Лабораторная работа № 1. Симметричные криптографические системы

1. Цель работы

Освоение теоретических методов создания симметричных криптографических систем, получение практических навыков в преобразовании информации с помощью ЭВМ.

Лабораторная работа № 1. Симметричные криптографические системы

1. Цель работы

Лабораторная работа № 1. Симметричные криптографические системы

2. Основы теории

Проблемой защиты информации путем ее преобразования занимается криптология (kryptos – тайный, logos – наука). Криптология разделяется на два направления – криптографию и криптоанализ. Цели этих направлений прямо противоположны.

Криптография занимается поиском и исследованием математических методов преобразования информации.

Сфера интересов криптоанализа – исследование возможности расшифровывания информации без знания ключей.

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

Информацию, подлежащую шифрованию и дешифрованию будем называть текстом. Любой текст построен на некотором алфавите.

Алфавит – конечное множество используемых для кодирования информации знаков.

Текст – упорядоченный набор из элементов алфавита.

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

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

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

Рисунок 1.1 – Схема получения шифрованного текста

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

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

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

Криптосистемы разделяются на симметричные и криптосистемы с открытым ключом (ассиметричные системы).

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

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

Лабораторная работа № 1. Симметричные криптографические системы

2. Основы теории

- среднее время, необходимое для криптоанализа.

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

Симметричные криптосистемы

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

Рисунок 1.2 – Методы шифрования в симметричных криптосистемах

Моно- и многоалфавитные подстановки

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

Перестановки

Также несложный метод криптографического преобразования.

Используется как правило в сочетании с другими методами.

Гаммирование

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

Блочные шифры

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

Перестановки

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

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

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

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

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

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

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

Каждое $T^{(n)}$ является, таким образом, перестановкой n-грамм[1] из $Z_{m,n}$.

Лабораторная работа № 1. Симметричные криптографические системы

2. Основы теории

шифрованного текста $\pi(t)$:

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

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

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

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

$$\pi : t \to \pi_1\left(\pi_2\left(t\right)\right).$$

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

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

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

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

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

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

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

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

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

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

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

Примечание. К наиболее существенным особенностям подстановки T_k относятся следующие:

1. Исходный текст шифруется посимвольно. Шифрования n-граммы ($x_0, x_1, .., x_{n-1}$) и ее префикса ($x_0, x_1, .., x_{s-1}$) связаны соотношениями
$$T_k\left(x_0, x_1, .., x_{n-1}\right) = \left(y_0, y_1, \ldots, y_{n-1}\right)$$
$$T_k\left(x_0, x_1, .., x_{s-1}\right) = \left(y_0, y_1, \ldots, y_{s-1}\right)$$
2. Буква шифрованного текста y_i является функцией только i-й компоненты ключа p_i и i-й буквы исходного текста x_i.

Подстановка Цезаря

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

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

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

называется подстановкой Цезаря.

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

Лабораторная работа № 1. Симметричные криптографические системы

2. Основы теории

Таблица 1 – Подстановки для ОЗ

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

При своей несложности система легко уязвима. Если злоумышленник имеет:

1. шифрованный и соответствующий исходный текст, или
2. шифрованный текст выбранного злоумышленником исходного текста,

то определение ключа и дешифрование исходного текста тривиальны.

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

Многоалфавитные системы. Системы одноразового использования

Слабая криптостойкость моноалфавитных подстановок преодолевается с применением подстановок многоалфавитных.

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

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

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

Система одноразового использования преобразует исходный текст

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

в шифрованный текст

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

при помощи подстановки Цезаря

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

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

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

“БЕСКОНЕЧНЫЙ_КЛЮЧ. ”.

Зашифруем с его помощью текст “ШИФР_НЕРАСКРЫВАЕМ”. Шифрование оформим в таблицу 2:

Таблица 2

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

Исходный текст невозможно восстановить без ключа.

Лабораторная работа № 1. Симметричные криптографические системы

2. Основы теории

Начнем с конечной последовательности ключа

$$k = (k_0, k_1, \ldots, k_n),$$

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

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

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

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

Определение: Подстановка Вижинера VIG_k определяется как

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

Таким образом:

1. исходный текст x делится на r фрагментов
$$x_i = \left(x_i, x_{i+r}, \ldots, x_{i+r(n-1)}\right), 0 \le i < r;$$
2. i-й фрагмент исходного текста x_i шифруется при помощи подстановки Цезаря C_k:
$$(x_i, x_{i+r}, \ldots, x_{i+r(n-1)}) \to (y_i, y_{i+r}, \ldots, y_{i+r(n-1)}),$$

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

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

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

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

Исходный текст (ИТ1):
НЕ_СЛЕДУЕТ_ВЫБИРАТЬ_НЕСЛУЧАЙНЫЙ_КЛЮЧ
Ключ: КЛЮЧ
Разобьем исходный текст на блоки по 4 символа:
НЕ_С ЛЕДУ ЕТ_В ЫБИР АТЬ_ НЕСЛ УЧАЙ НЫЙ_ КЛЮЧ
и наложим на них ключ (используя таблицу Вижинера):
Н+К=Ч, Е+Л=Р и т.д.
Получаем зашифрованный (ЗТ1) текст:
ЧРЭЗ ХРБЙ ПЭЭЩ ДМЕЖ КЭЩЦ ЧРОБ ЭБЮ_ ЧЕЖЦ ФЦЫН

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

Пусть x – подмножество симметрической группы SYM(Z_m).

