Документ подписан простой электронной подписью
Сертификат: 03489EA30058B4708
Владелец: "АНО ВО ""РОССИЙСКИЙ НОВЫЙ УНИВЕРСИТЕТ"
Действителен: с 28.05.2026 по 28.08.2027

АВТОНОМНАЯ НЕКОММЕРЧЕСКАЯ ОРГАНИЗАЦИЯ ВЫСШЕГО ОБРАЗОВАНИЯ
«РОССИЙСКИЙ НОВЫЙ УНИВЕРСИТЕТ»
(АНО ВО «РосНОУ»)

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

ПО ВЫПОЛНЕНИЮ ПРАКТИЧЕСКИХ РАБОТ

для специальности 10.02.05 «Обеспечение информационной безопасности автоматизированных систем»
по МДК.02.02 «Криптографические средства защиты информации»

Москва 2026

СОДЕРЖАНИЕ

ВВЕДЕНИЕ 3
ПРАВИЛА ВЫПОЛНЕНИЯ ПРАКТИЧЕСКИХ РАБОТ 5
ОПИСАНИЕ РАБОЧЕГО МЕСТА ОБУЧАЮЩЕГОСЯ 6
ПРАКТИЧЕСКИЕ РАБОТЫ 7
Практическая работа № 1 7
Практическая работа № 2 12
Практическая работа № 3 17
Практическая работа № 4 20
Практическая работа № 5 22
Практическая работа № 6 26
Практическая работа № 7 29
Практическая работа № 8 33
Практическая работа № 9 40
Практическая работа № 10 41

ВВЕДЕНИЕ

Место дисциплины в основной образовательной программе: МДК.02.02 «Криптографические средства защиты информации» является обязательным разделом профессионального модуля ПМ.02 «Защита информации в автоматизированных системах программными и программно-аппаратными средствами» основной образовательной программы по специальности 10.02.05 Обеспечение информационной безопасности автоматизированных систем.

В результате освоения МДК.02.02 обучающийся должен:

уметь:

- проверять выполнение требований по защите информации от несанкционированного доступа при аттестации объектов информатизации по требованиям безопасности информации;
- применять математический аппарат для выполнения криптографических преобразований;
- использовать типовые программные криптографические средства, в том числе электронную подпись;
- применять средства гарантированного уничтожения информации;
- осуществлять мониторинг и регистрацию сведений, необходимых для защиты объектов информатизации, в том числе с использованием программных и программно-аппаратных средств обнаружения, предупреждения и ликвидации последствий компьютерных атак

знать:

- типовые модели управления доступом, средств, методов и протоколов идентификации и аутентификации;
- основные понятия криптографии и типовых криптографических методов и средств защиты информации;
- особенности и способы применения программных и программно-аппаратных средств гарантированного уничтожения информации.

Формируемые МДК.02.02 «Криптографические средства защиты информации» компетенции:

Код	**Наименование**
ПК 2.1.	Осуществлять обработку, хранение и передачу информации ограниченного доступа.
ПК 2.2.	Осуществлять регистрацию основных событий в автоматизированных (информационных) системах, в том числе с использованием программных и программно-аппаратных средств обнаружения, предупреждения и ликвидации последствий компьютерных атак.
ОК 1.	Выбирать способы решения задач профессиональной деятельности, применительно к различным контекстам.
ОК 2.	Осуществлять поиск, анализ и интерпретацию информации, необходимой для выполнения задач профессиональной деятельности.
ОК 3.	Осуществлять устную и письменную коммуникацию на государственном языке с учетом особенностей социального и культурного контекста.
ОК 4.	Проявлять гражданско-патриотическую позицию, демонстрировать осознанное поведение на основе традиционных общечеловеческих ценностей.
ОК 5.	Использовать информационные технологии в профессиональной деятельности.
ОК 6.	Пользоваться профессиональной документацией на государственном и иностранном языках.

Методические указания предназначены для проведения практических занятий по МДК.02.02, закрепления теоретических знаний и получения навыков работы в области криптографической защиты информации.

