда называлась Network Associates, Inc), они предложили стандарт OpenPGP (RFC 4880 и предыдущая его версия RFC 2440), который описывает то, как работает PGP для того, чтобы сторонние программисты могли бы создавать свой софт, совместимый с PGP.

Кроссплатформенная реализация с открытым кодом стандарта OpenPGP называется GnuPG. Самое интересное, что изначально GnuPG начал писать немецкий программист Вернер Кох, а потом немецкое министерство экономики и технологий профинансировало портирование GnuPG под Windows (gpg4win).

GnuPG может использовать меньшее количество алгоритмов по сравнению с PGP, так как в GnuPG реализованы только те алгоритмы, на которые нет каких-либо патентных ограничений, хотя некоторые запатентованные алгоритмы можно подключать как дополнительные модули.

Мы (от лица Алисы и Боба) будем использовать свободную реализацию стандарта OpenPGP — GnuPG. Для демонстрации Алиса будет использовать реализацию GnuPG под Windows (для повторения примеров качайте полную версию), а Боб — под Линукс. Так как GnuPG — это набор консольных утилит, то для простого пользователя проще воспользоваться какой-нибудь графической оболочкой над ней, в данном посте это будет кроссплатформенная программа Kleopatra, которая входит в том числе и в Windows-версию GnuPG.

GnuPG под Windows ставится очень просто, запускаем инсталлятор, а дальше Next->Next->Next... Под Линукс программа

ставится еще проще, например, под Ubuntu просто выполняем команду:

```
sudo apt-get install kleopatra
```

а дальше все необходимое само скачается и установится.

На некоторых шагах поведение Kleopatra под Windows и Линукс может отличаться, но не значительно.

Генерация ключей

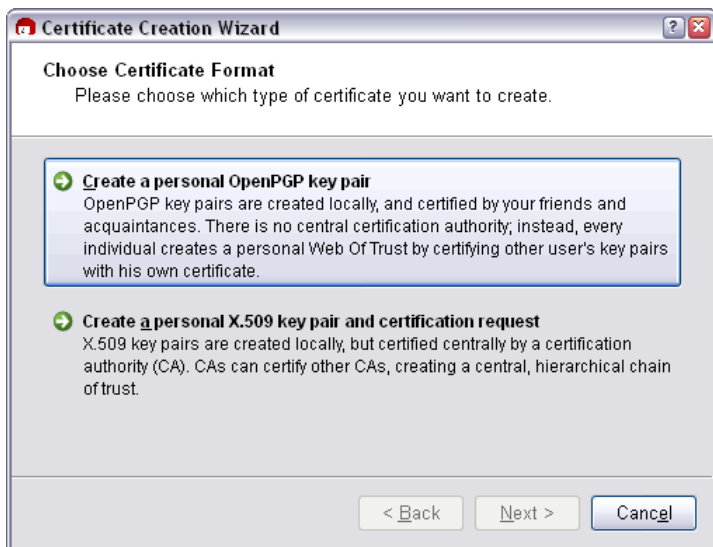
Первое, что нужно сделать Алисе и Бобу — это сгенерировать себе по паре ключей (открытый + закрытый). Вообще-то в дальнейших примерах сообщения будет слать только Алиса, поэтому пару ключей нужно создавать только Бобу, но так как у нас Алиса работает под Windows, то я покажу процесс генерации ключей на ее примере, чтобы это было ближе к большинству читателей.

Алиса запускает Kleopatra и видит главное окно программы:

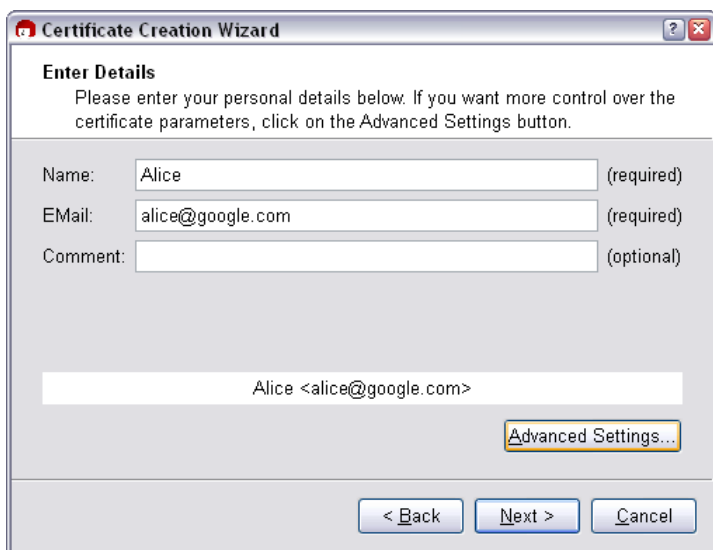


В большой таблице, которая на этом скриншоте пустая, отображаются известные программе ключи (свои и чужие открытые)