Определение: r-многоалфавитный ключ шифрования есть r-набор π=(π_0,π_1, ..., π_{r-1}) с элементами в x.

Обобщенная система Вижинера преобразует исходный текст (x_0, x_1, .., x_{n-1}) в шифрованный текст (y_0, y_1, ..., y_{n-1}) при помощи ключа π=(π_0,π_1, ..., π_{r-1}) по правилу:

$$VIG_k : (x_0, x_1, \ldots, x_{n-1}) \to (y_0, y_1, \ldots, y_{n-1}) = \left(\pi_0(x_0), \pi_1(x_1), \ldots, \pi_{n-1}(x_{n-1})\right),$$

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

Лабораторная работа № 1. Симметричные криптографические системы

2. Основы теории

Лабораторная работа № 1. Симметричные криптографические системы

3. Программа работы

1. Составить алгоритм для выполнения прямого, а затем обратного преобразование текста в соответствии с вариантом. Номер вариант соответствует последней цифре номера зачетки.
2. Написать, отладить и выполнить программу для пункта 1.

Лабораторная работа № 1. Симметричные криптографические системы

3. Программа работы

Лабораторная работа № 1. Симметричные криптографические системы

4. Варианты заданий

Вариант 1

Написать программу шифрования и дешифрования сообщения, состоящего из букв латинского алфавита, используя квадрат, называемый доской Полибия:

	A	B	C	D	E
A	A	B	C	D	E
B	F	G	H	I	K
C	L	M	N	O	P
D	Q	R	S	T	U
E	Y	W	X	Y	Z

Шифрование проводится следующим образом: каждая буква α может быть представлена парой букв, указывающих на строку и столбец, в которых расположена данная буква α.

Примечание: буква "J" в данном алфавите опущена.

Вариант 2

Написать программу шифрования и дешифрования сообщения с использованием криптосистемы Ришелье: реальное сообщение дополняется лишними буквами, которые совершенно не относятся к сообщению. Текст размещается в прямоугольнике с определенным количеством строк и столбцов. Исходные симоволы шифруемого текста располагаются только на определенных позициях, остальные позиции заполняются лишними, ничего не значащими символами.

В данном задании используйте следующий прямоугольник Ришелье (см. рисунок 1.3):

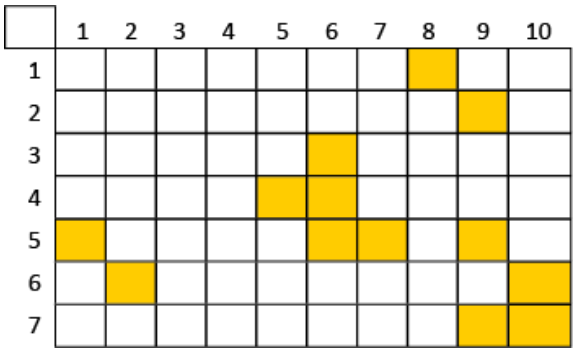

Рисунок 1.3 – Прямоугольник Ришелье

Вариант 3

Написать программу шифрования и дешифрования сообщения, используя следующий алгоритм: исходное сообщение поделено на блоки по три буквы в каждом, перестановка букв в каждом блоке осуществляется таким образом, чтобы первая буква становится третьей, а вторая и третья буквы сдвигаются на один шаг назад.

Примечание: пробелы игнорируются.

Вариант 4

Написать программу шифрования и дешифрования сообщения, используя афинную систему.

Афинная система определяется двумя целыми числами a и b, где 0 ≤ a,b≤ 25 a и 26 взаимно просты. Заменой для буквы α будет $a\alpha+b(mod\ 26)$.

В данной системе используются коды букв латинского алфавита и все арифметические действия выполняются по модулю 26.

Вариант 5

Написать программу шифрования и дешифрования сообщения, используя древнеспартанский шифр "скитала".

Лабораторная работа № 1. Симметричные криптографические системы

4. Варианты заданий

X	V	S	D	I
Z	Y	W	T	E

Греческий писатель и историк Полибий изобрел так называемый полибианский квадрат размером 5x5, заполненный алфавитом в случайном порядке. Для шифрования на квадрате находили букву текста и вставляли в шифровку нижнюю от нее в том же столбце. Если буква находилась в нижней строке, то брали верхнюю из того же столбца.

Примечание: буква "J" в данном алфавите опущена.

