ПРИДНЕСТРОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
им. Т.Г. ШЕВЧЕНКО
Физико-технический институт
Факультет информатики и вычислительной техники
Кафедра информационных технологий

Методические указания «Лабораторные работы по предмету «Защита информации»»

Тирасполь
Издательство Приднестровского Университета

2023

УДК 004.056.5
ББК 16.84

Составители:
В.С. Попукайло, канд. техн. наук доцент кафедры информационных технологий

А.В. Шмелёва, преподаватель кафедры информационных технологий

Рецензенты:
С.В. Помян, канд. пед. наук, доцент кафедры программного обеспечения вычислительной техники
С.П. Коноп, канд. Физ.-мат. наук, директор ООО «Санкор»

Защита информации: лабораторный практикум / составители: В.С. Попукайло, А.В. Шмелёва – Тирасполь: Изд-во Приднестр. ун-та, 2023. – 66 с.

Настоящие методические указания имеют целью изучение теоретических и практических основ защиты информации для реализации программ на языке высокого уровня.

Предназначены для студентов направлений 2.09.03.01 «Информатика и вычислительная техника», 2.09.03.02 «Информационные системы и технологии», 2.09.03.04 «Программная инженерия» ФТИ ПГУ, а также могут быть использованы, при соответствующем изменении объёма, и для студентов других направлений ФТИ.

УДК 004.056.5
ББК 16.84

Рекомендовано Научно-методическим советом ПГУ им. Т.Г. Шевченко

ВВЕДЕНИЕ

Данные методические указания освещают теоретические и практические основы защиты информации для реализации программ на языке высокого уровня, а также ознакомления с программным обеспечением.

На практике используют несколько групп методов защиты информации, в том числе:

– регламентация, или разработка нормативно-правовых актов и набора мер, направленных на то, чтобы побудить пользователей, взаимодействующих с базами данных, к должному поведению;

– маскировка, или преобразование данных, обычно – криптографическими способами;

– принуждение, или создание таких условий, при которых пользователь будет вынужден соблюдать правила обращения с данными – разграничение прав доступа;

– на пути предполагаемого похитителя, которое создают физическими и программными средствами.

Каждый из методов защиты информации реализуется при помощи различных категорий средств – организационные и технические.

Регламент по обеспечению информационной безопасности – внутренний документ организации, который учитывает особенности бизнес-процессов и информационной инфраструктуры, а также архитектуру системы.

Группа технических средств защиты информации совмещает аппаратные и программные средства. Основные:

– резервное копирование и удаленное хранение наиболее важных массивов данных в компьютерной системе – на регулярной основе;

– шифрование данных;

– установка программного обеспечения, которое обеспечивает защиту баз данных и другой информации от несанкционированного доступа

– создание возможности перераспределять ресурсы сети в случаях нарушения работоспособности отдельных элементов;

– обеспечение возможности использовать резервные системы электропитания;

– обеспечение безопасности от пожара или повреждения оборудования водой.

В комплекс технических мер входят и меры по обеспечению физической недоступности объектов компьютерных сетей, например, такие практические способы, как оборудование помещения камерами и сигнализацией.

ЛАБОРАТОРНАЯ РАБОТА №1

ШИФР ПРОСТОЙ ПЕРЕСТАНОВКИ

Теоретические сведения

Защита конфиденциальных сообщений в открытых сетях требует применения криптографических методов. Криптография является самым мощным в настоящее время средством защиты информации и представляет самостоятельную прикладную науку, основанную на глубоких математических знаниях.

Криптографические методы позволяют обеспечить конфиденциальность данных, устанавливать подлинность отправителя и контролировать целостность передаваемых сообщений.

Защита информации методами криптографического преобразования шифрования заключается в изменении составных частей (слов, букв, слогов, цифр) с помощью специальных алгоритмов или аппаратных решений.

Шифр – совокупность алгоритмов или отображений открытой информации, представленной в формализованном виде, в недоступный для восприятия, который зависит от внешнего параметра (ключа). **Шифрование** – процесс приведение текста к неявному виду. **Ключ** – неизвестный параметр шифра, позволяющий выбрать конкретное преобразование, должен передаваться способом, исключающим его перехват. Восстановление сообщения или ключа из зашифрованного текста называется дешифрование.

Преобразование информации служит средством, дающим возможность скрыть её смысл от большинства неквалифицированных нарушителей.

Множество методов защитных преобразований можно разбить на группы: перестановки, замены, аддитивные и комбинированные методы.

Методы перестановки и подстановки обычно характеризуются короткой длиной ключа, а надежность их защиты определяется сложностью алгоритмов преобразования.

Все современные методы шифрования делятся на одноключевые или симметричные и двухключевые или асимметричные.

В симметричных криптографических системах ключи, используемые на передающей и приемной сторонах, полностью идентичны. Такой ключ несет в себе всю информацию о засекреченном сообщении и поэтому не должен быть известен никому, кроме двух участвующих в разговоре сторон. Противник, наблюдая зашифрованное сообщение (шифротекст), не может прочитать сообщение. Отсюда возникает задача для противника – получение открытого сообщения по наблюдаемому шифротексту, а именно задача сводится к раскрытию секретного ключа шифрования и дешифрования.

Действия, предпринимаемые противником с целью раскрытия секретного ключа, называется **атакой**.

Секретный канал, используемый для передачи секретного ключа от отправителя к получателю, должен исключать возможность утечки информации.

В асимметричных криптографических системах для шифрования и дешифрования применяются различные ключи. Для шифрования информации, предназначенной конкретному получателю, используют уникальный открытый ключ получателя-адресата. Соответственно для дешифрования получатель использует парный секретный ключ. Для передачи открытого ключа от получателя к отправителю секретный канал не нужен. Вместо секретного канала используется аутентичный канал, гарантирующий подлинность источника передаваемой информации (открытого ключа отправителя). Аутентичный канал является открытым и доступен противнику.

Данный метод относятся к симметричному шифрованию. В симметричных алгоритмах для зашифровки и расшифровки сообщения используется один и тот же ключ.

Шифр, преобразования которого изменяют только порядок следования символов исходного текста, но не изменяют их самих, называется шифром перестановки. Алгоритм шифрования состоит из двух этапов:

– открытый текст делится на блоки равного размера, если размер текста не кратен размеру блока, в конец сообщения дописывают пробелы;

– в каждом блоке выполняется перестановка структурных элементов, правило выполнения перестановки и является ключом к шифру.

Рассмотрим преобразование с использованием шифра перестановки, предназначенное для зашифровки сообщения длиной *n* символов:

$$\begin{pmatrix} 1 & 2 & \dots & n \\ i_1 & i_2 & \dots & i_n \end{pmatrix}$$

где i_1 номер шифротекста, на которое попадает первая буква исходного сообщения при выбранном преобразовании, i_2 номер для второй буквы и т.д.

Например, исходное сообщение: «МАМА МЫЛА РАМУ», перестановка:

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 4 & 1 & 3 \end{pmatrix}$$

Размер блока: 4 символа (в перестановке участвуют 4 последовательных символа).

Длина текста: 14 символов, учитывая пробелы, размер сообщения не является кратным размеру блока. Разбиение открытого текста на блоки (с добавлением пробелов, для наглядности пробелы изображены символами подчеркивания):

М А М А	_ М Ы Л	А _ Р А	М У _ _
(блок 1)	(блок 2)	(блок 3)	(блок 4)

Перестановка в блоке 1:

Открытый текст	Шифротекст
М А М А	А А М М
1 2 3 4	2 4 1 3

Перестановка всех блоков:

А А М М	М Л _ Ы	_ А А Р	У _ М _
(блок 1)	(блок 2)	(блок 3)	(блок 4)

Шифротекст: «ААММММЛ_Ы_ААР_У_М_»

Для расшифровки сообщения необходимо применить обратную перестановку.

Прямая перестановка: 1→2, 2→4, 3→1, 4→3

Обратная перестановка: 1→3, 2→1, 3→4, 4→2

Правило выполнения перестановки и является ключом к шифру. Данный вариант шифра не является особенно стойким, поскольку размер ключа существенно меньше размера текста. Кроме того, в зашифрованном тексте сохраняются статистические зависимости между символами. Например, частота появлений отдельных символов в открытом тексте и шифртексте остается постоянной, что облегчает подбор ключа.

Задание на лабораторную работу

Реализовать алгоритм шифрования подстановкой степени *n*, среда разработки и язык программирования используется на усмотрение студента.

При выполнении лабораторной работы необходимо предусмотреть выведение на экран: исходного текста, ключа, зашифрованного и расшифрованного текстов.

Варианты заданий:

1. Баран карабкался с карабином
2. Таракан попал в капкан
3. Барабанщик бил в ящик
4. Колокол из Волоколамска
5. Полотенце попало в болото
6. Колобок полотенце уволок
7. Херес попал на перевязь
8. Мел емеля мел в мельнице
9. На лапу упала капля пакли
10. Не пей пену у репейника

Вариант 1

Вместо	Подставить
1	4
2	3
3	5
4	1
5	2

Вариант 2

Вместо	Подставить
1	4
2	1
3	3
4	2

Вариант 3

Вместо	Подставить
1	2
2	4
3	3
4	1

Вариант 4

Вместо	Подставить
1	4
2	3
3	2
4	1

Вариант 5

Вместо	Подставить
1	3
2	5
3	1
4	2
5	4

Вариант 6

Вместо	Подставить
1	4
2	1
3	2
4	5
5	3

Вариант 7

Вместо	Подставить
1	1
2	4
3	2
4	3

Вариант 8

Вместо	Подставить
1	3
2	4
3	1
4	2

Вариант 9

Вместо	Подставить
1	1
2	4
3	5
4	3
5	2

Вариант 10

Вместо	Подставить
1	3
2	5
3	1
4	4
5	2

Контрольные вопросы

1. Чем шифр отличается от шифрования?
2. Что такое ключ?
3. Симметричные криптографические алгоритмы – это?
4. Асимметричные криптографические алгоритмы – это?
5. Перечислите группы криптографических алгоритмов.
6. Какие шифры называются шифрами простой замены.
7. Что влияет на сложность алгоритма перестановки?

ЛАБОРАТОРНАЯ РАБОТА №2

ШИФР ЦЕЗАРЯ

Теоретические сведения

Большое влияние на развитие криптографии оказали появившиеся в середине *XX* века работы американского математика Клода Шеннона. В этих работах были заложены основы теории информации, а также был разработан математический аппарат для исследований во многих областях науки, связанных с информацией.

До появления компьютеров криптография состояла из символьных алгоритмов. Криптографические алгоритмы либо заменяли одни символы другими, либо переставляли символы. Лучшие алгоритмы делали и то и другое, причем многократно.

Подстановочным (*substitution cipher*) называется шифр, который каждый символ открытого текста заменяет другим символом в шифротексте. Получатель выполняет обратную подстановку в шифротексте, восстанавливая открытый текст. В классической криптографии существуют четыре типа подстановочных шифров.

Простой подстановочный шифр (*simple substitution cipher*), или **моноалфавитный** шифр (*monoalphabetic cipher*), – это шифр, который заменяет каждый символ открытого текста соответствующим символом шифротекста. Примером простых подстановочных шифров являются криптограммы в газетах.

Омофонический подстановочный шифр (*homophonic substitution cipher*) похож на простую подстановочную криптосистему, за исключением того, что один символ открытого текста заменяется несколькими символами шифротекста. Например, букве A может соответствовать набор чисел 5, 13, 25 или 56, букве B — 7, 19, 31 или 42 и т.д.

Полиграммный подстановочный шифр (*polygram substitution cipher*) – это шифр, который заменяет одни блоки символов другими. Например, символам ABA могут соответствовать символы *RTQ* и т.д.

Полиалфавитный подстановочный шифр (*polyalphabetic substitution cipher*) состоит из нескольких простых подстановочных шифров. Например, можно использовать пять разных простых подстановочных фильтров так, что каждый символ открытого текста заменяется с использованием одного конкретного шифра.

Шифр Цезаря, также известный как шифр сдвига (замены), код Цезаря или сдвиг Цезаря – один из самых простых и наиболее широко известных методов шифрования. Шифр Цезаря – это вид шифра подстановки, в котором каждый символ в открытом тексте заменяется символом, находящимся на некотором постоянном числе позиций левее или правее него в алфавите.

Прежде всего, выбирается нормативный алфавит, т.е. набор символов, которые будут использоваться при составлении сообщений, требующих зашифровки. Допустим, это будут прописные буквы русского алфавита (исключая буквы “Ё” и “Ъ”) и пробел. Таким образом, наш нормативный алфавит состоит из 32 символов. Затем выбирается алфавит шифрования и устанавливается взаимно однозначное соответствие между символами нормативного алфавита и символами алфавита шифрования.

Алфавит шифрования может состоять из произвольных символов, в том числе и из символов нормативного алфавита. Чтобы зашифровать исходное сообщение, каждый символ открытого текста заменяется соответствующим ему символом алфавита шифрования.

Например, в шифре со сдвигом вправо на 1, А была бы заменена на Б, Б станет Г, и так далее (рисунок 1.1).