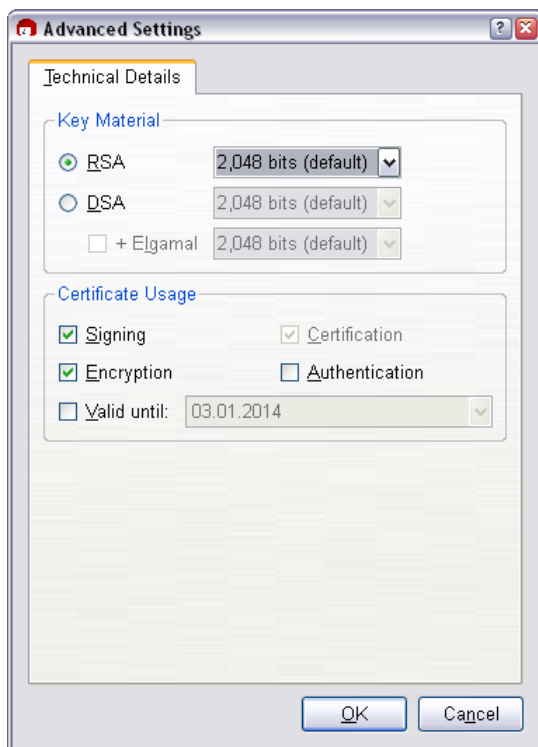
Ключи в программе называются сертификатами. Чтобы создать новую пару ключей она выбирает пункт меню *File -> New Certificate...*, появится окно, показанное на следующем скриншоте:



Нас в данный момент будет интересовать работа по стандарту OpenPGP, поэтому нужно нажать верхнюю кнопку и перейти в окно, где надо заполнить информацию о владельце ключа:



В принципе, можно нажимать кнопку *Next*, но любопытные могут заглянуть в настройки сертификата:

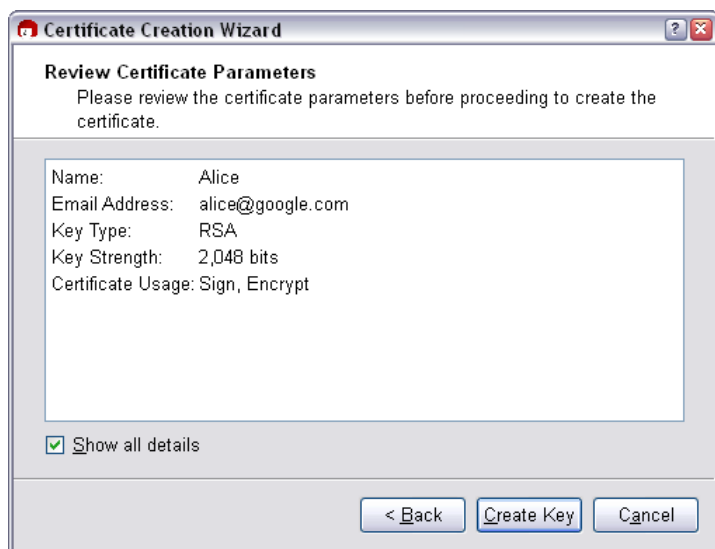


Настройки по умолчанию сейчас нас устроят. Ключ будет создаваться для алгоритма RSA с разрядностью 2048 бит, на данный момент это считается устойчивый ко взлому ключ. Этот ключ можно будет использовать для шифрования (Encryption) и для электронной подписи (Signing). Срок годности ключа ограничивать не будем, а если надо, чтобы сертификат «протух» после определенной даты, то надо установить галку Valid until и установить до какой даты ключ будет действовать.

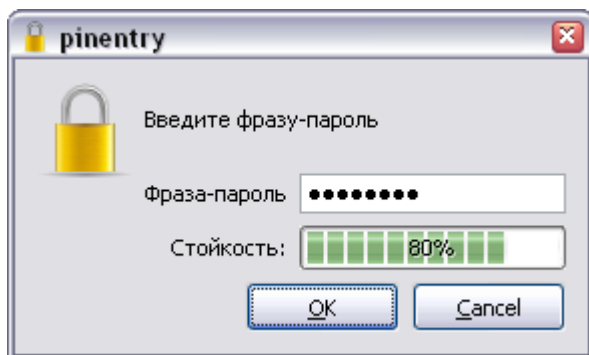
Электронная подпись выходит за рамки этого поста, но пару слов все-таки о ней скажу. По сути это шифрование наоборот. Представьте, что открытый и закрытый ключ поменяли местами. Алиса шифрует текст с помощью своего закрытого ключа и оставляет это сообщение где-то в интернете вместе с открытым ключом, с помощью которого любой может расшифровать ее сообщение. Таким образом получается, что если расшифровать сообщение Алисы удалось, значит оно было подписано ее закрытым

ключом, следовательно это сообщение оставила именно Алиса, а не кто-то другой. Правда, здесь возникает проблема подтверждения публичного ключа, что это именно ключ Алисы. Для этого служат различные сервисы сертификации, с помощью которых можно проверить, кому принадлежит тот или иной ключ, который у них зарегистрирован.