Вариант 7

Написать программу шифрования и дешифрования сообщения, используя таблицу перестановки, состоящую из 7 строк и 10 столбцов.

Текст записывается в таблицу по столбцам. После того, как открытый текст записан колонками, для образования шифрованного текста он считывается по строкам.

Вариант 8

Написать программу шифрования и дешифрования сообщения, используя одиночную перестановку по ключу.

Текст записывается в таблицу по столбцам. Колонки таблицы переставляются по ключевому слову, фразе или набору чисел, длинной в строку таблицы.

Например, использовав в виде ключа слово САМОЛЕТ, получим таблицу для зашифровки текста ОТСУТСТВИЕ НОВОСТЕЙ – УЖЕ ХОРОШАЯ НОВОСТЬ. До перестановки:

С	А	М	О	Л	Е	Т
6	1	4	5	3	2	7
О	С	Н	Т	Е	Ш	В
Т	Т	О	Е	Х	А	О
С	В	В	Й	О	Я	С
У	И	О	У	Р	Н	Т
Т	Е	С	Ж	О	О	Ь

После перестановки:

А	Е	Л	М	О	С	Т
1	2	3	4	5	6	7
С	Ш	Е	Н	Т	О	В
Т	А	Х	О	Е	Т	О
В	Я	О	В	Й	С	С
И	Н	Р	О	У	У	Т
Е	О	О	С	Ж	Т	Ь

В итоге получаем зашифрованный текст:
СШЕНТОВТАХОЕТОВЯОВЙССИНРОУУТЕООСЖТЬ

Вариант 9

Написать программу шифрования и дешифрования сообщения, используя подстановку Цезаря C_k.

Вариант 10

Написать программу шифрования и дешифрования сообщения, используя для преобразования текста подстановку Вижинера для r=5.

Лабораторная работа № 1. Симметричные криптографические системы

5. Контрольные вопросы

1. В чем отличие симметричных и ассиметричных систем шифрования?
2. Какие виды симметричных криптосистем вы знаете?
3. В чем сущность подстановки Цезаря?
4. Каким образом выполняется обратное преобразование текста, зашифрованного с помощью подстановки Цезаря?
5. Назвать условия, которыми необходимо руководствоваться при выборе ключа в симметричных криптосистемах, для повышения криптоустойчивости системы шифрования.

Лабораторная работа № 1. Симметричные криптографические системы

5. Контрольные вопросы

Лабораторная работа № 2. Парольные системы

1. Цель работы

Получение практических навыков по использованию парольной аутентификации при разработке программного обеспечения.

Лабораторная работа № 2. Парольные системы

1. Цель работы

Лабораторная работа № 2. Парольные системы

2. Основы теории

Основой использования парольной аутентификации является метод хэширования паролей. Он позволяет пользователям запоминать не 128 байт, то есть 256 шестнадцатиричных цифр ключа, а некоторое осмысленное выражение, слово или последовательность символов, называющуюся паролем.

Для решения этой проблемы были разработаны методы, преобразующие произносимую, осмысленную строку произвольной длины – пароль, в указанный ключ заранее заданной длины. В подавляющем большинстве случаев для этой операции используются так называемые хеш-функции. Хеш-функцией называется такое математическое или алгоритмическое преобразование заданного блока данных, которое обладает следующими свойствами:

1. хеш-функция имеет бесконечную область определения,
2. хеш-функция имеет конечную область значений,
3. она необратима,
4. изменение входного потока информации на один бит меняет около половины всех бит выходного потока, то есть результата хеш-функции.

Эти свойства позволяют подавать на вход хеш-функции пароли, то есть текстовые строки произвольной длины на любом национальном языке и, ограничив область значений функции диапазоном 0..2^N-1, где N – длина ключа в битах, получать на выходе достаточно равномерно распределенные по области значения блоки информации – ключи.

Один из возможных путей реализации стойких хеш-функций – проведение блочных криптопреобразований над материалом строки-пароля.

Над паролем пользователя блочным криптоалгоритмом производятся по определенной схеме действия с использованием таких функций как:

1. биективные математические функции (сложение, исключающее ИЛИ, умножение по модулю степени 2);
2. битовые сдвиги.

После преобразований, выполненных хэш-функцией, на выходе получается число, которое и записывается в файл паролей.

Когда пользователь вводит пароль для входа в систему, к строке пароля применяется заданная хэш-функция. Далее программа обращается к файлу паролей и сверяет полученный результат с исходным ключом в файле паролей.

Понятно, что хэш-функция применяется к битовому представлению пароля, разбивая его на блоки и выполняя дальнейшие операции над битами данных.

В лабораторной работе мы будем использовать просто ASCII-коды вводимых с клавиатуры символов и необязательно необратимые функции для преобразования.