Методические указания разработаны в соответствии с рабочей программой профессионального модуля ПМ.02. «Защита информации в автоматизированных системах программными и программно-аппаратными средствами» по специальности 10.02.05 «Обеспечение информационной безопасности автоматизированных систем».

По учебному плану, и в соответствии с рабочей программой профессионального модуля ПМ.02, на изучение МДК.02.02. обучающимися предусмотрено 196 часов, из них практических – 50.

Методические указания включают 10 практических работ. Каждая практическая работа содержит сведения о теме, цели ее проведения и формируемых компетенциях, включает пояснения к работе, содержание отчета, контрольные задания или вопросы, список литературы.

К выполнению практических работ обучаемые приступают после подробного изучения соответствующего теоретического материала и прохождения инструктажа по технике безопасности.

Характер практических работ репродуктивный и частично-репродуктивный.

ПРАВИЛА ВЫПОЛНЕНИЯ ПРАКТИЧЕСКИХ РАБОТ

Практические работы студент выполняет самостоятельно на своем рабочем месте. Для выполнения работ студенту необходимо знание пройденной теории по дисциплине. При возникновении вопросов студент может задать их преподавателю.

После окончания занятий обучающиеся приводят в порядок рабочее место, показывают преподавателю полученные результаты. После утверждения преподавателем предъявленных результатов, каждый обучающийся оформляет отчет о проделанной работе, представляет его на проверку и подпись преподавателю в тот же день или на следующем практическом занятии.

ОПИСАНИЕ РАБОЧЕГО МЕСТА ОБУЧАЮЩЕГОСЯ

1. Практические работы по МДК.02.02 «02 «Криптографические средства защиты информации»» выполняются в компьютерном классе.
2. Для выполнения практических работ необходимы:
    a. Персональный компьютер;
    b. Операционная система Windows;
    c. Приложения MS Office;
    d. Доступ к сети Internet;
    e. Microsoft Visual Studio;
    f. Методические указания.

ПРАКТИЧЕСКИЕ РАБОТЫ

Практическая работа № 1

Тема: Математические основы криптографии

Цель:

Изучить:

- применение алгоритма Евклида для нахождения НОД;
- решение линейных диофантовых уравнений;
- алгоритм проверки чисел на простоту;
- решение задач с элементами теории чисел.

Формируемые компетенции: ПК 2.4, ОК 1, ОК 2, ОК 5.

Пояснения к работе:

Алгоритм Евклида

Нахождение наибольшего общего делителя (НОД) двух положительных целых чисел путем составления списка всех общих делителей непригодно для достаточно больших чисел.

Алгоритм Евклида основан на следующих двух фактах:

Факт 1: НОД (a, 0) = a
Факт 2: НОД (a, b) = НОД (b, r), где r — остаток от деления a на b

Первый факт говорит, что если второе целое число — 0, наибольший общий делитель равен первому числу. Второй факт позволяет нам изменять значение a на b, пока b не станет 0. Например, вычисляя НОД (36, 10), мы можем использовать второй факт несколько раз и один раз первый факт, как показано ниже.
НОД(36,10) = НОД(10,6) = НОД(6,4) = НОД(4,2) = НОД(2,0)

Для определения НОД мы используем две переменные, r_1 и r_2, чтобы запоминать изменяющиеся значения в течение всего процесса. Они имеют начальное значение a и b. На каждом шаге мы вычисляем остаток от деления r_1 на r_2 и храним результат в виде переменной r. Потом заменяем r_1, на r_2 и r_2 на r и продолжаем шаги, пока r не станет равным 0. В этот момент процесс останавливается и НОД (a, b) равен r_1.