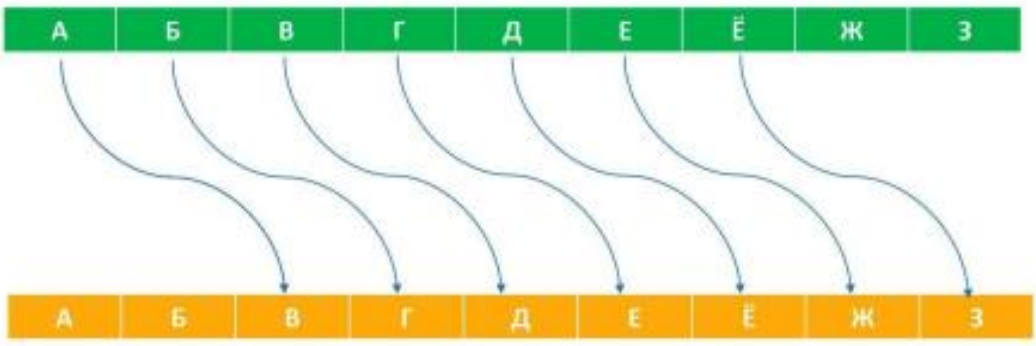

Рисунок 1.1 – Шифр Цезаря

Шифр Цезаря представляет собой простой подстановочный фильтр. На самом деле этот алгоритм еще проще, чем подстановочный, потому что алфавит шифротекста представляет собой результат смещения алфавита открытого текста, а не его случайную перестановку.

Задание на лабораторную работу.

Реализовать (среда разработки и язык программирования используется на усмотрение студента) алгоритм Цезаря степени *n*, при выполнении лабораторной работы необходимо предусмотреть вывод на экран: исходного текста и зашифрованного текста. Расшифруйте текст не зная значение сдвига:

Варианты заданий:

1. есффхгрсеозрлз фссдьзрлв лол нобъг лк кгылчусегррсёс хзнфхг ргкюегзхфв жзылчусегрлз

2. ё офмуцтжфдшмм зфиёсмщ ёфирис мхутпалтёдпмха зёд ёмзд ьмшфтё лдрисд м уифихцдстёод

3. рглдсозз жузерлп л угфтусфхугриррюп тулпзусп ылчуг кгпзрю веовзхфв ылчу щзкгув

4. рцохшфицёъое фйуё он чшёцкпюоы уёщр кл очшфцое уёчэошбзёкш укчрфсврф шбчеэ скш

5. щъзшмсарс р цлрх рп цщхцйхгэ щчцщциций пзбръг рхьцшфзюрр еъц щруцйгм фмъцлг, ъц мщъд цэшзхз лцтыфмхъз

6. рёкп кй усрургрд йвыкфэ кпцртовшкк яфр уфжевпретвцкб йвмнащвнуб д урмтэфкк уворер цвмфв пвнкщкб ужмтжфпрл кпцртовшкк

7. зт утгёписмг отруавцифтё офмуцтжфдшмг хтхцтгпд мхопвымципаст мл хмрётпасящ дпжтфмцртё

8. чщшв юоъцё ькнёце чфчшфош з шфт эшф жщрзб ёсъёзошё нёткуедшче жщрзёто шфиф мк ёсъёзошё уф чф чйзоифт

9. лштп цчхщпифпс ъофжт хз пшцхтгоълуху япычл щх хф ухнлщ чжшя-пычхижщг щлсшщ цъщму цлчлзхчж стеюлр

10. втюъе щоъоыьйчшлфт илхизьыи ыйцецт щъшыьецт т лоъшиьчш ыйцецт нъолчтцт втюъйцт

Контрольные вопросы

1. Какие типы криптографических алгоритмов замены существуют?
2. Чем отличаются полиграммный и полиалфавитный алгоритмы?
3. Насколько надёжный шифр Цезаря?

ЛАБОРАТОРНАЯ РАБОТА №3

КРИПТОАНАЛИЗ ШИФРА ПРОСТОЙ ЗАМЕНЫ

Теоретические сведения

Криптоанализ – наука о методах дешифровки информации без предназначенного для этого ключа, а также сам процесс такой дешифровки.

В большинстве случаев под криптоанализом понимается выяснение ключа; криптоанализ включает также методы выявления уязвимости криптографических алгоритмов или протоколов.

Попытку раскрытия конкретного шифра с применением методов криптоанализа называют криптографической атакой на этот шифр. Криптографическую атаку, в ходе которой раскрыть шифр удалось, называют «взломом» или «вскрытием» шифра.

Первоначально методы криптоанализа основывались на лингвистических закономерностях естественного текста и реализовывались с использованием только карандаша и бумаги. Со временем в криптоанализе нарастает роль чисто математических методов, для реализации которых используются специализированные криптоаналитические компьютеры.

Частотный анализ – основной инструмент для взлома большинства классических шифров перестановки или замены. Данный метод основывается на предположении о существовании нетривиального статистического распределения символов, а также их последовательностей одновременно и в открытом тексте, и в шифротексте. Данное распределение будет сохраняться с точностью до замены символов в процессе шифрования, а также в процессе дешифрования.

Стоит отметить, что при условии достаточно большой длины шифрованного сообщения моноалфавитные шифры легко поддаются частотному анализу: если частота появления буквы в языке и частота появления некоторого присутствующего в шифротексте символа приблизительно равны, то в этом

случае с большой долей вероятности можно предположить, что данный символ и будет этой самой буквой.

Простым примером частотного анализа служит подсчёт количества встречающихся символов, затем следуют представить окончательный ответ в процентах. Далее полученные процентные значения сравниваются с таблицей вероятностного распределения букв для предполагаемого языка оригинала.

Разберем предлагаемый метод криптоанализа на примере. Рассмотрим шифротекст (рисунок 1.2):

1	С	З	Я	И	Г	Н	Д	Л	В	С	З	Л	П	В	У
2	И	У	Л	З	Х	Г	З	Т	Э	Ф	Б	Д	А	Т	И
3	Г	К	П	З	Т	С	И	П	З	Н	Ф	Э	Л	К	И
4	Э	М	У	О	У	Д	Т	Ж	Г	З	Д	И	М	И	С
5	Э	М	И	Г	С	З	Ф	П	Л	З	М	Д	Ф	Г	Д
6	С	И	П	Э	Л	Э	Ц	З	Ю	В	С	З	Е	З	Л
7	З	Г	Э	Г	И	У	М	В	П	Т	Э	Г	Д	Э	З
8	У	М	И	Г	З	Ф	Т	Э	Г	З					

Рисунок 1.2 Шифротекст

При раскрытии будем иметь в виду следующие отличительные признаки шифра однобуквенной замены:

– в шифротексте встречается большое число сплошных повторений;

– диаграмма встречаемости шифробозначений – рельефная и соответствует диаграмме открытого текста.

Проведем предварительный анализ шифротекста.

Подчеркнем в шифротексте наиболее характерные неслучайные повторения и представим на рисунке 1.3.

Повторение	Количество
ВСЗ	2
МИГ	2
ТЭГ	2
ИУ	2

Рисунок 1.3 – Характерные неслучайные повторения

Прежде всего необходимо отметить, что в тексте явно имеется значимо завышенное число повторений. Наличие большого числа повторений свидетельствует о том, что для зашифрования использовался шифр простой замены.

Для раскрытия шифра простой замены производится разбиение шифротекста на шифробозначения (ШО), выполним маркировку шифробозначений и маркировку пар шифробозначений.

В результате получим следующий шифротекст (рисунок 1.4):

1	С	З	Я	И	Г	Н	Д	Л	В	С	З	Л	П	В	У
2	И	У	Л	З	Х	Г	З	Т	Э	Ф	Б	Д	А	Т	И
3	Г	К	П	З	Т	С	И	П	З	Н	Ф	Э	Л	К	И
4	Э	М	У	О	У	Д	Т	Ж	Г	З	Д	И	М	И	С
5	Э	М	И	Г	С	З	Ф	П	Л	З	М	Д	Ф	Г	Д
6	С	И	П	Э	Л	Э	Ц	З	Ю	В	С	З	Е	З	Л
7	З	Г	Э	Г	И	У	М	В	П	Т	Э	Г	Д	Э	З
8	У	М	И	Г	З	Ф	Т	Э	Г	З					

Рисунок 1.4 – Маркированный шифротекст

Подсчитаем частоты встречаемости шифробозначений и запишем их в порядке убывания частот:

З-16 С-7 В-4 Ж-1

Г-11 М-6 К-2 0-1

И-11 П-6 Н-2 Х-1

Э-10 Т-6 А-1 Ц-1

Д-7 У-6 Б-1 Ю-1

Л-7 Ф-5 Е-1 Я-1

Подсчитаем частоты встречаемости маркированных пар шифробозначений:

– биграммы ШО с частотой встречаемости 4: ГЗ, ИГ, СЗ;

– биграммы ШО с частотой встречаемости 3: ЛЗ, МИ, ТЭ, ЭГ;

– биграммы ШО с частотой встречаемости 2: ВС, ГД, ЗЛ, ЗТ, ЗФ, ИП, ИУ, ПЗ, СИ, УМ, ЭЛ, ЭМ.

Далее необходимо разбить все шифробозначения на гласные и согласные с учетом основных свойств открытого текста – сочетаемости и чередования, повторяемости букв в русском языке. Работа начинается с самых частых шифробозначениями. При этом самые частые надо раскрашивать, чтобы наглядно видеть их взаимосвязи.

Рассмотрим самой частое шифробозначение – «З». По шифротексту оно распределено равномерно, в отдельных местах чередуется через одно, два или три шифробозначение.

В маркировке пар шифробозначений – в сочетаниях, оно встречается чаще всех других ШО. Причем сочетается как с частыми шифробозначениями («Г», «Л», «С», «Т»), так и с ШО средней и низкой повторяемости («М», «Ф», …, «Ю», «Я»). Все это дает основание полагать, что за этим ШО скрывается гласная буква (причем, скорее всего, буква «О»).

По свойствам открытого текста чаще встречаются сочетания гласных и согласных букв, чем согласных с гласными или гласных с гласными. Значит можно предположить, что все шифробозначения, которые часто встречаются с ШО «З», скорее всего, скрывают согласные буквы. Поэтому шифробозначения «Г», «С», «Л», «Т», «Ф», «П», часто сочетающиеся с ШО «З», будем считать согласными буквами (рисунок 1.3). Дальнейший анализ подтвердит эти предположения.

Разнесем полученные результаты по шифротексту, как на рисунке 1.5.

1	С	З	Я	И	Г	Н	Д	Л	В	С	З	Л	П	В	У
2	И	У	Л	З	Х	Г	З	Т	Э	Ф	Б	Д	А	Т	И
3	Г	К	П	З	Т	С	И	П	З	Н	Ф	Э	Л	К	И
4	Э	М	У	О	У	Д	Т	Ж	Г	З	Д	И	М	И	С
5	Э	М	И	Г	С	З	Ф	П	Л	З	М	Д	Ф	Г	Д
6	С	И	П	Э	Л	Э	Ц	З	Ю	В	С	З	Е	З	Л
7	З	Г	Э	Г	И	У	М	В	П	Т	Э	Г	Д	Э	З
8	У	М	И	Г	З	Ф	Т	Э	Г	З					

Рисунок 1.5 – Размеченный шифротекст

Но если «Г» – согласная буква, то «И» – гласная, т.к. эти ШО сочетаются между собой 4 раза (рисунок 1.3). Далее ШО «Э» хорошо сочетается с «Т» и «Г». Значит, видимо, что «М» и «У» – согласные буквы.

Из группы шифробозначений высокой повторяемости неопределенным осталось ШО «Д». Изучив сочетаемость этого ШО в шифртексте, можно сделать вывод, что оно чаще встречается с согласными буквами и, кроме того, по тексту оно чередуется как гласная буква. Скорее всего, за этим шифробозначением скрывается гласная буква.

В пользу этого предположения говорит также то, что в группе ШО высокой повторяемости должно быть четыре гласные буквы, а пока выявлено только три. Значит «Д» – гласная.

Продолжая процесс, определяем, что «В», «Ж», «О» – гласные буквы, а «Е», «К» «Н», «Ц», «Ю», «Я» – согласные.

Окончательные результаты разбиения шифробозначений на гласные и согласные приведены в рисунке 1.6.

Гласная/Согласная	-	⌣	-	-	-	⌣	⌣	⌣	⌣	⌣	⌣	⌣
Шифробозначение	З	Г	И	Э	Д	Л	С	М	П	Т	У	Ф
Частота	16	11	11	10	7	7	7	6	6	6	6	5

1	С	З	Я	И	Г	Н	Д	Л	В	С	З	Л	П	В	У
2	И	У	Л	З	Х	Г	З	Т	Э	Ф	Б	Д	А	Т	И
3	Г	К	П	З	Т	С	И	П	З	Н	Ф	Э	Л	К	И
4	Э	М	У	О	У	Д	Т	Ж	Г	З	Д	И	М	И	С
5	Э	М	И	Г	С	З	Ф	П	Л	З	М	Д	Ф	Г	Д
6	С	И	П	Э	Л	Э	Ц	З	Ю	В	С	З	Е	З	Л
7	З	Г	Э	Г	И	У	М	В	П	Т	Э	Г	Д	Э	З
8	У	М	И	Г	З	Ф	Т	Э	Г	З					

Рисунок 1.6 – Шифротекст с разделением на гласные и согласные

При этом не определены самые редкие ШО: «А, «Б», «Х». Можно попробовать определить их, но при этом велика вероятность ошибки.