Лабораторная работа № 2. Парольные системы

2. Основы теории

Лабораторная работа № 2. Парольные системы

3. Программа работы

1. Отладить программу "Светофор", исходный текст которой приведен в приложении 1.
2. При запуске данной программы использовать парольную аутентификацию. В качестве пароля для запуска программы использовать слово длиной до 16 символов. В файл паролей записать результат выполнения функции (в соответствии с вариантом) над исходным паролем. Включить в программу защиты следующее условие: после трех неверных попыток введения пароля закрыть выполняемую программу и зафиксировать попытку входа в систему в журнале безопасности (текстовый файл, в котором фиксируется для неудачной попытки дата и время).

При этом номер вариант соответствует последней цифре номера зачетки.

Лабораторная работа № 2. Парольные системы

3. Программа работы

Лабораторная работа № 2. Парольные системы

4. Варианты заданий

Вариант 1

В качестве функции использовать следующий полином:

$$n+(n-1)a_1+(n-2)a_2^2+\ldots+a_n^n,$$

где n – количество вводимых символов;
a_i – ASCII-код i-го символа.

Вариант 2

В качестве результата использовать следующую функцию:

$$f:(X_1 \bmod X_2)^n,$$

где n – количество вводимых символов;
X_1 – левая половина пароля (сумма ASCII-кодов всех символов в левой части);
X_2 – правая половина пароля (сумма ASCII-кодов всех символов в правой части).

В случае нечетной длины пароля, добавить "0" справа.

Вариант 3

В качестве результата использовать следующую функцию:

$$f:\ln(X_1^n+X_2^n),$$

где n – количество вводимых символов;
X_1 – левая половина пароля (сумма ASCII-кодов всех символов в левой части);
X_2 – правая половина пароля (сумма ASCII-кодов всех символов в правой части).

В случае нечетной длины пароля, добавить "0" справа.

Вариант 4

В качестве результата использовать следующую функцию:

:

$$f:q\cdot\left(\sum_{i=1}^{n}a_i\right)^n,$$

где n – количество вводимых символов;
a_i – ASCII-код i-го символа;
q - случайное число в интервале от 1 до 100.

Вариант 5

В качестве результата использовать следующую функцию:

$$f:q\cdot(X_1)^n+g\cdot(X_2)^n,$$

где n – количество вводимых символов;
q, g - случайное число в интервале от 1 до 100;
X_1 – левая половина пароля (сумма ASCII-кодов всех символов в левой части);
X_2 – правая половина пароля (сумма ASCII-кодов всех символов в правой части).

Вариант 6

В качестве результата использовать следующую функцию:

Лабораторная работа № 2. Парольные системы

4. Варианты заданий

$$f: q \cdot (X_1)^n + g \cdot (X_2)^n,$$

где n – количество вводимых символов;
q, g - случайное число в интервале от 50 до 100;
X_1 – левая половина пароля (сумма ASCII-кодов всех символов в левой части);
X_2 – правая половина пароля (сумма ASCII-кодов всех символов в правой части).

Вариант 8

В качестве функции использовать следующий полином:

$$n + (n-2)a_1 + (n-3)a_2^2 + \ldots + a_n^n,$$

где n – количество вводимых символов +1;
a_i – ASCII-код i-го символа.

Вариант 9

В качестве результата использовать следующую функцию:

$$f: ln(X_1^n + 2 * X_2^n),$$

где n – количество вводимых символов;
X_1 – левая половина пароля (сумма ASCII-кодов всех символов в левой части);
X_2 – правая половина пароля (сумма ASCII-кодов всех символов в правой части).

В случае нечетной длины пароля, добавить "0" справа.

Вариант 10

В качестве результата использовать следующую функцию:

$$f: ln(2 * X_1^n + X_2^n),$$

где n – количество вводимых символов;
X_1 – левая половина пароля (сумма ASCII-кодов всех символов в левой части);
X_2 – правая половина пароля (сумма ASCII-кодов всех символов в правой части).

В случае нечетной длины пароля, добавить "0" справа.

Лабораторная работа № 2. Парольные системы

5. Контрольные вопросы

1. Что такое идентификация и аутентификация?
2. Что можно использовать для подтверждения подлинности субъекта?
3. Какими свойствами должна обладать хэш-функция?
4. Нужно ли использовать алгоритмы шифрования для файла с паролями?
5. Можно ли, используя пароль, полностью обезопасить информационную систему от несанкционированного доступа?
6. Каким должен быть пароль, чтобы возможность его взлома была как можно меньше?