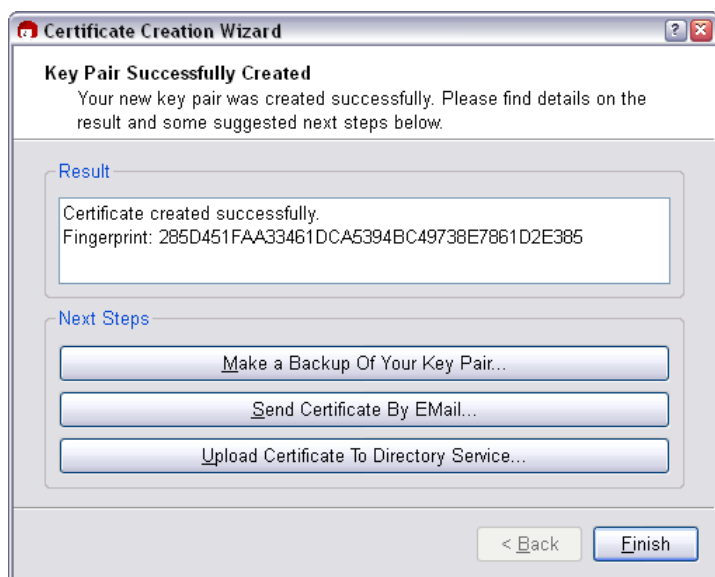
Вернемся в Kleopatra. Настройки оставляем без изменений и переходим к следующему окну, где можно окончательно убедиться в верности введенных данных:



После этого будет предложено ввести пароль для шифрования закрытого ключа, чтобы он не хранился в явном виде.

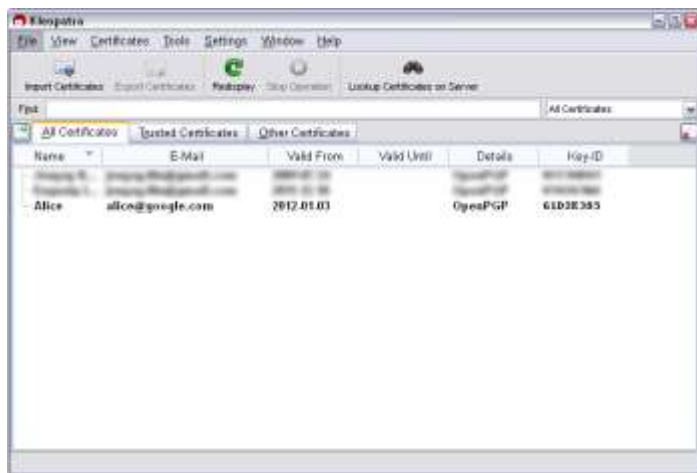


Придумываем надежный пароль, затем его подтверждаем. Все, ключ создан.

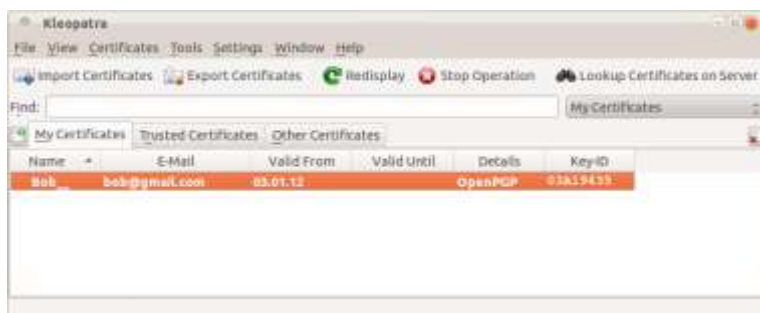


Здесь можно сохранить резервную копию ключей в файл. При этом будет создан бинарный файл с расширением pgr, например, alice.pgr.

На следующем скриншоте показано окно Kleopatra, где появился ключ Алисы (Alice):



Будем считать, что то же самое проделал и Боб:



Для генерации ключа Боба, к его имени пришлось добавить несколько символов подчеркивания, потому что программа не хотела создавать ключ для имени длиной три символа.

Обмен ключами

Алиса и Боб имеют по паре ключей. Теперь им надо обменяться открытыми ключами, но как их достать из программы? На самом деле очень просто. Выбираем нужный сертификат из списка и выбираем пункт меню *File -> Export Sertificates*. Программа предложит сохранить файл с расширением *asc*. Сохраняем его. Это текстовый файл, который будет выглядеть примерно так:

——BEGIN PGP PUBLIC KEY BLOCK——

Version: GnuPG v2.0.17 (MingW32)