Пример
Нужно найти наибольший общий делитель 2740 и 1760.
Решение
Мы присваиваем начальное значение r_1 2740 и r_2 значение 1760. В таблице также показаны значения q на каждом шаге. Мы имеем НОД (2740, 1760) = 20.

q	**r_1**	**r_2**	**r**
1	2740	1760	980
1	1760	980	780
1	980	780	200
3	780	200	180

1	200	180	20
9	180	20	0
	20	0	

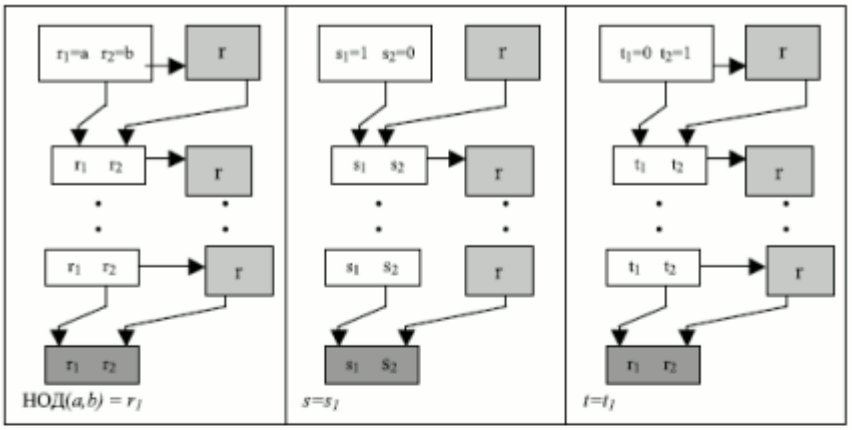
Расширенный алгоритм Евклида

Даны два целых числа a и b. Нам зачастую надо найти другие два целых числа, s и t, такие, которые

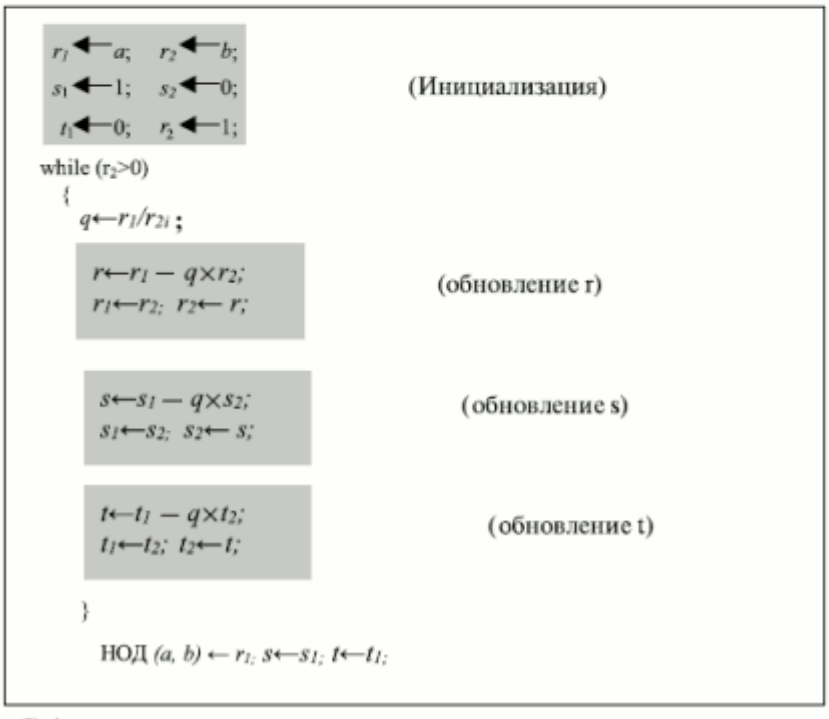
s x a + t x b = НОД(a,b)

Расширенный алгоритм Евклида может вычислить НОД (a, b) и в то же самое время вычислить значения s и t. Алгоритм и процесс такого вычисления показан на рисунке 1.

Здесь расширенный алгоритм Евклида использует те же самые шаги, что и простой алгоритм Евклида. Однако в каждом шаге мы применяем три группы вычислений вместо одной. Алгоритм использует три набора переменных: r, s и t.



а) Процесс



б) Алгоритм

Рис. 1. Расширенный алгоритм Евклида

На каждом шаге переменные r_1, r_2 и r используются так же, как в алгоритме Евклида. Переменным r_1 и r_2 присваиваются начальные значения a и b соответственно. Переменным s_1 и s_2 присваиваются начальные значения 1 и 0 соответственно. Переменным t_1 и t_2 присваиваются начальные значения 0 и 1, соответственно. Вычисления r, s и t одинаковы, но с одним отличием. Хотя r — остаток от деления r_1 на r_2, такого соответствия в других двух группах вычислений нет. Есть только одно частное, q, которое вычисляется как r_1/r_2 и используется для других двух вычислений.