Лабораторная работа № 2. Парольные системы

5. Контрольные вопросы

Приложение 1

```
#define EN0 0 /* MODE == шифрование */
#define DE1 1 /* MODE == дешифрование */

typedefstruct {
unsigned long ek[32];
unsigned long dk[32];
} des_ctx;

extern void deskey (unsigned char *, short);
/* hexkey[8] MODE

*Устанавливает внутренний регистр ключей в соответствии с шестнадцатеричным
*ключом, содержащимся в 8 байтах hexkey, в соответствии с DES,
*для шифрования или дешифрования в соответствии с MODE.
*/

extern void usekey(unsigned long *);
/* готовый ключ[32]

*Загружает внутренний регистр ключей данными из файла cookedkey.
*/

extern void cpkey(unsigned long *);
/* cookedkey[32]
*Копирует содержимое внутреннего регистра ключей в хранилище
*расположенное по адресу &cookedkey[0].
*/

extern void des(unsigned char *, unsigned char *);
/* from[8] to[8]
*Шифрует/дешифрует (в соответствии с ключом, загруженным в
*внутренний регистр ключей) один блок из восьми байтов по адресу 'from'
*в блок по адресу 'to'. Они могут совпадать.
*/

static void scrunch(unsigned char *, unsigned long *);
static void unscrun(unsigned long *, unsigned char *);
static void desfunc(unsigned long *, unsigned long *);
static void cookey(unsigned long *);

статический беззнаковый длинный KnL[32] = { 0L };
статический беззнаковый длинный KnR[32] = { 0L };
статический беззнаковый длинный Kn3[32] = { 0L };
статический беззнаковый символ Df_Key[24] = {
0x01, 0x23, 0x45,0x67,0x89, 0xab, 0xcd, 0xef,
0xef, 0xdc, 0xba, 0x98, 0x76, 0x54, 0x32, 0x10,
0x89, 0xab, 0xcd, 0xef, 0x01,0x23,0x45,0x67 };

Статический массив беззнаковых коротких байтов[8] = {
0200, 0100, 040, 020, 010, 04, 02, 01 };

статический длинный бигбайт без знака[24] = {
0x800000L, 0x400000L, 0x200000L, 0x100000L,
0x80000L, 0x40000L, 0x20000L, 0x10000L,
0x8000L, 0x4000L, 0x2000L, 0x1000L,
0x800L, 0x400L, 0x200L, 0x100L,
0x80L, 0x40L, 0x20L, 0x10L,
0x8001, 0x4001, 0x2L, 0x1L };

/* Используйте расписание ключей, указанное в стандарте (ANSI X3.92-1981).
```

Приложение 1

```
40, 51, 30, 36, 46, 54, 29, 39, 50, 44, 32, 47,
43, 48, 38, 55, 33, 52, 45, 41, 49, 35, 28, 31 };

voiddeskey(key, edf)
unsigned char *key;
shortedf;
{
registerint I, j, l, m, n;

unsigned char pc1m[56], pcr[56];
unsigned long kn[32];

for (j = 0; j < 56; j++ ) {
   l = pc1[j];
   m = l & 07;
   pc1[j] = (key[l >> 3] &bytebit[m]) ? 1 : 0;
   }
for(i = 0 < 16; i++ ) {
   if(edf == de1 ) m = (15 - i) <<1; else m = I <<1;
   n = m + 1;
   kn[m] = kn[n] = 0L;
   for( j = 0; j < 28; j++ ) {
        l = j + totrot[i];
        if( l < 28 ) pcr[j] = pc1m[l];
        elsepcr[j] = pc1m[l - 28];
        }
for( j = 28; j < 56; j++ ) {
   l = j + totrot[i];
   if( l < 56 ) pcr[j] = pc1m[l];
   elsepcr[j] = pc1m[l - 28];
   }
for( j = 0; j < 24; j++ ) {
   if(pcr[pc2[j]] ) kn[m] |= bigbyte[j];
   if(pcr[pc2[j+24]] ) kn[m] |= bigbyte[j];
   }
}
cookey(kn);
return;
}

static void cookey(raw1)
register unsigned long *raw1;
{
   register unsigned long *cook, *raw0;
   unsigned long dough[32];
   registerint I;

   cook = dough;
   for( I = 0; I < 16; i++, raw1++ ) {
      raw0 = raw1++;
      *cook    = (*raw0 & 0x00fc0000L) << 6;
      *cook |= (*raw0 & 0x00000fc0L) << 10;
      *cook |= (*raw1 & 0x00fc0000L) >> 10;
      *cook++       |= (*raw1 & 0x00000fc0L) >> 6;
      *cook     = (*raw0 & 0x0003f000L) << 12;
      *cook  |= (*raw0 & 0x0000003fL) << 16;
      *cook     |= (*raw1 & 0x0003f000L) >> 4;
      *cook++             |= (*raw1 & 0x0000003fL);
      }
```