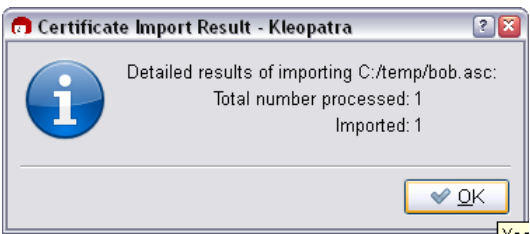
```
mQENBE8CuKABCADXPmSlhNvWqOwWouF1BS98SB5Ye
O16C2vVLoR1FokkDrG5j6U
hjGZ2V00jPzKGuGdvGz0LvU0zJpFfUk5t4zoVo4Hb3D6WAGR+vhr
TRV7P/QJe396
G0JEW9RILxfriUdZ6sJ599+pocHPVUAKEPTIxDdU7ZRMpmszwlu
PWfXdxgfwxQ2
IOxIe2vdN2HAIOBG9Ce1HR6tf0amkgUINzU+JdXSLhKpy4zWLo2
giyonw4TBuAP
/nVhXJbTXWOyZaQw+fbHlFy2LiPCRfQUaksNOqwMdXs+LjhYxh
HOojSMuFw3U1xL
kp+mfB531CJFxZxFDKXz2EDN7IDB1HvdHQnfABEBAAG0GEFsa
WNlIDxbGJjZUBn
b29nbGUuY29tPokBOAQTAAIAIgUCTwK4oAIbDwYLCQgHAwIG
FQgCCQoLBBYCAwEC
HgECF4AACgkQXSXOOeGHS44XplAgAngT686Et08AdejMua2AgdI
oSznKEbBT1cD3t
Hagr7B+bJw6p11aFwP+5hXivqIOrJv4ASUuJ+LfMxP9tR2aNNH5vrjq
WTacQabQg
JAAC3L4PFyQDaYBbb98CWfQJmXPUAehIPdmmy5dfoNI3CHwuP
M9GMiZlVbHdxI4n
OuvvG5/vzYLPDi5gjFdRQs6vg9iT1JoLHDXFKGCOxOTsUrhcZWX
go++T0j0PIemi
dE695QhV79vqxpSufKDtZSsoT7UfPTCtDmBmia1sSbcgS56XM1+Ij
w90PyFJ0rG7
vmIAuQ5cHJ/p40APHOpR4STGl+NLymMnivzcutsYyhNaoboUtA==
=ctUV
```

——END PGP PUBLIC KEY BLOCK——

Теперь Алиса должна переслать свой файл asc Бобу, а Боб, соответственно, свой файл Алисе. Или можно просто послать текст ключа, добавив его в текст электронной почты, при этом сохранив шапку ——BEGIN PGP PUBLIC KEY BLOCK—— и подвал ——END PGP PUBLIC KEY BLOCK——

Теперь Алиса должна добавить открытый ключ Боба в список сертификатов. Этот процесс тоже очень простой. В программе Kleopatra Алиса выбирает пункт меню *File -> Import Certificates...* и

выбирает файл asc, который ей послал Боб. После этого будет выведено сообщение о том, что ключ импортировался:



Теперь в списке сертификатов Алисы появился открытый ключ Боба:



Можно на него дважды щелкнуть и посмотреть информацию:

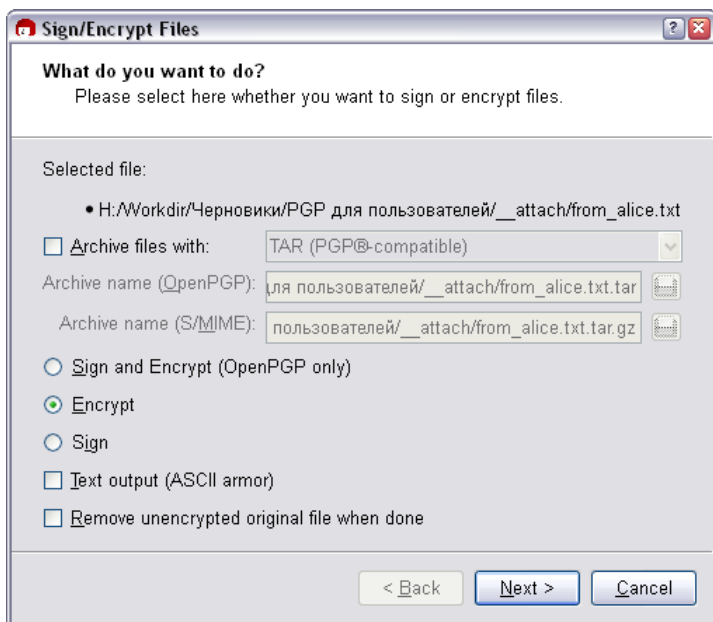


Если бы Боб послал Алисе свой ключ не в виде файла, а в виде текста, то Алиса должна была бы скопировать его в буфер обмена (вместе с шапкой и подвалом), а затем нажать правую кнопку мыши на значок Kleopatra в трее и выбрать пункт меню *Clipboard -> Sertificate Import*, ключ точно так же был бы импортирован.

Шифрование

А теперь самое интересное. Алиса хочет отправить Бобу зашифрованное сообщение. Для этого она должна проделать следующие операции. Есть два варианта отправить секретное сообщение: зашифровать файл или зашифровать содержимое буфера обмена. Начнем с шифрования файлов, это более безопасный способ.