Кроме того, так как они встречаются редко, то их определение не имеет существенной роли для раскрытия шифра простой замены.

Следующий этап – соотнесение ШО их буквам открытого текста:

– ШО «З». Исходя из частоты встречаемости и очень высокой сочетаемости с другими ШО логично предположить, что за самым частым ШО «З» скрывается буква О».

– ШО «Г». Оно также очень часто участвует в сочетаниях – «ГЗ», «ИГ», «ЭГ», «ГД» и т.д. По свойствам открытого текста, видимо, это буква «Н».

– ШО «Д». Оно дважды сочетается с другими гласными буквами («З», «Д», «И», и «Д», «Э»), скорее всего, оно скрывает букву «И».

Среди частых пар ШО только одна типа «согласная несогласная» – это пара «УМ». Кроме того, в шифртексте встречается и одна пара «МУ». Исходя из свойств русского текста, можно предположить, что за парой «МУ» скрывается сочетание «СТ». Поэтому ШО «У» соответствует «С», а ШО «М» – «Т».

По свойствам русского открытого текста среди самых частых букв должно быть 4 гласных: «О», «И», «Е», «А». Мы соотнесли ШО для двух из них («З» – «О», «Д» – «И») и у нас осталось еще два частых ШО «И» и «Э», которые скрывают гласные буквы. Значит ШО «И» и «Э» скрывают буквы «А» и «Е». Не совсем ясно, как их правильно сопоставить. Здесь необходимо опробовать оба варианта:

«И» → «А», «Э» → «Е»;

«И» → «Е», «Э» → «А».

Путем опробования несложно установить, что верен первый вариант

«И» → «А», «Э» → «Е».

При этом в конце текста, в седьмой и восьмой строках (рисунок 1.7) получилась следующая группа букв «ЕНИЕОСТАНО..ЕНО». Так как подряд следуют три гласные буквы «ИБО», то логично предположить, что это конец одного слова и начало другого. Получаем: «ЕНИЕ ОСТАНО..ЕНО». Здесь нетрудно угадать слово «ОСТАНОВЛЕНО».

После чего дальнейшее раскрытие не представляет большой сложности. В пятой строке читается «ТАНКОВ ПРОТИВНИКА» и так далее. В итоге получаем следующий текст, приведенный на рисунке 1.7.

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

Рисунок 1.7 – Дешифрованный текст

По результатам раскрытия шифра можно составить следующую таблицу алфавита шифрования (рисунок 1.8).

А	Б	В	Г	Д	Е	Ж	З	И	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ы	Ь	Э	Ю	Я
И	Е	Ф	К	Н	Э	Ю	Р	Д	С	Т	Я	Г	З	П	Л	У	М	В	А	Ц	-	Х	-	-	Б	Ж	-	-	О

Рисунок 1.8 – Соответствие ШО буквам открытого текста

Необходимо подчеркнуть, что изложенные этапы раскрытия шифра простой замены основаны на статистических свойствах открытого текста и для реальных текстов возможны какие-либо отклонения от приведенных частот встречаемости. Поэтому при раскрытии шифра простой замены возможны ошибки и иногда необходим перебор нескольких вариантов.

Здесь разобран пример раскрытия шифра простой замены для текста на русском языке. Однако данная методика сохраняется при раскрытии шифра простой замены для текстов на других языках. При этом используются те же свойства открытого текста: повторяемость букв, сочетаемость букв, чередование гласных и согласных, но с учетом устойчивых статистических характеристик для конкретного языка.

Изложенная методика основана на опыте криптоаналитика и позволяет максимально сократить ошибки и перебор вариантов. Возникающие трудности по раскрытию шифра простой замены обычно связаны с отклонениями от данной методики.

Задание на лабораторную работу.

Расшифруйте сообщение если известно, что оно зашифровано с помощью моноалфавитной замены (в качестве алфавита открытого и шифрованного текста выбраны 32 русских букв). Провести анализ шифротекста и выявить отличительные признаки шифра. Построить таблицу шифрования, которая представляет собой результат проведения криптоанализа.

Варианты заданий:

1. ЯОАСРЮЭЫВЧ_ПЭЛС_ЧПТЯЯЧШБС_ЮХЬПИ_ЮСЫХСЯХХЬФЯМЧТ
РББОПВ_РЯУБСЮХТЩЛЯЬАГЙНСШЮХТРСЪЭЧНСМЩЮСТХФМБЯО
ФЗОБННОСАЫ_РШЭНЮУЮРЩНСДТВКУУ_НШСЭЮВ___ ЦЫСЯТ_

2. ЛПЬО_НУ_ДНЭТНБ_ЪПИ_ЮСЯВЮУЫСАТЪТХГЙПЬОЪСЬЪТХРЭЬАН
НОСПКЭНЯБОТСЫЮСТХФМБЯОФЗОБЪЩХГЫШЫНСЪЩЫССХЫ_АС-
ХРЯЦЬТЩПВЬТВУЬСТ_ДСВПНЮУЖХВ_ТЧЫЭДЛПВ_ВАУЭННБ_

3.РЬНТАЮЮИУЬСЦПФЯХСЭЮЦЮЮВ_ЪЪНЭЪШБ_НЧТНЬ_ЦЬЪНТМЯ
ЪТХКФОЭЪМЬЪНЭЧНАЭУФЪЩПЪНХВЩШСПКСАПЮУЭРНЭЪДХХЬ
ПЯУПФИЗЭЫПЯАПЪНЖГЫПСЯХХЬФЯМПЮЬХСЭЮЭЫЦЧЫШЧНЪДТ

4. ДСВЮГМПЯОФЧЛАННЬЯУПДТРЭЬАННШЩРЫЧДЛСА__ШШСЦЧС
ЦДСЬЯМ_РСРПЬЫЭУФШУП_ТЭ_НТБУЬРНЬЯУПЬОЧТЩЮВЙПИ_ЮС
МПФИСЧЮВСЬАЯЬТЯИЬСЭ_ЧТЬЧ_ЮЮН

5. МЬЬЭ_ЪШЬАПФНЪ_ЫЕЧНЪ_ЫЕ_РПРНЯЧЮХЦАЬТЩПЯЬПФУАННЬ_
ЧП_ЭКГН_ТПЮГИПФНЪ_ЪЯНЛБЧЮЭ_ЧПЪЫФДЯББЦШСРПЬОЪ_ЧБ_
НЬЧЮХСЯХБЦОСА__ШЮФНЯ_НМЬЬЭ_ЪШЬУПРНЭТНА_ПАГРХЯЫЮ

6. НЯ_ЩЮШЦБЧЩЛЯИДСЯЯЪЮРЭУЩСЦПЮЬФЧЩХЫНЭЧСШУШЮ
ХЬПФУФЧЫШРНСЪХЭЧЯРСМПЯОСЭЛФТЩПАЬПЬОЪ_ЪВСЭВГЦПА-
ЬЗЭЬПБОЧФЦБЪУПАЮЮЮИЗЭУЭЯИДСЯБТЫФТЮБ_РПРНСМЩПВРШ

7. РПГУДЯЬЫ_СШРВП_П_Т_ЭТМПВРОЩЙПЪНЯ_ЯБ_МЭЯЬХСЫЮФОБ
_ЮАГРЮСЦПДРХБУЭСДБ_НТВУПЮИПФЬБФЬБСАТЪТШЮНЭТШЮЯУ
ЕСЮХТЩШЩОЕЪЛПЮЬФЧЩШСЦФЧОЫНЫЮХЬПБИЭЬОПАЬПТТРЮ

8. ЭЙЪ_НФЭМПГЬУ_НЖГЬСМНЯБУФДСРЦИТТ_ЛСПВЦАИЧУПРФЧЩР
ПНЭТНЭЧСЮСЯБТРЪДНХКУПАЬФБЬАГШЮЮНОСЭ_ЧТТЪТХЭНЪТШ
ПАЬТЭЦОП_ПЯОПЯОЗДНЦЪХЭННЭЧТ

9. ЮБЬУЪУПЬЬЬАЙНГУ_МНЪ_ЪЯНЛБЧЮПЯОПЬОЦЦИЩСЯБ_ЩПЪН
ТСШРШТКЫНФ_ЪПВ_РЭЬПЦУТЪХЮЮНЪ_ЮЯ_ЮРЗЦШСЪШЬЮЮВ
ЬГГНШСЪКСЮРУЬБТЩШСЫРЦНБЧЪПИ_Ю

10. УИПО_ЮСЯСМЩЮВЙПГУЯЧЮЛС_РЬЦХСШЮЮЭЛП_ХБИПВ ЬХЦ
ЦЭЧЫКСТ_ДСПВНФБАУ_ЪПТНЬМНА_ХФТУЬСЭ__С_ТЪЬМНШЯЯБАЪ
ЗЦШСХРВ_РФЩОПЖШЧНЪ_ЪЯНЛБЧЮПЦУЫТ_ЛС_ЮГЬПЪНБ__ЮС

Контрольные вопросы

1. Криптоанализ – это?
2. Приведите известные методы криптоанализа.
3. Что является ключом шифра простой замены. Каково максимально возможное число ключей шифра простой замены.
4. Какое сообщение проще расшифровать – длинные или короткие?
5. Возможно ли вскрытие шифра простой замены при условии, что известны только статистические данные о языке сообщения и его алфавит?
6. Какие особенности русского языка используются при криптоанализе шифра простой замены?
7. Какой метод криптоанализа применяется в случае, когда известно, что сообщение зашифровано шифром простой замены?

ЛАБОРАТОРНАЯ РАБОТА №4

ШИФР ВИЖЕНЕРА

Теоретические сведения

В случае моноалфавитных подстановок используется только один алфавит шифрования. Существуют шифры, где используется целый набор алфавитов шифрования. Такие шифры называются полиалфавитными и позволяют, в отличие от моноалфавитных подстановок, скрыть естественную частоту появления символов в тексте.

Простая полиалфавитная подстановка (или шифр Виженера) последовательно и циклически меняет используемые алфавиты шифрования. Число используемых алфавитов называется периодом шифра. Для шифрования используется ключ – слово или бессмысленный набор символов нормативного алфавита. Каждая буква ключа определяет свой алфавит шифрования, который получается из нормативного циклического сдвига на количество символов, равное числовому эквиваленту буквы ключа. Очевидно, что длина ключа равна периоду шифра.

Алгоритм шифрования сообщение шифром Вижинера состоит из следующих этапов:

– Под каждой буквой открытого текста помещается буква ключа.

– Ключ циклически повторяется необходимое число раз.

– Вычисляют числовой эквивалент буквы шифротекста, числовой эквивалент буквы ключа складывается по модулю L с числовым эквивалентом буквы открытого текста, где L – мощность нормативного алфавита. То есть шифр Вижинера описывается следующим выражением:

$$Ei = (Mi + Ki(mod\ U)) \ mod\ L,$$

где Ei, Mi – числовые эквиваленты символов криптограммы и открытого текста соответственно, Ki ($mod\ U$) – числовой эквивалент буквы ключа, L – мощность нормативного алфавита, U – длина ключа или период шифра.

– Буквы ключа определяют величину смещения символов криптограммы относительно символов открытого текста. Зашифруем, например, текст «полиалфавитная_подстановка» ключом «краб». Будем использовать алфавит, приведенный на рисунке 1.9.

Нормативный алфавит	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П
Числовые эквиваленты	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Нормативный алфавит	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ы	Ь	Э	Ю	Я	«_»
Числовые эквиваленты	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

Рисунок 1.9 – Пронумерованный алфавит

Процесс шифрования показан на рисунке 1.10.

П	15	К	10	(15+10)mod 32	25	Щ
О	14	Р	16	(14+16)mod 32	30	Я
Л	11	А	0	(11+0) mod 32	11	Л
И	8	Б	1	(8 + 1)mod 32	9	Й
А	0	К	10	(0+10) mod 32	10	К
Л	11	Р	16	(11+16)mod 32	27	Ь
Ф	20	А	0	(20+0) mod 32	20	Ф
А	0	Б	1	(0+1) mod 32	1	Б
В	2	К	10	(2+10) mod 32	12	М
И	8	Р	16	(8+16) mod 32	24	Ш
Т	18	А	0	(18+0) mod 32	18	Т
Н	13	Б	1	(13+1) mod 32	14	О
А	0	К	10	(0+10) mod 32	10	К
Я	30	Р	16	(30+16) mod 32	14	О
_	31	А	0	(31+0) mod 32	31	_
П	15	Б	1	(15+1) mod 32	16	Р
О	14	К	10	(14+10) mod 32	24	Ш
Д	4	Р	16	(4+16) mod 32	20	Ф
С	17	А	0	(17+0) mod 32	17	С
Т	18	Б	1	(18+1) mod 32	19	У
А	0	К	10	(0+10) mod 32	10	К
Н	13	Р	16	(13+16) mod 32	29	Ю
О	14	А	0	(14+0) mod 32	14	О
В	2	Б	1	(2+1) mod 32	3	Г
К	10	К	10	(10+10) mod 32	20	Ф
А	0	Р	16	(0+16) mod 32	16	Р

Рисунок 1.10 - Алгоритм шифрование Вижинера

В результате получилась криптограмма: «ЩЯЛЙКЬФБМШТОКО_РШФСУКЮОГФР». Описанный в лабораторной работе №2 шифр Цезаря является частным случаем шифра Вижинера с периодом, равным единице.