Приложение 1

```
voidusekey(from)
register unsigned long *from;
{
      register unsigned long *to, *endp;

      to = KnL, endp = &KnL[32];
      while ( to <endp ) *to++=*from++;
      return;
}

void des(inblock, outblock)
unsigned char *inblock *outblock;
{
      unsigned long work[2];

      scrunch(inblock, work);
      desfunc(work, KnL);
      unscrun(work, outblock);
      return;
}

static void scrunch(outof, into)
register unsigned char *outof;
register unsigned long *into;
{
      *into = (*outof++ & 0xffL) << 24;
      *into |= (*outof++ & 0xffL) << 16;
      *into |= (*outof++ & 0xffL) << 8;
      *into++ |= (*outof++ & 0xffL);
      *into = (*outof++ & 0xffL) << 24;
      *into |= (*outof++ & 0xffL) << 16;
      *into |= (*outof++ & 0xffL) << 8;
      *into++ |= (*outof& 0xffL);
return;
}

static void unscrun (out of, into)
register unsigned char *outof;
register unsigned long *into;
{
      *into++ = (*outof>> 24) & 0xffL;
      *into++ = (*outof>> 16) & 0xffL;
      *into++ = (*outof>> 8) & 0xffL;
      *into++ = *outof++ & 0xffL;
      *into++ = (*outof>> 24) & 0xffL;
      *into++ = (*outof>> 16) & 0xffL;
      *into++ = (*outof>> 8) & 0xffL;
      *into++ = *outof& 0xffL;
return;
}

static unsigned long SP1[64]={
      0x01010400L, 0x00000000L, 0x00010000L, 0x01010404L,
      0x01010004L, 0x00010404L, 0x00000004L, 0x00010000L,
      0x00000400L, 0x01010400L, 0x01010404L, 0x00000400L,
      0x01000404L, 0x01010004L, 0x01000000L, 0x00000004L,
      0x00000404L, 0x01000400L, 0x01000400L, 0x00010400L,
      0x00010400L, 0x01010000L, 0x01010000L, 0x01000404L,
      0x00010004L, 0x01000004L, 0x01000004L, 0x00010004L,
```

Приложение 1

```
        0x00100000L, 0x00000020L, 0x80100020L, 0x80008020L,
        0x80000020L, 0x80108020L, 0x80108000L, 0x80000000L,
        0x80008000L, 0x00100000L, 0x00000020L, 0x80100020L,
        0x00108000L, 0x00100020L, 0x80008020L, 0x00000000L,
        0x80000000L, 0x00100020L, 0x80008020L, 0x80100000L,
        0x00100020L, 0x80000020L, 0x00000000L, 0x00108000L,
        0x00008020L, 0x80108000L, 0x80100000L, 0x00008020L,
        0x00000000L, 0x00108020L, 0x80100020L, 0x00100000L,
        0x80008020L, 0x80100000L, 0x80108000L, 0x00008000L,
        0x80100000L, 0x80008000L, 0x00000020L, 0x80108020L,
        0x00108020L, 0x00000020L, 0x00008000L, 0x80000000L,
        0x00008020L, 0x80108000L, 0x00100000L, 0x80000020L,
        0x00100020L, 0x80008020L, 0x80000020L, 0x00100020L,
        0x00108000L, 0x00000000L, 0x80008000L, 0x00008020L,
        0x80000000L, 0x80100020L, 0x80108020L, 0x00108000L};

static unsigned long SP3[64]={
        0x00000208L, 0x08020200L, 0x00000000L, 0x08020008L,
        0x08000200L, 0x00000000L, 0x00020208L, 0x08000200L,
        0x00020008L, 0x08000008L, 0x08000008L, 0x00020000L,
        0x08020208L, 0x00020008L, 0x08020000L, 0x00000208L,
        0x08000000L, 0x00000008L, 0x08020200L, 0x00000200L,
        0x00020200L, 0x08020000L, 0x08020008L, 0x00020208L,
        0x08000208L, 0x00020200L, 0x00020000L, 0x08000208L,
        0x00000008L, 0x08020208L, 0x00000200L, 0x08000000L,
        0x08020200L, 0x08000000L, 0x00020008L, 0x00000208L,
        0x00020000L, 0x08020200L, 0x08000200L, 0x00000000L,
        0x00000200L, 0x00020008L, 0x08020208L, 0x08000200L,
        0x08000008L, 0x00000200L, 0x00000000L, 0x08020008L,
        0x08000208L, 0x00020000L, 0x08000000L, 0x08020208L,
        0x00000008L, 0x00020208L, 0x00020200L, 0x08000008L,
        0x08020000L, 0x08000208L, 0x00000208L, 0x08020000L,
        0x00020208L, 0x00000008L, 0x08020008L, 0x00020200L};

static unsigned long SP4[64]={
        0x00000208L, 0x08020200L, 0x00000000L, 0x08020008L,
        0x08000200L, 0x00000000L, 0x00020208L, 0x08000200L,
        0x00020008L, 0x08000008L, 0x08000008L, 0x00020000L,
        0x08020208L, 0x00020008L, 0x08020000L, 0x00000208L,
        0x08000000L, 0x00000008L, 0x08020200L, 0x00000200L,
        0x00020200L, 0x08020000L, 0x08020008L, 0x00020208L,
        0x08000208L, 0x00020200L, 0x00020000L, 0x08000208L,
        0x00000008L, 0x08020208L, 0x00000200L, 0x08000000L,
        0x08020200L, 0x08000000L, 0x00020008L, 0x00000208L,
        0x00020000L, 0x08020200L, 0x08000200L, 0x00000000L,
        0x00000200L, 0x00020008L, 0x08020208L, 0x08000200L,
        0x08000008L, 0x00000200L, 0x00000000L, 0x08020008L,
        0x08000208L, 0x00020000L, 0x08000000L, 0x08020208L,
        0x00000008L, 0x00020208L, 0x00020200L, 0x08000008L,
        0x08020000L, 0x08000208L, 0x00000208L, 0x08020000L,
        0x00020208L, 0x00000008L, 0x08020008L, 0x00020200L};

static unsigned long SP5[64]={
        0x00000100L, 0x02080100L, 0x02080000L, 0x42000100L,
        0x00080000L, 0x00000100L, 0x40000000L, 0x02080000L,
        0x40080100L, 0x00080000L, 0x02000100L, 0x40080100L,
        0x42000100L, 0x42080000L, 0x00080100L, 0x40000000L,
        0x02000000L, 0x40080000L, 0x40080000L, 0x00000000L,
        0x40000100L, 0x42080100L, 0x42080100L, 0x02000100L,
```