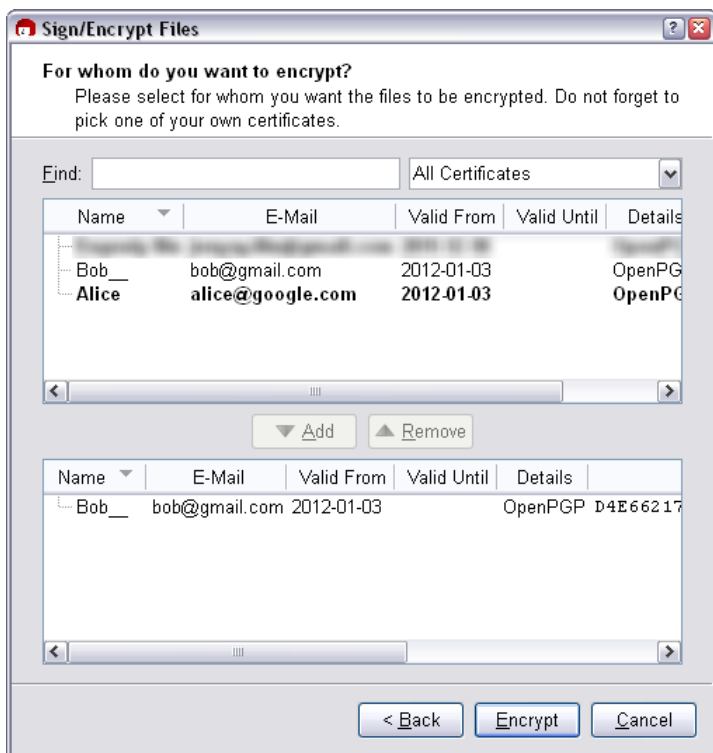
Итак, Алиса создает файл с сообщением Бобу (например, *from_alice.txt*). Затем в Kleopatra выбирает пункт меню *File -> Sign/Encrypt Files...* и выбирает файл с сообщением, после чего откроется следующее окно:



Здесь можно выбрать, что нужно сделать с файлом: зашифровать (*Encrypt*, выбрано по умолчанию), подписать файл (*Sign*), или сделать и то, и другое (*Sign and Encrypt*). Если установить галку *Text output (ASCII armor)*, то в результате шифрования будет создан не бинарный файл, а текстовый, содержимое из которого можно вставить, например, в текст письма. Если установить галку *Remove unencrypted original file when done*, то после шифрования исходный файл будет удален.

Только для шифрования нужно выбрать пункт *Encrypt*, а ставить галку *Text output (ASCII armor)* или нет — дело вкуса.

Затем нам нужно выбрать сертификаты тех, для кого предназначено сообщение. Как я уже говорил, в программе есть возможность шифровать таким образом, чтобы расшифровать сообщение могли несколько человек. Но в данном случае Алисе это не надо, она посылает сообщение только Бобу, поэтому только его сертификат и добавляет в нижний список.

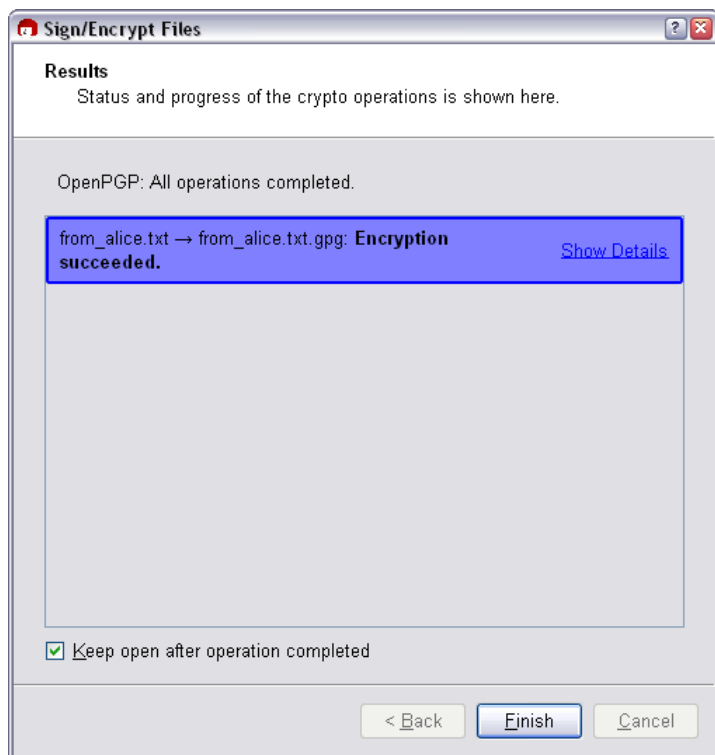


Нажимаем кнопку *Encrypt*, после чего программа может сообщить о том, что Алиса собирается зашифровать сообщение таким образом, что сама она его расшифровать уже не сможет.



Будем считать, что Алиса у нас еще в здравой памяти и расшифровывать свое сообщение ей не понадобится, поэтому нажимаем *Continue*.

После этого происходит шифрование, о его успешности говорит следующее окно:



Теперь рядом с файлом *from_alice.txt* создан файл *from_alice.txt.gpg*, если до этого не была установлена галка *Text output (ASCII armor)*, или файл *from_alice.txt.asc*, если эта галка установлена была.