Для того что бы расшифровать сообщение, необходимо проделать те же действия в обратном порядке.

Шифр Виженера «размывает» характеристики частотностей появления символов в тексте, но некоторые особенности появления символов в тексте остаются. Главный недостаток шифра Виженера состоит в том, что его ключ повторяется.

Многопетлевая полиалфавитная подстановка является наиболее интересным подстановочным шифром. В многопетлевом шифре используется не один, а несколько ключей шифрования. Их называют петлевыми или первичными ключами.

Криптоанализ многопетлевых шифров проводится, как и в случае шифра Виженера, в два этапа. Во-первых, необходимо определить период шифра (т.е. длину составного ключа) и, во-вторых, провести частотный анализ по группам периода. Существует **метод Казиски** помогающий определить длину ключа. В открытом тексте сообщения встречаются одинаковые сочетания символов. В процессе криптографического преобразования может так случиться, что эти одинаковые сочетания зашифрованы одинаковой частью ключа. В результате и в криптограмме появиться одинаковые сочетания символов. Криптоаналитик может определить между ними расстояние и разложить это число на множители. Один из множителей будет равным периоду шифра. Если в криптограмме встретилось несколько пар одинаковых сочетаний, нужно определить расстояние для каждой пары и разложить его на множители. Множитель, который встретился чаще других, скорее всего и является искомой длиной ключа. Как только распределение частотностей букв будет сильно отличаться от равномерного (например, по энтропии), то можно говорить о найденной длине ключа. Потом провести криптоанализ.

Однако если ключ будет длинным (неграниченым) и не повторяющим, то данный алгоритм является надёжным.

Для увеличения надежности шифра можно рекомендовать его использование после предварительной псевдослучайной перестановки букв в каждой строке таблицы. Возможны и другие модификации метода.

Шифр Виженера «размывает» характеристики частотностей появления символов в тексте, но некоторые особенности появления символов в тексте остаются. Главный недостаток шифра Виженера состоит в том, что его ключ повторяется. Поэтому простой криптоанализ шифра может быть построен в два этапа:

Поиск длины ключа. Можно анализировать распределение частотностей в зашифрованном тексте с различным прореживанием. То есть брать текст, включающий каждую вторую букву зашифрованного текста, потом каждую третью и т. д. Как только распределение частотностей букв будет сильно отличаться от равномерного (например, по энтропии), то можно говорить о найденной длине ключа.

Существует много других легкозапоминающихся квадратов, которые могут применяться в качестве основы для многоалфавитной системы так же, как и квадрат Виженера. Одним из наиболее известных является квадрат Бофора. Его строками являются строки квадрата Виженера, записанные в обратном порядке. Он назван в честь адмирала сэра Френсиса Бофора — создателя шкалы для определения скорости ветра. Если в квадрате Виженера первая строка и столбец указывают на строки и столбцы соответственно, то в квадрате Бофора этим целям служат первая строка и последний столбец.

Вариант *running key* (бегущий ключ) шифра Виженера когда-то был невзламываемым. Эта версия использует в качестве ключа блок текста, равный по длине исходному тексту. Так как ключ равен по длине сообщению, то методы, предложенные Фридманом и Касиски, не работают (так как ключ не повторяется). В 1920 году Фридман первым обнаружил недостатки этого варианта. Проблема с *running key* шифра Виженера состоит в том, что криптоаналитик имеет статистическую информацию о ключе (учитывая, что блок текста

написан на известном языке) и эта информация будет отражаться в шифрованном тексте. Если ключ действительно случайный, его длина равна длине сообщения и он использовался единожды, то шифр Виженера теоретически будет невзламываемым, фактически этот вариант будет уже шифром Вернама-Виженера, для которого доказана абсолютная криптостойкость.

Несмотря на очевидную стойкость шифра Виженера, он широко не использовался в Европе. Большее распространение получил шифр Гронсфельда, созданный графом Гронсфельдом, идентичный шифру Виженера, за исключением того, что он использовал только 10 различных алфавитов (соответствующих цифрам от 0 до 9). Преимущество шифра Гронсфельда состоит в том, что в качестве ключа используется не слово, а цифровая последовательность, которая повторяется до тех пор, пока не станет равной длине шифруемого сообщения. Шифр Гронсфельда широко использовался по всей Германии и Европе, несмотря на его недостатки.

Одна из возможностей усовершенствования простого многоалфавитного шифра заключается в использовании нескольких многоалфавитных подстановок, применяемых в ходе шифрования открытого текста в зависимости от определенных условий. Семейство шифров, основанных на применении таких методов шифрования, называется полиалфавитными шифрам. Подобные методы шифрования обладают следующими свойствами:

– используется набор связанных многоалфавитных подстановок;

– имеется некоторый ключ, по которому определяется, какое конкретное преобразование должно применяться для шифрования на данном этапе.

Задание на лабораторную работу.

Реализовать алгоритм шифрования Вижинера степени n, среда разработки и язык программирования используется на усмотрение студента. При выполнении лабораторной работы необходимо предусмотреть выдачу на экран: ключа исходного, шифрованного и расшифрованного текстов.

Варианты заданий

1. Криптография решает проблемы секретности, проверки подлинности, целостности.

Ключ: система

2. Протокол – это порядок действий, предпринимаемых двумя или более сторонами.

Ключ: энтропия

3. Информация – сведения, сообщения, данные независимо от формы их представления.

Ключ: партнер

4. Данные – фиксируемые сигналы, воспринимаемые факты окружающего мира.

Ключ: стандарт

5. Информация становится знанием, когда она переработана и проанализирована человеком.

Ключ: требования

6. Три группы свойств информации: атрибутивные, прагматические, динамические.

Ключ: показатель

7. Атрибутивные свойства – свойства, без которых информация не существует.

Ключ: условие

8. Прагматические свойства – свойства, которые характеризуют степень полезности информации

Ключ: техника

9. Динамические свойства – свойства, которые характеризуют изменение информации во времени.

Ключ: эксперт

10. Различают разные измерения информации: техническая и семантическая меры.

Ключ: решение

11. Количество информации – числовая характеристика информации, отражающую степень неопределенности.

Ключ: абонент

12. Качественные показатели информации, определяют её ценность: достоверность, актуальность и надёжность.

Ключ: сообщение

Контрольные вопросы

1. Какой шифр называется совершенным?
2. В чем отличие теоретической и практической стойкости шифра?
3. Определите основные этапы по вскрытию шифра Виженера.
4. От чего зависит стойкость шифра Виженера?
5. Какой из шифров известен как шифр одноразовых блокнотов?

ЛАБОРАТОРНАЯ РАБОТА №5

ШИФРОВАНИЕ RSA

Теоретические сведения

Для решения задачи распределения ключей была выдвинута концепция двухключевой (или асимметричной) криптографии.

В такой схеме для шифрования и дешифрования применяются различные ключи. Для шифрования информации, предназначенной конкретному получателю, используют уникальный открытый ключ (ОК) получателя-адресата.

Асимметричное шифрование – это метод шифрования данных, предполагающий использование двух ключей – открытого и закрытого. Открытый (публичный) ключ применяется для шифрования информации и может передаваться по незащищенным каналам. Закрытый (приватный) ключ применяется для расшифровки данных, зашифрованных открытым ключом. Открытый и закрытый ключи – это большие числа, связанные друг с другом определенной функцией, но так, что, зная одно, крайне сложно вычислить второе.

Асимметричное шифрование используется для защиты информации при ее передаче, также на его принципах построена работа электронных подписей.

Соответственно для дешифрования получатель использует парный секретный ключ (СК). Для передачи открытого ключа от получателя к отправителю секретный канал не нужен. Вместо секретного канала используется аутентичный канал, гарантирующий подлинность источника передаваемой информации (открытого ключа отправителя). Аутентичный канал является открытым и доступен всем, в том числе и противнику.

Асимметричные шифры не имеет смысла использовать для защищенного хранения документов. Их прерогатива – защита сообщений, электронной почты, прикрепленных к посланиям файлов и т.п.

Все криптосистемы с открытым ключом используют представление сообщений в виде целых чисел и преобразование этих чисел в криптограммы,

представляющие собой также целые числа, поэтому математической основой всех систем с открытым ключом является, прежде всего, теория чисел.

Асимметричное шифрование решает главную проблему симметричного метода, при котором для кодирования и восстановления данных используется один и тот же ключ. Если передавать этот ключ по незащищенным каналам, его могут перехватить и получить доступ к зашифрованным данным. С другой стороны, асимметричные алгоритмы гораздо медленнее симметричных, поэтому во многих криптосистемах применяются и те, и другие.

Для ознакомления с асимметричными системами шифрования разберём алгоритм **RSA** (аббревиатура от фамилий *Rivest, Shamir* и *Adelman*) – *криптографически* алгоритм с открытым ключом, основывающийся на вычислительной сложности задачи факторизации (разложения на множители) больших целых чисел. Используется для шифрования данных, борьбы со спамом и для создания цифровых подписей.

Криптосистема *RSA* стала первой системой, пригодной для шифрования и для цифровой подписи. Алгоритм используется в большинстве криптографически приложений.

RSA-ключи генерируются следующим образом:

1. Выбираются два различных случайных простых числа.

2. Вычисляется их произведение, которое называется модулем.

3. Вычисляется значение функции Эйлера от числа:

4. Выбирается целое число (), взаимно простое со значением функции. Обычно в качестве берут простые числа, содержащие небольшое количество единичных бит в двоичной записи, например, простые числа Ферма 17, 257 или 65537.

– Число называется открытой экспонентой (*public exponent*).

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

– Слишком малые значения потенциально могут ослабить безопасность схемы *RSA*.

5. Вычисляется число, мультипликативно обратное к числу по модулю, то есть число, удовлетворяющее сравнению:

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

6. Пара публикуется в качестве открытого ключа RSA (RSA public key).

7. Пара играет роль закрытого ключа RSA (RSA private key) и держится в секрете.

Шифрование и дешифрование.

Предположим, Боб хочет послать Алисе сообщение *m*. Сообщениями являются целые числа в интервале от 0 до *n*-1.

На рисунке 1.10 изображен процесс передачи сообщения.

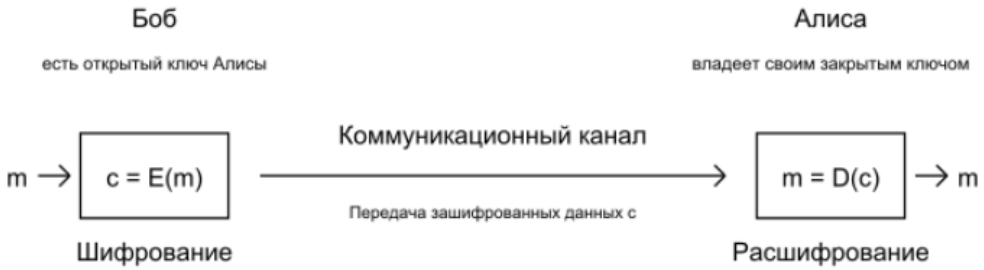

Рисунок 1.10 – Алгоритм RSA

Алгоритм шифрования:

– Взять открытый ключ (*e*, *n*) Алисы

– Взять открытый текст *m*

– Зашифровать сообщение с использованием открытого ключа Алисы:

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

Алгоритм дешифрования:

– Принять зашифрованное сообщение *c*

– Взять свой закрытый ключ (*d*, *n*)

– Применить закрытый ключ для дешифрования сообщения:

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

Рассмотрим на примере (рисунок 1.11).

Этап	Описание операции	Результат операции
Генерация ключей	Выбрать два простых различных числа	$p = 3557.$ $q = 2579$
	Вычислить модуль (произведение)	$n = p \cdot q = 3557 \cdot 2579 = 9173503$
	Вычислить функцию Эйлера	$\varphi(n) = (p-1)(q-1) = 9167368$
	Выбрать открытую экспоненту	$e = 3$
	Вычислить секретную экспоненту	$d = e^{-1} \mod \varphi(n)$ $d = 6111579$
	Опубликовать *открытый ключ*	$\{e, n\} = \{3, 9173503\}$
	Сохранить *закрытый ключ*	$\{d, n\} = \{6111579, 9173503\}$
Шифрование	Выбрать текст для зашифровки	$m = 111111$
	Вычислить шифротекст	$c = E(m)$ $= m^e \mod n$ $= 111111^3 \mod 9173503$ $= 4051753$
Расшифрование	Вычислить исходное сообщение	$m = D(c) =$ $= c^d \mod n$ $= 4051753^{6111579} \mod 9173503$ $= 111111$

Рисунок 1.11 – Нахождение переменных RSA

Алгоритм *RSA* во многом похож на алгоритм *DH* (Диффи-Хеллмана), хотя между ними существует принципиальное различие. Алгоритм *DH* основан на использовании односторонней функции. Предполагая, что *p* и *g* публично известны, можно найти g^x *(mod p)* по заданному *x*, но нельзя или чрезвычайно трудно найти x по известному (перехваченному) g^x *(mod p)*

Алгоритм *RSA*, в свою очередь, основан на использовании односторонней функции с «лазейкой». Зная публично известные n и e, мы можем найти m^e *mod n* по заданному *m*, но не наоборот. При этом, зная, как n раскладывается на множители, выполнить обратную операцию очень легко. Разложение числа n на множители и есть той самой «лазейкой». Если мы знаем эту информацию, то можем легко выполнить обратное действие, а если не знаем, то не можем. Наличие лазейки позволяет применять *RSA* как для шифрования, так и в схеме цифровой подписи.