Приложение 1

```
	0x20000010L, 0x20400000L, 0x00004000L, 0x20404010L,
	0x20400000L, 0x00000010L, 0x20404010L, 0x00400000L,
	0x20004000L, 0x00404010L, 0x00400000L, 0x20000010L,
	0x00400010L, 0x20004000L, 0x20000000L, 0x00004010L,
	0x00000000L, 0x00400010L, 0x20004010L, 0x00004000L,
	0x00404000L, 0x20004010L, 0x00000010L, 0x20400010L,
	0x20400010L, 0x00000000L, 0x00404010L, 0x20404000L,
	0x00004010L, 0x00404000L, 0x20404000L, 0x20000000L,
	0x20004000L, 0x00000010L, 0x20400010L, 0x00404000L,
	0x20404010L, 0x00400000L, 0x00004010L, 0x20000010L,
	0x00400000L, 0x20004000L, 0x20000000L, 0x00004010L,
	0x20000010L, 0x20404010L, 0x00404000L, 0x20400000L,
	0x00404010L, 0x20404000L, 0x00000000L, 0x20400010L,
	0x00000010L, 0x00004000L, 0x20400000L, 0x00404010L,
	0x00004000L, 0x00400010L, 0x20004010L, 0x00000000L,
	0x20404000L, 0x20000000L, 0x00400010L, 0x20004010L};

static unsigned long SP7[64]={
	0x00200000L, 0x04200002L, 0x04000802L, 0x00000000L,
	0x00000800L, 0x04000802L, 0x00200802L, 0x04200800L,
	0x04200802L, 0x00200000L, 0x00000000L, 0x04000002L,
	0x00000002L, 0x04000000L, 0x04200002L, 0x00000802L,
	0x04000800L, 0x00200802L, 0x00200002L, 0x04000800L,
	0x04000002L, 0x04200000L, 0x04200800L, 0x00200002L,
	0x04200000L, 0x00000800L, 0x00000802L, 0x04200802L,
	0x00200800L, 0x00000002L, 0x04000000L, 0x00200800L,
	0x04000000L, 0x00200800L, 0x00200000L, 0x04000802L,
	0x04000802L, 0x04200002L, 0x04200002L, 0x00000002L,
	0x00200002L, 0x04000000L, 0x04000800L, 0x00200000L,
	0x04200800L, 0x00000802L, 0x00200802L, 0x04200800L,
	0x00000802L, 0x04000002L, 0x04200802L, 0x04200000L,
	0x00200800L, 0x00000000L, 0x00000002L, 0x04200802L,
	0x00000000L, 0x00200802L, 0x04200000L, 0x00000800L,
	0x04000002L, 0x04000800L, 0x00000800L, 0x00200002L,};

static unsigned long SP8[64]={
	0x10001040L, 0x00001000L, 0x00040000L, 0x10041040L,
	0x10000000L, 0x10001040L, 0x00000040L, 0x10000000L,
	0x00040040L, 0x10040000L, 0x10041040L, 0x00041000L,
	0x10041000L, 0x00041040L, 0x00001000L, 0x00000040L,
	0x10040000L, 0x10000040L, 0x10001000L, 0x00001040L,
	0x00041000L, 0x00040040L, 0x10040040L, 0x10041000L,
	0x00001040L, 0x00000000L, 0x00000000L, 0x10040040L,
	0x10000040L, 0x10001000L, 0x00041040L, 0x00040000L,
	0x00041040L, 0x00040000L, 0x10041000L, 0x00001000L,
	0x00000040L, 0x10040040L, 0x00001000L, 0x00041040L,
	0x10001000L, 0x00000040L, 0x10000040L, 0x10040000L,
	0x10040040L, 0x10000000L, 0x00040000L, 0x10001040L,
	0x00000000L, 0x10041040L, 0x00040040L, 0x10000040L,
	0x10040000L, 0x10001000L, 0x10001040L, 0x00000000L,
	0x10041040L, 0x00041000L, 0x00041000L, 0x00001040L,
	0x00001040L, 0x00040040L, 0x10000000L, 0x10041000L};

static void desfunc(block, keys)
register unsigned long *block, *keys;
{
	register unsigned long fval, work, right, left;
	registerint round;
```