Файл asc напоминает файл с открытым ключом:

——BEGIN PGP MESSAGE——

Version: GnuPG v2.0.17 (MingW32)

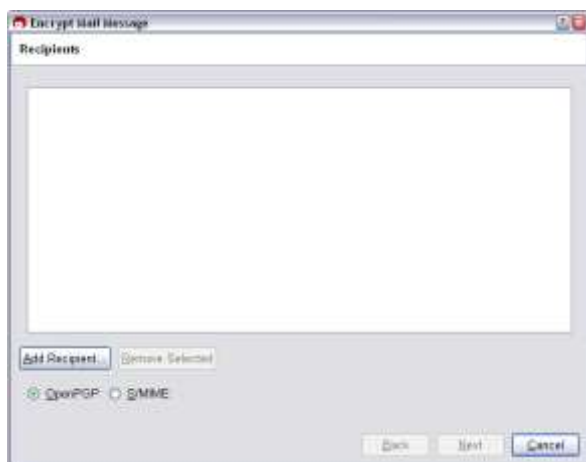
hQEMA8eYR6ADoZQ1AQf+OOyS4w/ri3aEFF1RbK5ph8ho8Z
bE8JGyUuXSPt+jR6Py
oJvxsZSfYjKf1vCjNFyn+TmHO0GxsHdFnPNfcegDk6LD4dNQok/

M1pqacku2Ijb
YZREYIO5Oa1whfATYE+Qs3Rx9fIsPrLn/1JhEv0IFAUJU+P2+YoK
ZFp3KzAk28lb
CzqqLhS3eSKp/pL06X4ZFcGsaivWn8DsQDcpFzBN8XCyYuHqKUh
9GhmIliBf0NcY
oNmlo7rNiHhPVUJLVONbx6YpZazQ/cR0nXn0IKGltYS3Y5BC6ug
UaRSa70huGxVd
160BvAr3Qtdzx5z34Ec/clJ/IfwBWKEPifN0h8KledKCAtpJnvifTtZT
HM//dsKT
GG3zQLLK4HQJ/tBUiTeLYpcdrSqv6RfQCyaxTJMQUUgCZ7yZ6Fq
fZG0dLhbhICEY
sCdrnS2aIhY0YQn3fZnqmdM2N0F/O4VCqOvKrtz16RHjB2WxXSTt
JJwnOgI9tzD
tVTjDCuuK0inWMuyn+SSvurC3Q==
=XNqU

——END PGP MESSAGE——

Это и есть зашифрованное сообщение Алисы. Теперь она может переслать файл *from_alice.txt.gpg* или *from_alice.txt.asc* Бобу, а может вставить полученное сообщение в текст письма.

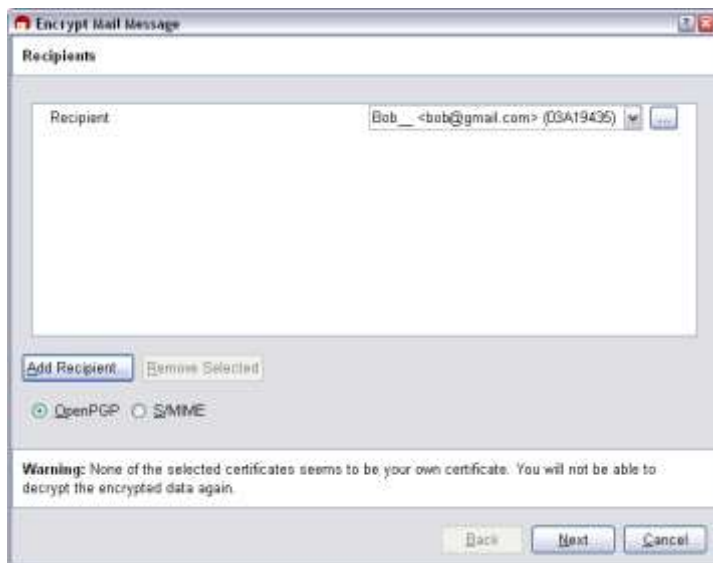
Кроме того, в программе Kleopatra есть возможность шифровать содержимое буфера обмена, для этого Алиса где-нибудь набирает текст, копирует его в буфер обмена и нажимает правую кнопку мыши на иконке в трее. В выпадающем меню выбирает пункт *Clipboard -> Encrypt*. Откроется окно



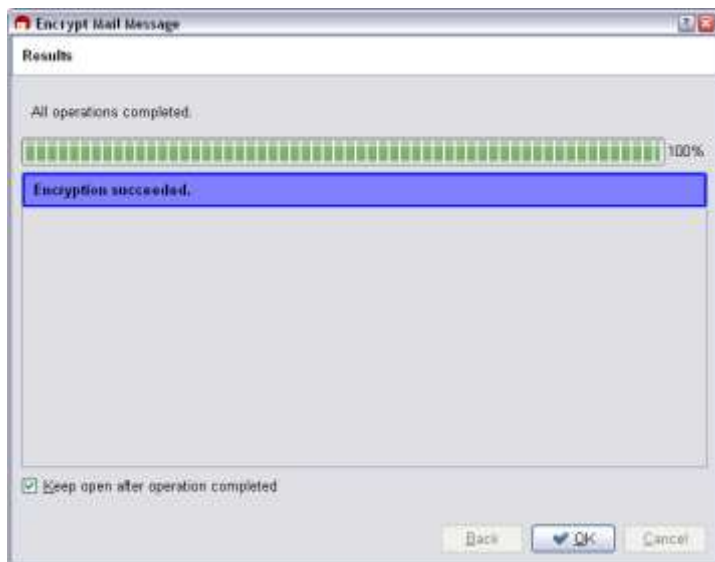
Нажимает кнопку *Add Recipient...* (добавить получателя), идет на вкладку *Other Certificates* и выбирает сертификат Боба.