Простыми словами: алгоритм Диффи – Хеллмана позволяет двум сторонам получить общий секретный ключ, используя незащищенный канал связи.

Однако этот алгоритм не решает проблему аутентификации. Без дополнительных средств пользователи не могут быть уверены, с кем именно они сгенерировали общий секретный ключ. *RSA* алгоритм решает данную проблему.

Надежность алгоритма *RSA* основана на трудности факторизации больших чисел и вычисления дискретных логарифмов в конечных полях.

Комбинирование симметричных и асимметричных алгоритмов

Симметричные алгоритмы быстрые, поэтому ими удобно шифровать большие объёмы информации. Однако для передачи ключа симметричного алгоритма требуется надёжный канал передачи, который очень часто отсутствует. Таким образом, преимущества таких алгоритмов сводятся на нет. С другой стороны, асимметричные алгоритмы не требуют секретного канала для передачи ключа, но на практике криптосистемы с открытым ключом используются для шифрования не сообщений, а ключей. На это есть две основные причины:

1. Алгоритмы шифрования с открытым ключом в среднем работают в тысячи раз медленнее, чем симметричные алгоритмы, а также они требовательны к памяти и вычислительной мощности компьютера, поэтому большие тексты кодировать этими алгоритмами нецелесообразно.

2. Алгоритмы шифрования с открытым ключом уязвимы по отношению к криптоаналитическим атакам со знанием открытого текста.

Возможно следующее решение: сообщение шифруется симметричным алгоритмом, что позволяет выиграть в скорости, т.к. сообщение может быть сколь угодно большим, а ключ симметричного алгоритма шифруется асимметричным алгоритмом.

Задание на лабораторную работу.

Реализовать программно-комбинированное шифрования используя асимметричный алгоритм *RSA* и любой симметричный алгоритм. Среда разработки, язык программирования и симметричный алгоритм используется на

усмотрение студента. При выполнении лабораторной работы необходимо предусмотреть вывод на экран: параметров p и q, открытого и закрытого ключей, исходного, зашифрованного и расшифрованного текстов.

Контрольные вопросы

1. Какие числа называются простыми?

2. Определите назначение открытого и секретного ключей в системе RSA.

3. Почему схема RSA называется асимметричной?

4. Какие особенности системы RSA необходимо учитывать при ее практическом использовании?

5. Какое число является простым?

6. Какому требованию должны удовлетворять простые делители модуля в системе RSA?

7. Нужен ли секретный канал связи при шифровании алгоритмом RSA? Обоснуйте ответ.

8. Для какого процесса (шифрование или дешифрование) используется открытый ключ в системе RSA?

9. Для какого процесса (шифрование или дешифрование) используется секретный ключ в системе RSA?

ЛАБОРАТОРНАЯ РАБОТА №6

ИССЛЕДОВАНИЕ ПРОЦЕССА ПОСТРОЕНИЯ ЭЛЕКТРОННОЙ ПОДПИСИ НА ОСНОВЕ АЛГОРИТМА RSA С ИСПОЛЬЗОВАНИЕМ ПРОГРАММНОЙ РЕАЛИЗАЦИИ

Теоретические сведения

Электронная подпись (ЭП), Электронная цифровая подпись (ЭЦП) – информация в электронной форме, присоединенная к другой информации в электронной форме (электронный документ) или иным образом связанная с такой информацией. Используется для определения лица, подписавшего информацию (электронный документ).

Электронная подпись предназначена для идентификации лица, подписавшего электронный документ, и является полноценной заменой собственноручной подписи в случаях, предусмотренных законом.

Электронная подпись выполняет следующие функции:

– Контроль целостности передаваемого документа: при любом случайном или преднамеренном изменении документа подпись станет недействительной, потому что вычислена она на основании исходного состояния документа и соответствует лишь ему.

– Защиту от изменений (подделки) документа: гарантия выявления подделки при контроле целостности делает подделывание нецелесообразным в большинстве случаев.

– Невозможность отказа от авторства. Так как создать корректную подпись можно, лишь зная закрытый ключ, а он известен только владельцу, он не может отказаться от своей подписи под документом.

– Доказательное подтверждение авторства документа: Так как создать корректную подпись можно, лишь зная закрытый ключ, а он известен только владельцу, он может доказать своё авторство подписи под документом. В зависимости от деталей определения документа могут быть подписаны такие поля, как «автор», «внесённые изменения», «метка времени» и т. д.

Использование хеш-функций даёт следующие преимущества: Стоит заметить, что использование хеш-функции не обязательно при электронной подписи, а сама функция не является частью алгоритма ЭП, поэтому хеш-функция может использоваться любая или не использоваться вообще.

Виды асимметричных алгоритмов ЭП Обеспечение этого во всех асимметричных алгоритмах цифровой подписи опирается на следующие вычислительные задачи:

– Вычисления также могут производиться двумя способами: на базе математического аппарата эллиптических кривых и на базе полей Галуа. В настоящее время самые быстрые алгоритмы дискретного логарифмирования и факторизации являются субэкспоненциальными. Принадлежность самих задач к классу NP-полных не доказана.

– Схемы электронной подписи могут быть одноразовыми и многоразовыми. В одноразовых схемах после проверки подлинности подписи необходимо провести замену ключей, в многоразовых схемах это делать не требуется.

Идея использования систем асимметричного шифрования с открытыми ключами для построения систем цифровой подписи как бы заложена в постановке задачи. Действительно, пусть имеется пара преобразований (*E, D*), первое из которых зависит от открытого ключа, а второе – от секретного. Для того чтобы вычислить цифровую подпись *S* для сообщения, владелец секретного ключа может применить к сообщению *M* второе преобразование *D: S = D(M).* В таком случае вычислить подпись может только владелец секретного ключа, в то время как проверить равенство *E(S) = M* может каждый. Основными требованиями к преобразованиям *E* и *D* являются:

– выполнение равенства *M = E(D(M))* для всех сообщений *M*;

– невозможность вычисления значения *D(M*) для заданного сообщения *M* без знания секретного ключа.

Отличительной особенностью предложенного способа построения цифровой подписи является возможность отказаться от передачи самого подписываемого сообщения *M*, так как его можно восстановить по значению подписи.

В связи с этим подобные системы называют схемами цифровой подписи с восстановлением текста.

Заметим, что если при передаче сообщение дополнительно шифруется с помощью асимметричного шифра, то пара преобразований (*E, D*), используемая в схеме цифровой подписи, должна отличаться от той, которая используется для шифрования сообщений. В противном случае появляется возможность передачи в качестве шифрованных ранее подписанных сообщений. При этом более целесообразно шифровать подписанные данные, чем делать наоборот, то есть подписывать шифрованные данные, поскольку в первом случае противник получит только шифротекст, а во втором – и открытый, и шифрованный тексты.

Очевидно, что рассмотренная схема цифровой подписи на основе пары преобразований (*E, D*) удовлетворяет требованию невозможности подделки, в то время как требование невозможности создания подписанного сообщения не выполнено: для любого значения S каждый может вычислить значение $M = E(S)$ и тем самым получить подписанное сообщение. Требование невозможности подмены сообщения заведомо выполняется, так как преобразование E взаимно однозначно.

Для защиты от создания злоумышленником подписанного сообщения можно применить некоторое взаимно однозначное отображение $R: M \rightarrow M^*$, вносящее избыточность в представление исходного сообщения, например, пу-
64 тем увеличения его длины, а затем уже вычислять подпись $S = D(M^*)$.

В этом случае злоумышленник, подбирая *S* и вычисляя значения $M^* = E(S)$, будет сталкиваться с проблемой отыскания таких значений M^*, для которых существует прообраз M. Если отображение *R* выбрано таким, что число возможных образов M^* значительно меньше числа всех возможных последовательностей той же длины, то задача создания подписанного сообщения будет сложной.

Другой подход к построению схем цифровых подписей на основе систем шифрования с открытым ключом состоит в использовании бесключевых хэш-

функций. Для заданного сообщения M сначала вычисляется значение хэш-функций *h(M),* а затем уже значение подписи *S = D(h(M)).* Ясно, что в таком случае по значению подписи уже нельзя восстановить сообщение. Поэтому подписи необходимо передавать вместе с сообщениями. Такие подписи получили название цифровых подписей с дополнением. Заметим, что системы подписи, построенные с использованием бесключевых хэш-функций, заведомо удовлетворяют всем требованиям, предъявляемым к цифровым подписям.

Задание на лабораторную работу

1. Изучить теоретический материал работы.

2. Провести исследование процедуры построения подписи RSA. Передаваемое сообщение m и кодирующее число p.

Студенты самостоятельно выбирают значение ключей для обеих сторон (A и B) и исследуют процесс зашифрования и дешифрования сообщения. Отчет по лабораторной работе, оформленный письменно в рабочей тетради, должен содержать процесс исследование процедуры построения цифровой подписи на основе RSA по своему варианту.

Варианты заданий

Вариант	Кодирующее число	Кодирующее число	Сообщение *m*
1	53	7	17
2	47	13	18
3	43	11	19
4	41	17	20

Контрольные вопросы

1. Основные принципы построения цифровых подписей.

2. Основные характеристики построения цифровых подписей на основе алгоритма RSA.

3. Достоинства и недостатки построения цифровой подписи на основе алгоритма RSA.

ЛАБОРАТОРНАЯ РАБОТА №7

ШИФРОВАНИЕ SHA-256

Теоретические сведения

Криптографические методы позволяют обеспечить конфиденциальность данных, устанавливать подлинность отправителя и контролировать целостность передаваемых сообщений.

Целостность – это гарантия того, что информация сейчас существует в ее исходном виде, то есть при ее хранении или передаче не было произведено несанкционированных изменений. Целостность предполагает неизменность информационных объектов от их исходного состояния, определяемого автором или источником информации. Целостность является важнейшим аспектом информационной безопасности в тех случаях, когда информация используется для управления различными процессами, например, техническими, социальными и т.д. Так, ошибка в управляющей программе приведет к остановке управляемой системы, неправильная трактовка закона может привести к его нарушениям, точно так же неточный перевод инструкции по применению лекарственного препарата может нанести вред здоровью. Все эти примеры иллюстрируют нарушение целостности информации, что может привести к катастрофическим последствиям.

Информация, хранимая в технических средствах, может подвергаться случайному или умышленному несанкционированному изменению. Одним из способов проверки целостности информации является хранение эталонных копий, используемых для сравнения. Преимущество этого метода состоит в том, что не только обнаруживаются любые возможные изменения, но и возможно 100 % быстрое восстановление исходного состояния информации. Однако такая процедура при необходимости частых проверок целостности данных большого объема приводит к существенным временным задержкам, так же существенно (в два раза) увеличивается требуемый для хранения информации

объем памяти, а в случае сжатия информации требуются еще и дополнительные преобразования, поэтому этот метод применяется только в специальных случаях.

Более технологичным является способ, основанный на вычислении некоторых контрольных сумм. Вычисляются по заданному алгоритму контрольные суммы для массивов данных (например, файлов, каталогов), находящихся в исходном состоянии. Эти вычисленные контрольные суммы записываются в таблицы, которые затем используются для проверки целостности информации. В этом случае проверка целостности состоит в вычислении по заданному алгоритму контрольной суммы для данного блока информации и сопоставления полученного значения с эталонным значением контрольной суммы. Выигрыш состоит в том, что, во-первых, теперь нет необходимости проводить сравнение двух больших массивов данных и иметь существенный дополнительный объем памяти, а во-вторых, появляется больше возможностей по нейтрализации угроз преднамеренного синхронного изменения исходного массива и его контрольной копии. Целостность проверяется путем сравнения, например, 64-битовых, 128-битовых или 256-битовых контрольных сумм. Такие контрольные суммы называются защитными контрольными суммами (ЗКС).

При фиксации исходного состояния для информационных массивов вычисляются их образы, имеющие функциональную зависимость от всех, или некоторой части символов информационного массива. Таким образом, при изменении составляющих информационного массива изменяются (или должны изменяться) и вычисляемые при проверке их информационные образы.

В зависимости от объемов хранимых информационных массивов, существующих технических ограничений и характера воздействия на хранимую информацию могут выбираться различные способы получения эталонных образов информации. При этом такие образы могут иметь большую, равную или меньшую длину, чем исходная информация, а функциональная зависимость символов образа от символов исходной информации может быть известной или содержать скрываемые (секретные) составляющие. Последнее особенно

актуально в случае, когда имеет место целенаправленное умышленное воздействие на хранимую информацию. В случае, когда объемы хранимой информации являются достаточно большими, а допустимое время проверки ограничено – применяют функциональные преобразования, дающие образы исходной информации меньшей длины, чем длина защищаемого информационного массива. Такие преобразования принято называть хэш-функциями, процесс получения образа информации – **хэшированием**, а значение хэш-функции – хэш-образом.