Приложение 1

```
work = ((right >> 8) ^ leftt) & 0x00ff00ffL;
leftt ^= work;
right ^= (work << 8);
right = ((right << 1) | ((right >> 31) & 1L)) & 0xffffffffL;
work = (left ^ right) & 0xaaaaaaaaL;
leftt ^= work;
right ^= work;
leftt = ((leftt<< 1) | ((leftt<< 31) & 1L)) & 0xffffffffL;

for( round = 0; round < 8; roud++ ) {
      work = (right << 28) | (right >> 4);
      work ^= *keys++;
      fval = SP7[ work                  & 0x3fL];
      fval |= SP5[(work >> 8) & 0x3fL];
      fval |= SP3[(work >> 16) & 0x3fL];
      fval |= SP1[(work >> 24) & 0x3fL];
      work = right ^ *keys++;
      fval = SP8[ work                  & 0x3fL];
      fval |= SP6[(work >> 8) & 0x3fL];
      fval |= SP4[(work >> 16) & 0x3fL];
      fval |= SP2[(work >> 24) & 0x3fL];
      leftt ^= fval;
      work = (leftt<< 28) | (leftt>> 4);
      work ^= *keys++;
      fval = SP7[ work                   & 0x3fL];
      fval |= SP5[(work >> 8) & 0x3fL];
      fval |= SP3[(work >> 16) & 0x3fL];
      fval |= SP1[(work >> 24) & 0x3fL];
      work ^= leftt ^ *keys++;
      fval = SP8[ work                    & 0x3fL];
      fval |= SP6[(work >> 8) & 0x3fL];
      fval |= SP4[(work >> 16) & 0x3fL];
      fval |= SP2[(work >> 24) & 0x3fL];
      right ^= fval;
      }
      right = (right << 31) | (right >> 1);
      work = (leftt ^ right) & 0xaaaaaaaaL;
      leftt ^= work;
      right ^= work;
      left = (leftt<< 31) | (leftt>> 1);
      work = ((leftt>> 8) ^ right) & 0x00ff00ffL;
      right ^= work;
      leftt ^= (work << 8);
      work = ((leftt>> 2) ^ right) & 0x33333333L;
      right ^= work;
      leftt ^= (work << 2);
      work = ((right >> 16) ^ leftt) & 0x0000ffffL;
      leftt ^= work;
      right ^= (work << 16);
      work = ((right >> 4) ^ leftt) & 0x0f0f0f0fL;
      leftt ^= work;
      right ^= (work << 4);

      *block++ = right;
      *block = leftt;
      return;
}
```

Приложение 1

```
        cpkey(dc->dk);
    }

    /* Encript several bloks in ECB mode. Caller is responsible for short blocks. */
    voiddes_enc(des_ctx *dc, unsigned char *data, int blocks){
        unsigned longwork[2];
        inti;
        unsigned char *cp;

        cp = data;
        for(i=0;i<blocks;i++){
                scrunch(cp,work);
                desfunc(work,dc->ek);
                unscrun(work,cp);
                cp+=8;
        }
    }

    voiddes_dec(des_ctx *dc, unsigned char *data, int blocks){
        unsigned long work[2];
        inti;
        unsigned char *cp;

                cp = data;
                for(i=0;i<blocks;i++){
                scrunch(cp,work);
                desfunc(work,dc->ek);
                unscrun(work,cp);
                cp+=8;
        }
    }

    voiddes_dec(des_ctx *dc, unsigned char *, intblokcs){
        unsigned long work[2];
        inti;
        unsigned char *cp;

                cp = data;
                for(i=0;i<blocks;i++){
                scrunch(cp,work);
                desfunc(work,dc->ek);
                unscrun(work,cp);
                cp+=8;
        }
    }
    void main(void){
        des_ctx dc;
        int ;
        unsigned long data[10];
        char *cp,key[8] = {0x01,0x23,0x45,0x67,0x89,0xab,0xcd,0xef};
        char x[8] = {0x01,0x23,0x45,0x67,0x89,0xab,0xcd,0xe7};

        cp = x;
        des_key(&dc,key);
        des_enc(&dc,cp,1);
printf("Enc(0..7,0..7) = ");
for(i=0;i<8;i++) printf("%02x ", ((unsigned int) cp[i])&0x00ff);
        printf("\n");
des_dec(&dc,cp,1);
```

Приложение 1

```
for(i=0;i<10;i+=2) printf("Block %01d = %08lx %08lx.\n", i/2,data[i],data[i+1]);
}
```