Внизу появляется предупреждение, что так как Алиса не выбрала свой сертификат, то сама она уже не сможет расшифровать свое сообщение.



Но ей это и не надо, она нажимает кнопку *Next*, после этого происходит шифрование, а затем откроется следующее окно, которое сообщает о том, что шифрование успешно завершено.



В буфере обмена теперь будет содержаться зашифрованный текст, который можно отправить по почте

——BEGIN PGP MESSAGE——

Version: GnuPG v2.0.17 (MingW32)

hQEMA8eYR6ADoZQ1AQf+IEMRfO49t6MvExiyeEtN8Gr7jh
 YCfzAltDvCbqQuanuy
 6SoSq2FR4mSytuOKWLyXvRys+pNDpZONbXzbGVtTJbTQbj8wuB
 SAh4nSKFpyWeTy
 4CHO5zkmztYJ10aH84njEDuXjqdlKChsaoJwQnK1yQtQ+8TMBjRH
 vfMvEwHTRglz
 Lgesd5tvXK0hn2jpO1ujZeqOurh3RExkFw0Mb8D6ZaR3bmo5QtFLr
 YxdYEA+Itmd
 SSKoQqmslhY2vR8uSFk6Jeo7BpglT/g+82J95qTfdN8RyC/5yZa21Ak
 Ge46gw+Io
 cDSIIItsKgcVqhHlUCf2p17OWbjMDHOCAaaCJQcw7mdKAAXbJIt8
 dF2UzOsUOmUCT
 FQ/kjovcTcG98fwzZqlVfewWYNOpFVE9RaOIhuL/Q4+BcF6xXEgI
 Bj2RxxdMLMB0
 O7Pjize5B+YaDeu69795xhzNLP2vnBLDyZcMT7jAzhWILWWTua
 WZeYT+mnnMD/8

DN6gBhFyfweJxP/1x48jY6o=
=GR54

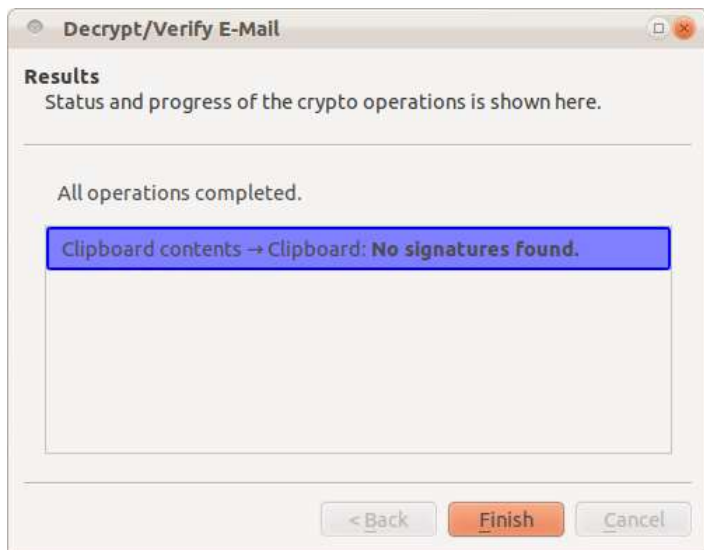
—END PGP MESSAGE—

Несмотря на то, что шифрование буфера обмена кажется более удобным, это потенциальная дыра, потому что буфер обмена легко отслеживается с помощью троянов или даже с помощью Punto Switcher.

Расшифровка

Боб получил письмо от Алисы с шифровкой. Рассмотрим шаги, которые он должен сделать, чтобы ее расшифровать. Для расшифровки сообщений Бобу не нужен никакой ключ, кроме собственного закрытого ключа.

Если Алиса послала сообщение в виде прикрепленного файла gpg или asc, то Боб запускает Kleopatra, выбирает пункт меню *File - > Decrypt/Verify Files...* и выбирает полученный от Алисы файл. Если все происходит удачно, то появится сообщение об успешной расшифровке:



А рядом с файлом from_alice.txt.gpg или from_alice.txt.asc будет создан файл from_alice.txt, который содержит исходный текст, который написала Алиса.

Если же Алиса послала Бобу сообщение в виде текста письма, то Боб копирует его в буфер обмена, затем нажимает правую кнопку мыши на иконке Kleopatra в трее и в выпадающем меню выбирает *Clipboard -> Decrypt/Verify*. Если расшифровка проходит удачно, то после этого в буфере обмена будет храниться расшифрованное сообщение.

Таким образом можно обмениваться зашифрованными письмами и тешить себя надеждой, что ваши послания не будут никем прочитаны, кроме получателя. Разумеется, в реальности все сложнее, и 100%-ой гарантии того, что вас не читают никто не даст. Ко многим почтовикам и программам мгновенного обмена сообщений существуют плагины, которые облегчают шифрование/расшифровку сообщений, делая это на лету, прозрачно для пользователя.