Хэш-функция представляет собой криптографическую функцию от сообщения произвольной длины, значение которой зависит сложным образом от каждого бита сообщения. Хэш-функции можно разделить на два класса:

– Бесключевые хэш-функции – имеют один вход (сообщение).

– Ключевые хэш-функции – имеют два входа (сообщение и ключ).

Хеш-функция (англ. *hash* мещанина/путаница), или функция свёртки – функция, осуществляющая преобразование массива входных данных произвольной длины в выходную битовую строку установленной длины, выполняемое определённым алгоритмом. Преобразование, производимое хеш-функцией, называется хешированием. Исходные данные называются входным массивом или «сообщением». Результат преобразования называется «хешем», «хеш-кодом» или «хеш-суммой»

В общем случае (согласно принципу Дирихле) нет однозначного соответствия между хеш-кодом и исходными данными. Возвращаемые хеш-функцией значения менее разнообразны, чем значения входного массива. Случай, при котором хеш-функция преобразует более чем один массив входных данных в одинаковый код, называется коллизией. Вероятность возникновения коллизий используется для оценки качества хеш-функций.

Хеш-функции предназначены для создания «отпечатков» для сообщений произвольной длины. Применяются в различных приложениях или компонентах, связанных с защитой информации.

Криптографические хэш-функции – это математические операции, выполняемые с цифровыми данными; сравнивая вычисления с известным и ожидаемым хэш-значением, человек может определить целостность данных. Например, вычисление хэша загруженного файла и сравнение результата с ранее опубликованным результатом хэша может показать, была ли загрузка изменена или подделана.

Ключевым аспектом криптографических хэш-функций является их сопротивление столкновению: никто не должен быть в состоянии найти два разных входных значения, которые приводят к одному и тому же хэш-выходу.

SHA-2 (англ. *Secure Hash Algorithm Version 2* – безопасный алгоритм хеширования, версия 2) – семейство криптографических алгоритмов – однонаправленных и детерминированных хеш-функций, включающее в себя алгоритмы *SHA-224, SHA-256, SHA-384, SHA512, SHA-512/256* и *SHA-512/224*.

Хеш-функции семейства *SHA-2* построены на основе структуры Меркла – Дамгора. Исходное сообщение после дополнения разбивается на блоки, каждый блок – на 16 слов. Алгоритм пропускает каждый блок сообщения через цикл с 64 или 80 итерациями (раундами). На каждой итерации 2 слова преобразуются, функцию преобразования задают остальные слова. Результаты обработки каждого блока складываются, сумма является значением хеш-функции. Тем не менее, инициализация внутреннего состояния производится результатом обработки предыдущего блока. Поэтому независимо обрабатывать блоки и складывать результаты нельзя.

Хэш-сумма по алгоритму *SHA-256* генерируются следующим образом:

Шаг 1. Предварительная обработка. Преобразуем сообщение в двоичный вид. Добавим одну единицу в конце. Заполняем нулями до тех пор, пока данные не станут кратны 512 без последних 64 бит.

Шаг 2. Инициализация значений хэша. Создадим 8 значений хэша. Это константы, представляющие первые 32 бита дробных частей квадратных корней первых 8 простых чисел: 2, 3, 5, 7, 11, 13, 17, 19.

Шаг 3. Инициализация округлённых констант. Создадим ещё немного констант, на этот раз их 64. Каждое значение – это первые 32 бита дробных частей кубических корней первых 64 простых чисел (2–311).

Шаг 4. Создаём очередь сообщений. Копируем входные данные из шага 1 в новый массив, где каждая запись является 32-битным словом. Добавляем ещё 48 слов, инициализированных нулями, чтобы получить массив. Изменяем нулевые индексы в конце массива.

Шаг 5. Цикл сжатия. Инициализируем переменные *a, b, c, d, e, f, g, h* и установим их равными текущим значениям хэша соответственно. *h0, h1, h2, h3, h4, h5, h6, h7*. Запустим цикл сжатия, который будет изменять значения *a…h*.

Шаг 6. Изменяем окончательные значения После цикла сжатия, но ещё внутри основного цикла, мы модифицируем значения хэша, добавляя к ним соответствующие переменные *a…h*. Сложение происходит по модулю 2^{32} .

Шаг 7. Получаем финальный хэш.

Рассмотрим на примере: *SHA-256 «hello world».*

1. Преобразуем «*hello world*» в двоичный вид:

01101000 01100101 01101100 01101100 01101111 00100000 01110111 01101111 01110010 01101100 01100100

2. Добавим одну единицу:

01101000 01100101 01101100 01101100 01101111 00100000 01110111 01101111 01110010 01101100 01100100 1

3. Заполняем нулями до тех пор, пока данные не станут кратны 512 без последних 64 бит (в нашем случае 448 бит):

01101000 01100101 01101100 01101100 01101111 00100000 01110111 01101111
01110010 01101100 01100100 10000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

4. Добавим 64 бита в конец, где 64 бита — целое число с порядком байтов big-endian, обозначающее длину входных данных в двоичном виде. В нашем случае 88, в двоичном виде — «1011000». Теперь у нас есть ввод, который всегда будет без остатка делиться на 512.

01101000 01100101 01101100 01101100 01101111 00100000 01110111 01101111
01110010 01101100 01100100 10000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 01011000

5. Инициализация значений хеша (*h*). Создадим 8 значений хеша. Это константы, представляющие первые 32 бита дробных частей квадратных корней первых 8 простых чисел: 2, 3, 5, 7, 11, 13, 17, 19.

h0 := 0x6a09e667
h1 := 0xbb67ae85
h2 := 0x3c6ef372
h3 := 0xa54ff53a
h4 := 0x510e527f
h5 := 0x9b05688c
h6 := 0x1f83d9ab
h7 := 0x5be0cd19

6. Инициализация округлённых констант (*k*). Создадим ещё немного констант, на этот раз их 64. Каждое значение — это первые 32 бита дробных частей кубических корней первых 64 простых чисел (2–311).

0x428a2f98 0x71374491 0xb5c0fbcf 0xe9b5dba5 0x3956c25b 0x59f111f1 0x923f82a4 0xab1c5ed5
0xd807aa98 0x12835b01 0x243185be 0x550c7dc3 0x72be5d74 0x80deb1fe 0x9bdc06a7 0xc19bf174
0xe49b69c1 0xefbe4786 0x0fc19dc6 0x240ca1cc 0x2de92c6f 0x4a7484aa 0x5cb0a9dc 0x76f988da
0x983e5152 0xa831c66d 0xb00327c8 0xbf597fc7 0xc6e00bf3 0xd5a79147 0x06ca6351 0x14292967
0x27b70a85 0x2e1b2138 0x4d2c6dfc 0x53380d13 0x650a7354 0x766a0abb 0x81c2c92e 0x92722c85
0xa2bfe8a1 0xa81a664b 0xc24b8b70 0xc76c51a3 0xd192e819 0xd6990624 0xf40e3585 0x106aa070
0x19a4c116 0x1e376c08 0x2748774c 0x34b0bcb5 0x391c0cb3 0x4ed8aa4a 0x5b9cca4f 0x682e6ff3
0x748f82ee 0x78a5636f 0x84c87814 0x8cc70208 0x90befffa 0xa4506ceb 0xbef9a3f7 0xc67178f2

7. Основной цикл. Следующие шаги будут выполняться для каждого 512-битного «куска» входных данных. Наша тестовая фраза «*hello world*» довольно короткая, поэтому «кусок» всего один. На каждой итерации цикла мы будем изменять значения хеш-функций *h0–h7*, чтобы получить окончательный результат.

8. Создаём очередь сообщений (*w*). 1. Копируем входные данные из шага 1 в новый массив, где каждая запись является 32-битным словом:

01101000011001010110110001101100 01101111001000000111011101101111
01110010011011000110010010000000 00000000000000000000000000000000
00000000000000000000000000000000 00000000000000000000000000000000
00000000000000000000000000000000 00000000000000000000000000000000
00000000000000000000000000000000 00000000000000000000000000000000
00000000000000000000000000000000 00000000000000000000000000000000
00000000000000000000000000000000 00000000000000000000000000000000
00000000000000000000000000000000 00000000000000000000000001011000

9. Добавляем ещё 48 слов, инициализированных нулями, чтобы получить массив *w* [0…63]:

01101000011001010110110001101100 01101111001000000111011101101111
01110010011011000110010010000000 00000000000000000000000000000000
00000000000000000000000000000000 00000000000000000000000000000000
00000000000000000000000000000000 00000000000000000000000000000000
00000000000000000000000000000000 00000000000000000000000000000000
00000000000000000000000000000000 00000000000000000000000000000000
00000000000000000000000000000000 00000000000000000000000000000000
00000000000000000000000000000000 00000000000000000000000001011000
00000000000000000000000000000000 00000000000000000000000000000000
00000000000000000000000000000000 00000000000000000000000000000000
00000000000000000000000000000000 00000000000000000000000000000000
00000000000000000000000000000000 00000000000000000000000000000000
...
00000000000000000000000000000000 00000000000000000000000000000000

10. Изменяем нулевые индексы в конце массива, используя алгоритм:

For i from w[16…63]:
s0 = (w[i-15] rightrotate 7) xor (w[i-15] rightrotate 18) xor (w[i-15] righthift 3)
s1 = (w[i-2] rightrotate 17) xor (w[i-2] rightrotate 19) xor (w[i-2] righthift 10)
w [i] = w[i-16] + s0 + w[i-7] + s1

w[1] rightrotate 7:
01101111001000000111011101101111 -> 11011110110111100100000011101110
w[1] rightrotate 18:
01101111001000000111011101101111 -> 00011101110110111101101111001000
w[1] rightshift 3:
01101111001000000111011101101111 -> 00001101111001000000111011101101
s0 = 11011110110111100100000011101110 XOR
00011101110110111101101111001000 XOR 00001101111001000000111011101101
s0 = 11001110111000110010101111001011
w[14] rightrotate 17:
00000000000000000000000000000000 -> 00000000000000000000000000000000
w[14] rightrotate19:
00000000000000000000000000000000 -> 00000000000000000000000000000000
w[14] rightshift 10:
00000000000000000000000000000000 -> 00000000000000000000000000000000
s1 = 00000000000000000000000000000000 XOR
00000000000000000000000000000000 XOR 00000000000000000000000000000000
s1 = 00000000000000000000000000000000
w[16] = w[0] + s0 + w[9] + s1
w[16] = 01101000011001010110110001101100 +
11001110111000011001010111001011 + 00000000000000000000000000000000 +
00000000000000000000000000000000
// сложение рассчитывается по модулю 2^{32}
w[16] = 00110111010001110000001000110111

Это оставляет нам 64 слова в нашей очереди сообщений (*w*):

01101000011001010110110001101100 01101111001000000111011101101111
01110010011011000110010010000000 00000000000000000000000000000000
00000000000000000000000000000000 00000000000000000000000000000000
00000000000000000000000000000000 00000000000000000000000000000000
00000000000000000000000000000000 00000000000000000000000000000000
00000000000000000000000000000000 00000000000000000000000000000000
00000000000000000000000000000000 00000000000000000000000000000000
00000000000000000000000000000000 00000000000000000000000001011000
00110111010001110000001000110111 10000110110100001100000000110001
11010011101111010001000100001011 01111000001111110100011110000010
00101010100100000111110011101101 01001011001011110111110011001001
00110001111000011001010001011101 10001001001101100100100101100100
01111111011110100000011011011010 11000001011110011010100100111010
10111011111010001111011001010101 00001100000110101110001111100110
10110000111111100000110101111101 01011111011011100101010110010011
00000000100010011001101101010010 00000111111100011100101010010100
00111011010111111110010111010110 01101000011001010110001011100110
11001000010011100000101010011110 00000110101011111001101100100101

10010010111011110110010011010111 01100011111110010101111001011010
11100011000101100110011111010111 10000100001110111101111000010110
11101110111011001010100001011011 10100000010011111111001000100001
11111001000110001010110110111000 00010100101010001001001000011001
00010000100001000101001100011101 01100000100100111110000011001101
10000011000000110101111111101001 11010101101011100111100100111000
00111001001111110000010110101101 11111011010010110001101111101111
11101011011101011111111100101001 01101010001101101001010100110100
00100010111111001001110011011000 10101001011101000000110100101011
01100000110011110011100010000101 11000100101011001001100000111010
00010001010000101111110110101101 10110000101100000001110111011001
10011000111100001100001101101111 01110010000101111011100000011110
10100010110101000110011110011010 00000001000011111001100101111011
11111100000101110100111100001010 11000010110000101110101100010110

10. Цикл сжатия. Инициализируем переменные *a, b, c, d, e, f, g, h* и установим их равными текущим значениям хеша соответственно.*h0, h1, h2, h3, h4, h5, h6, h7.*