Контрольные вопросы:

1. Какие основные функции выполняет программа GnuPG/PGP?
2. Что такое ключ?
3. Поясните назначение открытого и закрытого ключей.
4. Поясните принцип шифрования и дешифрования информации с помощью GnuPG/PGP.
5. Как можно обмениваться открытыми ключами со своими корреспондентами?
6. Назовите основные способы шифрования сообщений при помощи GnuPG/PGP.
7. Кто может расшифровать сообщение, зашифрованное открытым ключом?

ОБЩИЕ ПРАВИЛА БЕЗОПАСНОСТИ ПРИ ВЫПОЛНЕНИИ ЛАБОРАТОРНЫХ РАБОТ

Перед началом работы необходимо:

- слегка влажной тряпочкой удалить пыль с экрана и поверхности монитора, принтера и процессорного блока, при этом выключатель и разъем электропитания должны быть выключены;
- произвести внешний осмотр устройств, шнуров питания и интерфейса.

При обнаружении механических повреждений корпусов, экрана монитора, шнуров питания и интерфейса пользователь должен незамедлительно обратиться в ремонтную службу.

Включение питания техники производится соответствующим переключателем на корпусе при подключенном к сети разьеме питания. Запрещается подключать или отключать разъем питания к сети при включенном переключателе на корпусе устройства. При выключении питания устройства его повторное включение производится не ранее чем через 1 мин.

Подключение и выключение интерфейсных кабелей между устройствами производится при выключенном питании обеих устройств.

По окончании работы следует отключить монитор, блок питания компьютера, принтер и др. оргтехнику, отключить разъем питания.

Запрещается эксплуатировать технику:

- признанную специалистом неисправной или непригодной к эксплуатации;
- не подключенную к контуру заземления;
- имеющую механические повреждения корпусов;
- имеющую нарушение пломб;
- имеющую неисправное электропитание.

В кабинете информатики строго запрещается:

- Находиться в верхней одежде и грязной обуви.

- Класть одежду и сумки на столы.
- Находиться с едой и напитками
- Трогать разъемы соединительных кабелей.
- Удалять и перемещать чужие файлы.
- Приносить и запускать компьютерные игры.
- Работать на ЭВМ грязными или мокрыми руками.
- Прикасаться пальцами к мониторам, стучать по ним.
- Включать и выключать аппаратуру без указания преподавателя.
- Класть диски, книги, тетради на составляющие компьютера.

Литература

Основная литература

1. Адаменко М.В. *Основы классической криптологии: секреты шифров и кодов.* / М.В. Адаменко. - М.: ДМК Пресс, 2012 – 256 с.
2. Иванов М.А. *Криптографические методы защиты информации в компьютерных системах и сетях: учеб. пособие* / М.А. Иванов, И.В. Чугунков - М.: МИФИ, 2012 – 400 с.
3. Петров А.А. *Компьютерная безопасность. Криптографические методы защиты.* / А.А. Петров. - М.: ДМК Пресс, 2008 – 448 с.
4. Фороузан Б.А. *Криптография и безопасность сетей: учеб. пособие* / Б.А. Фороузан – М.: Интернет-Университет, 2010 – 784 с.


Дополнительная литература

5. *Теория чисел в криптографии: учеб. пособие* / В.А. Орлов, Н.В. Медведев, Н.А. Шимко, А.Б. Домрачева. – М.: Изд-во МГТУ им. Н.Э. Баумана, 2011 – 223с.
6. Левин М. *Криптография: руководство пользователя: учеб. пособие* / М. Левин. – М.: Познавательная книга плюс, 2001 – 320с.
7. Фомичев В.М. *Методы дискретной математики в криптологии.* – М.: ДИАЛОГ-МИФИ, 2010. – 424 с.

ПРИЛОЖЕНИЕ

ФОРМЫ ОТЧЕТОВ ПО ЛАБОРАТОРНЫМ РАБОТАМ

Титульный лист

	<p>Министерство образования и науки Российской Федерации Калужский филиал федерального государственного бюджетного образовательного учреждения высшего профессионального образования «Московский государственный технический университет имени Н.Э. Баумана» (КФ МГТУ им. Н.Э. Баумана)</p>
ФАКУЛЬТЕТ	«Электроника, информатика, управление»
КАФЕДРА	«Информационная безопасность автоматизированных систем»
О Т Ч Е Т	
ЛАБОРАТОРНАЯ РАБОТА №__	
ДИСЦИПЛИНА: "Теоретическая информатика"	
ТЕМА:	"_____"
Выполнил: студент гр. БАС-11	Ф.ИО студента_____
Проверил:	Лавина А.Б. _____
Дата сдачи (защиты) лабораторной работы: _____	
Результаты сдачи (защиты):	
Количество рейтинговых баллов	_____
Оценка	_____
	зачтено
Калуга, 201_ г.	

Отчет должен в себя включать:

1. цель работы;
2. задание с вариантом;
3. ход выполнения работы;
4. результат;
5. вывод.