Запустим цикл сжатия, который будет изменять значения *a…h* . Цикл выглядит следующим образом:

for i from 0 to 63
S1 = (e rightrotate 6) xor (e rightrotate 11) xor (e rightrotate 25)
ch = (e and f) xor ((not e) and g)
temp1 = h + S1 + ch + k[i] + w[i]
S0 = (a rightrotate 2) xor (a rightrotate 13) xor (a rightrotate 22)
maj = (a and b) xor (a and c) xor (b and c)
temp2 := S0 + maj
h = g
g = f
e = d + temp1
d = c
c = b
b = a
a = temp1 + temp2

11. Давайте пройдём первую итерацию. Сложение рассчитывается по модулю2^{32}:

a = 0x6a09e667 = 01101010000010011110011001100111
b = 0xbb67ae85 = 10111011011001111010111010000101
c = 0x3c6ef372 = 00111100011011011110011011110010

d = 0xa54ff53a = 10100101010011111111010100111010
e = 0x510e527f = 01010001000011100101001001111111
f = 0x9b05688c = 10011011000001010110100010001100
g = 0x1f83d9ab = 00011111100000111101100110101011
h = 0x5be0cd19 = 01011011111000001100110100011001
e rightrotate 6:
01010001000011100101001001111111 -> 11111101010001000011100101001001
e rightrotate 11:
01010001000011100101001001111111 -> 01001111111010100010000111001010
e rightrotate 25:
01010001000011100101001001111111 -> 10000111001010010011111110101000
S1 = 11111101010001000011100101001001 XOR
01001111111010100010000111001010 XOR 10000111001010010011111110101000
S1 = 00110101100001110010011100101011
e and f:
01010001000011100101001001111111
& 10011011000001010110100010001100 =
00010001000001000100000000001100
not e:
01010001000011100101001001111111 -> 10101110111100011010110110000000
(not e) and g:
10101110111100011010110110000000
& 00011111100000111101100110101011 =
00001110100000011000100110000000
ch = (e and f) xor ((not e) and g)
= 00010001000001000100000000001100 xor
00001110100000011000100110000000
= 00011111100001011100100110001100
// k[i] is the round constant
// w[i] is the batch
temp1 = h + S1 + ch + k[i] + w[i]
temp1 = 01011011111000001100110100011001 +
00110101100001110010011100101011 + 00011111100001011100100110001100 +
10000101000101000101111110011000 + 01101000011001010110110001101100
temp1 = 01011011110111010101100111010100
a rightrotate 2:
01101010000010011110011001100111 -> 11011010100000100111100110011001
a rightrotate 13:
01101010000010011110011001100111 -> 00110011001110110101000001001111
a rightrotate 22:
01101010000010011110011001100111 -> 00100111100110011001110110101000
S0 = 11011010100000100111100110011001 XOR
00110011001110110101000001001111 XOR 00100111100110011001110110101000
S0 = 11001110001000001011010001111110
a and b:

01101010000010011110011001100111
& 10111011011001111010111010000101 =
00101010000000001101001100000010 1
a and c:
01101010000010011110011001100111
& 00111100011011101111001101110010 =
00101000000010001110001001100010
b and c:
10111011011001111010111010000101
& 00111100011011101111001101110010 =
00111000011001101010001000000000
maj = (a and b) xor (a and c) xor (b and c)
= 00101010000000011010011000000101 xor
00101000000010001110001001100010 xor 00111000011001101010001000000000
= 00111010011011111110011001100111
temp2 = S0 + maj
= 11001110001000001011010001111110 + 00111010011011111110011001100111
= 00001000100100001001101011100101
h = 00011111100000111101100110101011
g = 10011011000001010110100010001100
f = 01010001000011100101001001111111
e = 10100101010011111111010100111010 +
01011011110111010101100111010100
= 00000001001011010100111100001110
d = 00111100011011101111001101110010
c = 10111011011001111010111010000101
b = 01101010000010011110011001100111
a = 01011011110111010101100111010100 +
00001000100100001001101011100101
= 01100100011011011111010010111001

12. Все расчёты выполняются ещё 63 раза, изменяя переменные a…h. В итоге мы должны получить следующее:

h0 = 6A09E667 = 01101010000010011110011001100111
h1 = BB67AE85 = 10111011011001111010111010000101
h2 = 3C6EF372 = 00111100011011101111001101110010
h3 = A54FF53A = 10100101010011111111010100111010
h4 = 510E527F = 01010001000011100101001001111111
h5 = 9B05688C = 10011011000001010110100010001100
h6 = 1F83D9AB = 00011111100000111101100110101011
h7 = 5BE0CD19 = 01011011111000001100110100011001
a = 4F434152 = 00100111101000011010000010101010
b = D7E58F83 = 01101011111100101100011111000011
c = 68BF5F65 = 00110100010111111010111101100101

d = 352DB6C0 = 00011010100101101101011011000000
e = 73769D64 = 01110011011101101001110101100100
f = DF4E1862 = 11011111010011100001100001100010
g = 71051E01 = 01110001000001010001111000000001
h = 870F00D0 = 10000111000011110000000011010000

13. Изменяем окончательные значения. После цикла сжатия, но ещё внутри основного цикла, мы модифицируем значения хеша, добавляя к ним соответствующие переменные *a…h*. Как обычно, всё сложение происходит по модулю 2^{32}.

h0 = h0 + a = 10111001010011010010011110111001
h1 = h1 + b = 10010011010011010011111000001000
h2 = h2 + c = 10100101001011100101001011010111
h3 = h3 + d = 11011010011111011010101111111010
h4 = h4 + e = 11000100100001001110111111100011
h5 = h5 + f = 01111010010100111000000011101110
h6 = h6 + g = 10010000100010001111011110101100
h7 = h7 + h = 11100010111011111100110111101001

14. Получаем финальный хеш. И последний важный шаг - собираем всё вместе.

digest = h0 append h1 append h2 append h3 append h4 append h5 append h6 append h7 = B94D27B9934D3E08A52E52D7DA7DABFAC484EFE37A5380EE9088F7ACE2EFCDE9

Задание на лабораторную работу.

Вариант задания выбирается студентом самостоятельно.

Вариант 1.

Реализовать программно алгоритм хэщирования *SHA256*, без использования библиотеки. Среда разработки, язык программирования и симметричный алгоритм используется на усмотрение студента. При выполнении лабораторной работы необходимо предусмотреть выдачу на экран: исходного сообщения, его двоичную запись и хэш-суммы. Можно использовать словосочетание-константу на любом языке.

Вариант 2.

Реализовать программно алгоритм хэширования *SHA256*. Хэшируемый текст должен храниться в файле и содержать: буквы на кириллице и латинице, цифры, знаки препинания, специальные символы. Полученное хэш-значение должно сохраняться в файл. Среда разработки и язык программирования выбирается на усмотрение студента. Выполнить проверку. Добавить в код программы сравнение полученного результата и хранимого хэша, полученного ранее (контрольный документ), выдать сообщение о сохранности документа или его изменении. Сравнить полученную хэш-сумму документа с хэш-суммой вычисляемой системой *Windows/Linux/MacOS*.

Контрольные вопросы

1. Дайте определение хэш-функции.

2. Почему изменение одного бита в конце сообщения приводит к разным хэш-функциям?

3. Почему перестановка двух слов в тексте, которая не меняет смысл текста и его объем влияет на полученные значения хэш-функций?

4. Какие основные требования предъявляются к хэш-функциям?

5. Что такое коллизии хэш-функции?

6. Какова вероятность того, что две произвольные последовательности символов будут иметь равные хэш-образы для хэш-функции с n-битным выходным значением?

7. Какими свойствами обладают хэш-функции?

8. Дайте определение однонаправленной функции.

9. Что означают принципы рассеивания и перемешивания, реализованные в алгоритмах получения хэш-образов файлов?

ЛАБОРАТОРНАЯ РАБОТА №8

ИДЕНТИФИКАЦИЯ И АУТЕНТИФИКАЦИЯ

Теоретические сведения

Идентификацию и аутентификацию можно считать основой программно-технических средств безопасности, поскольку остальные сервисы рассчитаны на обслуживание именованных субъектов. Идентификация и аутентификация – это первая линия обороны, «проходная» информационного пространства организации (рисунок 1.12).

Рисунок 1.12 – Структурная схема терминов

Идентификация и аутентификации применяются для ограничения доступа случайных и незаконных субъектов (пользователи, процессы) информационных систем к ее объектам (аппаратные, программные и информационные ресурсы).

Общий алгоритм работы таких систем заключается в том, чтобы получить от субъекта (например, пользователя) информацию, удостоверяющую его личность, проверить ее подлинность и затем предоставить (или не предоставить) этому пользователю возможность работы с системой.

Наличие процедур аутентификации и/или идентификации пользователей является обязательным условием любой защищенной системы, поскольку все механизмы защиты информации рассчитаны на работу с поименованными субъектами и объектами информационных систем.

Стойкость подсистемы идентификации и аутентификации (И/А) пользователя в системе защиты информации (СЗИ) во многом определяет устойчивость к взлому самой СЗИ. Данная стойкость определяется гарантией того, что злоумышленник не сможет пройти аутентификацию, присвоив чужой идентификатор или украв его. По используемому методу подсистемы И/А делят на четырегруппы:

– основанные на некоторой секретной информации, например, пароля;

– основанные на использовании уникального предмета: карты, жетона;

– основанные на измерении биометрических параметров человека – физиологических или поведенческих атрибутах (отпечатки пальцев, почерк, голос, геометрия лица/руки);

– основанные на информации, ассоциированной с пользователем, например, с его координатами.

Дадим определения этих понятий.

Идентификация – присвоение субъектам и объектам доступа личного идентификатора и сравнение его с заданным.

Аутентификация (установление подлинности) – проверка принадлежности субъекту доступа предъявленного им идентификатора и подтверждение его подлинности. Другими словами, аутентификация заключается в проверке: является ли подключающийся субъект тем, за кого он себя выдает.

Также существует ещё одно родственное понятие: **авторизация** – это предоставление доступа к какому-либо ресурсу на основе результатов идентификации и аутентификации пользователя.

При построении систем идентификации и аутентификации возникает проблема выбора идентификатора, на основе которого осуществляются процедуры идентификации и аутентификации пользователя. В качестве идентификаторов обычно используют:

– набор символов (пароль, секретный ключ, персональный идентификатор и т. п.), который пользователь запоминает или для их запоминания использует специальные средства хранения (электронные ключи);

– физиологические параметры человека (отпечатки пальцев, рисунок радужной оболочки глаза и т. п.) или особенности поведения (особенности работы на клавиатуре и т. п.).

Наиболее распространенными простыми и привычными являются методы аутентификации, основанные на паролях – конфиденциальных идентификаторах субъектов. В этом случае при вводе субъектом своего пароля подсистема аутентификации сравнивает его с паролем, хранящимся в базе эталонных данных в зашифрованном виде. В случае совпадения паролей подсистема аутентификации разрешает доступ к ресурсам системы.

Парольные методы аутентификации по степени изменяемости паролей делятся на:

– использующие постоянные (многократно используемые) пароли;

– использующие одноразовые (динамично изменяющиеся) пароли.

Использование одноразовых или динамически меняющихся паролей является более надежным методом парольной защиты.

В последнее время получили распространение комбинированные методы идентификации и аутентификации, требующие, помимо знания пароля, наличие карточки (***token***) – специального устройства, подтверждающего подлинность субъекта.

Карточки разделяют на два типа:

– пассивные (карточки с памятью);

– активные (интеллектуальные карточки).

Самыми распространенными являются пассивные карточки с магнитной полосой, которые считываются специальным устройством, имеющим клавиатуру и процессор. При использовании указанной карточки пользователь вводит свой идентификационный номер. В случае его совпадения с электронным вариантом, закодированным в карточке, пользователь получает доступ в систему. Это позволяет достоверно установить лицо, получившее доступ к си-

стеме и исключить несанкционированное использование карточки злоумышленником (например, при ее утере). Такой способ часто называют двукомпонентной аутентификацией.

Интеллектуальные карточки кроме памяти имеют собственный микропроцессор. Это позволяет реализовать различные варианты парольных методов защиты: многоразовые пароли, динамически меняющиеся пароли.

Методы аутентификации, основанные на измерении биометрических параметров человека, обеспечивают почти 100% идентификацию, решая проблемы утери или утраты паролей и личных идентификаторов. Однако эти методы нельзя использовать при идентификации процессов или данных (объектов данных), они только начинают развиваться, требуют пока сложного и дорогостоящего оборудования. Это обусловливает их использование пока только на особо важных объектах.

Примерами внедрения указанных методов являются системы идентификации пользователя по рисунку радужной оболочки глаза, по почерку, по тембру голоса и другие.

Новейшим направлением аутентификации является доказательство подлинности удаленного пользователя по его местонахождению. Данный защитный механизм основан на использовании системы космической навигации, типа *GPS* (*Global Positioning System*). Пользователь, имеющий аппаратуру *GPS*, многократно посылает координаты заданных спутников, находящихся в зоне прямой видимости. Подсистема аутентификации, зная орбиты спутников, может с точностью до метра определить месторасположение пользователя. Высокая надежность аутентификации определяется тем, что орбиты спутников подвержены колебаниям, предсказать которые достаточно трудно. Кроме того, координаты постоянно меняются, что исключает их перехват. Такой метод аутентификации может быть использован в случаях, когда авторизованный удаленный пользователь должен находиться в нужном месте.

Механизм идентификация и аутентификация пользователей

Общая процедура идентификации и аутентификации пользователя при его доступе в защищенную информационную систему заключается в следующем. Пользователь предоставляет системе свой личный идентификатор (например, вводит пароль или предоставляет палец для сканирования отпечатка). Далее система сравнивает полученный идентификатор со всеми хранящимися в ее базе идентификаторами. Если результат сравнения успешный, то пользователь получает доступ к системе в рамках установленных полномочий. В случае отрицательного результата система сообщает об ошибке и предлагает повторно ввести идентификатор. В тех случаях, когда пользователь превышает лимит возможных повторов ввода информации (ограничение на количество повторов является обязательным условием для защищенных систем) система временно блокируется и выдается сообщение о несанкционированных действиях (причем, может быть, и незаметно для пользователя).

Если в процессе аутентификации подлинность субъекта установлена, то система защиты информации должна определить его полномочия (совокупность прав). Это необходимо для последующего контроля и разграничения доступа к ресурсам.

В целом аутентификация по уровню информационной безопасности делится на три категории:

1. Статическая аутентификация.

2. Устойчивая аутентификация.

3. Постоянная аутентификация.

Первая категория обеспечивает защиту только от несанкционированных действий в системах, где нарушитель не может во время сеанса работы прочитать аутентификационную информацию. Примером средства статической аутентификации являются традиционные постоянные пароли. Их эффективность преимущественно зависит от сложности угадывания паролей и, собственно, от того, насколько хорошо они защищены.

Устойчивая аутентификация использует динамические данные аутентификации, меняющиеся с каждым сеансом работы. Реализациями устойчивой

аутентификации являются системы, использующие одноразовые пароли и электронные подписи. Устойчивая аутентификация обеспечивает защиту от атак, где злоумышленник может перехватить аутентификационную информацию и использовать ее в следующих сеансах работы.

Однако устойчивая аутентификация не обеспечивает защиту от активных атак, в ходе которых маскирующийся злоумышленник может оперативно (в течение сеанса аутентификации) перехватить, модифицировать и вставить информацию в поток передаваемых данных.

Постоянная аутентификация обеспечивает идентификацию каждого блока передаваемых данных, что предохраняет их от несанкционированной модификации или вставки. Примером реализации указанной категории аутентификации является использование алгоритмов генерации электронных подписей для каждого бита пересылаемой информации.

При санкционированном доступе в информационную систему пользователь должен идентифицировать себя, а система – проверить подлинность идентификации (произвести аутентификацию).

Идентификация – это присвоение какому-либо объекту или субъекту, реализующему доступ к ИС, уникального имени (логина), образа или числового значения. Установление подлинности (аутентификация) заключается в проверке, является ли данный объект (субъект) в самом деле тем, за кого себя выдает. Конечная цель идентификации и установления подлинности объекта в вычислительной системе – его допуск к информации ограниченного пользования в случае положительного результата проверки или отказ в допуске при отрицательном результате.

Как правило, любая процедура идентификации предполагает ввод пользователем своего логина (*login*) и пароля (*password*). В зависимости от особенностей функционирования системы пароль выбирается самим пользователем либо назначается администратором (или же его генерирует сама система).

Пароль должен быть таким, чтобы его нельзя было легко раскрыть. Для этого при выборе и использовании пароля рекомендуется руководствоваться следующими правилами:

– пароль не должен содержать личных данных пользователя (таких, как фамилия, имя, серия или номер паспорта либо другого документа, удостоверяющего личность, дата рождения, адрес;

– пароль не должен быть словом из какого-либо словаря (входить в какой-либо тезаурус), так как перебор слов заданного словаря – технически достаточно простая задача;

– пароль не должен быть слишком коротким (подобрать сочетание символов в этом случае также не представляет сложности);

– пароль не должен состоять из повторяющихся букв или фрагментов текста;

– пароль не должен состоять из символов, соответствующих подряд идущим клавишам на клавиатуре (например, «*QWERTY*» – образец недопустимого пароля);

– желательно включать в пароль символы в разных регистрах (прописные и строчные буквы, кириллицу и латиницу), знаки препинания, цифры и другие;

Меры предосторожности, которые необходимо соблюдать при использовании пароля:

– старайтесь сохранять пароль в тайне (лучше всего его запоминать, а не записывать;

– периодически (при регулярном обращении к системе – не реже одного раза в месяц) заменяйте пароль на новый, но он не должен выдаваться пользователю в конце сеанса работы. Заметим, что в разное время могут применяться различные пароли;

– в паспорте пользователя пароль должен храниться в зашифрованном виде. Наиболее подходящими для этих целей являются методы необратимого шифрования (при которых обратное преобразование невозможно). Введенный

пользователем пароль тоже должен шифроваться, а уже затем сравниваться с хранящимся.

Несоблюдение этих и ряда других правил ведет к раскрытию пароля и к возможности несанкционированного доступа к данным.

Среднее время безопасности пароля определяется по формуле

$$T = \left(d + \frac{m}{n}\right) \bullet \frac{S}{2}$$

где d – промежуток времени между двумя неудачными попытками несанкционированного входа в систему, m – количество символов в пароле, n – скорость набора пароля (количество символов, набираемых в единицу времени),S – количество всевозможных паролей указанной длины.

Таким образом, среднее время безопасности пароля фактически равно времени, за которое можно ввести (перебрать) половину всевозможных паролей заданной длины. Однако большинство информационных систем предусматривают возможность ввода идентифицирующих данных не более заданного количества раз (как правило не более трех раз за один сеанс работы).

В некоторых случаях процесс идентификации и аутентификации включает реализацию какого-либо несложного алгоритма. При этом после анализа логина и пароля система может, в частности, выдать на экран несколько значений данных указанного типа (например, сгенерированных чисел или последовательностей символов). Пользователь должен произвести с ними манипуляции в соответствии с некоторым алгоритмом (в простейшем случае в соответствии с заданной формулой). Система тоже производит указанные манипуляции с этими данными, а затем сверяет полученный результат с введенным пользователем.

Задание на лабораторную работу.

1. Изучить теоретическую часть по приведенным выше данным и дополнительной литературе.

2. С использованием одного из языков программирования составить программу, которая выполняет действия, указанные в таблице с номером вашего варианта.

Вариант 1.

Задание	**Алгоритм**
Пусть на экран выведены следующие три слова: «Sony», «Hewlett» и «Packard». Составить программу, которая записывает пароль следующим образом	Исходные данные – строковые константы 1. В строку <*результат*> в качестве первого символа записать букву, которая в алфавите стоит на месте, соответствующем сумме количеств символов в первом и третьем словах; если эта сумма больше 26, найти и использовать в качестве номера позиции искомой буквы в алфавите остаток от деления указанной суммы на 26. 2. В качестве второго символа записать букву, которая в алфавите предшествует букве, являющейся последним символом второго слова на экране; если это буква «a», записать «z». 3. Если третье слово содержит нечетное количество букв, то в качестве третьего символа записать букву, которая в алфавите следует за буквой, являющейся средним символом третьего слова; если это буква «z», записать «a». Если же третье слово содержит четное количество символов, то в качестве третьего символа записать букву, которая в алфавите предшествует букве, являющейся первым из двух средних символов третьего слова; если это буква «a», записать «z». 4. в качестве первого символа записать букву, которая в алфавите следует за буквой, являющейся первым символом первого слова на экране; если это буква «z», записать «a». 5. Вывести полученную строку.
Дополнить полученную программу средствами аутентификации	1. Ввести пароль пользователя. При вводе пароля пользователя обеспечить ввод пароля с отображением вместо каждого символа знаков «*». 2. Сравнить пароль пользователя с паролем, вычисленным ЭВМ. 3. Вывести результат аутентификации: пароль верен или неверен?
Пусть на экран выведены следующие три слова: «scleroses», «scoliosis», «paradantoz». Составить программу, которая записывает пароль следующим образом	Исходные данные — строковые константы 1. В строку <результат> в качестве первого символа записать букву, которая в алфавите следует за буквой, являющейся вторым символом первого слова на экране; если это буква «z», записать «a». 2. В качестве второго символа записать букву, которая в алфавите предшествует предпоследней букве, являющейся последним символом второго слова на экране; если это буква «a», записать «z». 3. Если третье слово содержит нечетное количество букв, то в качестве третьего символа записать букву, которая в алфавите следует за буквой, являющейся предшественником среднего символа третьего слова; если это буква «z», записать «a». Если же третье слово содержит четное количество символов, то в качестве третьего символа записать букву, которая в алфавите

	предшествует букве, являющейся первым из двух средних символов третьего слова; если это буква «a», записать «z». 4. В качестве четвертого символа записать букву, которая в алфавите стоит на месте, соответствующем сумме количеств символов в первом и третьем словах плюс 1 символ; если эта сумма больше 26, найти и использовать в качестве номера позиции искомой буквы в алфавите остаток от деления указанной суммы на 26. 5. Вывести полученную строку.
Дополнить полученную программу средствами аутентификации	1. Ввести пароль пользователя. При вводе пароля пользователя обеспечить ввод пароля с отображением вместо каждого символа знаков «*». 2. Сравнить пароль пользователя с паролем, вычисленным ЭВМ. 3. Вывести результат аутентификации: пароль верен или неверен?

Вариант 2.

Задание	**Алгоритм**
Пусть на экран выведены следующие три слова: «computer», «maus», «scanner». Составить программу, которая записывает пароль следующим образом	Исходные данные — строковые константы 1. В строку <результат> в качестве первого символа записать букву, которая в алфавите следует за буквой, являющейся последним символом первого слова на экране; если это буква «z», записать «a». 2. В качестве второго символа записать букву, которая в алфавите следует за буквой, являющейся последним символом второго слова на экране; если это буква «a», записать «z». 3. Если третье слово содержит нечетное количество букв, то в качестве третьего символа записать букву, которая в алфавите следует через пять позиций за буквой, являющейся средним символом третьего слова; если это буква «z», записать «a». Если же третье слово содержит четное количество символов, то в качестве третьего символа записать букву, которая в алфавите предшествует букве, являющейся первым из двух средних символов третьего слова; если это буква «a», записать «z». 4. В качестве четвертого символа записать букву, которая в алфавите стоит на месте, соответствующем сумме количеств символов в третьем и втором словах; если эта сумма больше 26, найти и использовать в качестве номера позиции искомой буквы в алфавите остаток от деления указанной суммы на 26. 5. Вывести полученную строку.
Дополнить полученную программу средствами аутентификации	1. Ввести пароль пользователя. При вводе пароля пользователя обеспечить ввод пароля с отображением вместо каждого символа знаков «*». 2. Сравнить пароль пользователя с паролем, вычисленным ЭВМ. 3. Вывести результат аутентификации: пароль верен или неверен?

Пусть на экран выведены следующие три слова: «mathematic», «physic», «hemi». Составить программу, которая записывает пароль следующим образом	Исходные данные — строковые константы 1. Если первое слово содержит нечетное количество букв, то в качестве первого символа в строку <результат> записать букву, которая в алфавите следует через три позиции за буквой, являющейся средним символом третьего слова; если это буква «z», записать «a». Если же первое слово содержит четное количество символов, то в качестве первого символа записать букву, которая в алфавите предшествует букве, являющейся первым из двух средних символов первого слова; если это буква «a», записать «z». 2. В качестве второго символа записать букву, которая в алфавите предшествует букве, являющейся последним символом второго слова на экране; если это буква «a», записать «z». 3. В качестве третьего символа записать букву, которая в алфавите следует за буквой, являющейся первым символом третьего слова на экране; если это буква «z», записать «a». 4. В качестве четвертого символа записать букву, которая в алфавите стоит на месте, соответствующем сумме количеств символов в первом и втором словах минус 1 символ; если эта сумма больше 26, найти и использовать в качестве номера позиции искомой буквы в алфавите остаток от деления указанной суммы на 26. 5. Вывести полученную строку.
Дополнить полученную программу средствами аутентификации	1. Ввести пароль пользователя. При вводе пароля пользователя обеспечить ввод пароля с отображением вместо каждого символа знаков «*». 2. Сравнить пароль пользователя с паролем, вычисленным ЭВМ. 3. Вывести результат аутентификации: пароль верен или неверен?

Контрольные вопросы

1. Что такое идентификация?
2. Что называется аутентификацией?
3. Правила выбора и использования пароля.
4. Что называется паролем пользователя?
5. Что представляют собой парольные системы аутентификации?
6. Особенности разделения привилегий и разграничения доступа?

Оглавление

Учебное издание

ЗАЩИТА ИНФОРМАЦИИ
Лабораторный практикум

Составители**: Попукайло Владимир Сергеевич, Шмелёва Анастасия Владимировна**
Компьютерная вертка: Головачук И. И.

Издается в авторской редакции

Подписано в печать 01.01.23.
Уч.-изд. л. 9,25. Электронное издание.

Опубликовано на Образовательном портале ПГУ им. Т.Г. Шевченко moodle@spsu.ru