Пример
Дано a = 161 и b = 28, надо найти НОД (a, b) и значения s и t.
Решение
$r = r_1 – q \times r_2$ $s = s_1 – qs_2$ $t = t_1 – q \times t_2$
Для отображения алгоритма мы используем следующую таблицу:

q	**r_1**	**r_2**	**R**	**s_1**	**s_2**	**s**	**t_1**	**t_2**	**t**
5	161	28	21	1	0	1	0	1	-5
1	28	21	7	0	1	-1	1	-5	6
3	21	7	0	1	-1	4	-5	6	-23
	7	0		-1	4		6	-23	

Мы получаем НОД (161, 28) = 7, s = –1 и t = 6. Ответы могут быть проверены, как это показано ниже.
(–1) x 161 + 6 x 28 = 7

Линейные диофантовы уравнения
Очень важное приложение расширенного алгоритма Евклида "нахождение решения линейных диофантовых уравнений двух переменных", а именно, уравнения ax + by = c. Мы должны найти значения целых чисел для x и y, которые удовлетворяют этому уравнению. Этот тип уравнения либо не имеет решений, либо имеет бесконечное число решений. Пусть d = НОД (a, b). Если d†c, то уравнение не имеет решения. Если d|c, то мы имеем бесконечное число решений. Одно из них называется частным, остальные — общими.
Линейное диофантово уравнение — это уравнение двух переменных: $ax + by = c$.
Частное решение
Если d|c, то можно найти частное решение вышеупомянутого уравнения, используя следующие шаги.

1. Преобразуем уравнение к $a_1x + b_1y = c_1$, разделив обе части уравнения на d. Это возможно, потому, что d делит a, b, и c в соответствии с предположением.
2. Найти s и t в равенстве $a_1s + b_1t = 1$, используя расширенный алгоритм Евклида.
3. Частное решение может быть найдено:

Частное решение: $X_0 = (c/d)s$ и $y_0 = (c/d)t$
Общие решения
После нахождения частного решения общие решения могут быть найдены:
Общие решения: $x = x_0 + k(b/d)$ и $y = y_0 – k(a/d)$, где k — целое число
Пример
Найти частные и общие решения уравнения 21x + 14y = 35.
Решение
Мы имеем d = НОД (21, 14) = 7. При 7|35 уравнение имеет бесконечное число решений. Мы можем разделить обе стороны уравнения на 7 и получим уравнение 3x + 2y = 5. Используя расширенный алгоритм Евклида, мы находим s и t, такие, что 3s + 2t = 1. Мы имеем S = 1 и t = –1. Решения будут следующие:
Частное решение : $x_0 = 5 \times 1 = 5$ и $y_0 = 5 \times (–1) = -5$ тогда 35/7 =5
Общие: $x = 5 + k \times 2$ $y = –5 – k \times 3$ где k — целое
Поэтому решения будут следующие (5, –5), (7, –8), (9, –11)...
Мы можем легко проверить, что каждое из этих решений удовлетворяет первоначальному уравнению.

Малая теорема Ферма
Первая версия
Первая версия говорит, что если p — простое число и a — целое число, такое, что p не является делителем a, тогда $a^{p-1} \equiv 1 \mod p$.
Вторая версия
Вторая версия вводит ограничивающие условие на a. Она утверждает, что если p — простое число и a — целое число, то $a^p \equiv a \mod p$.
Приложения
Возведение в степень. Малая теорема Ферма иногда полезна для того, чтобы быстро найти решение при возведении в степень.
Пример
Найдите результат 6^{10} mod 11.
Решение
Мы имеем $6^{10} \mod 11 \equiv 1$. Это первая версия малой теоремы Ферма, где p = 11.
Пример
Найдите результат 3^{12} mod 11.
Решение
Здесь степень (12) и модуль (11) не соответствуют условиям теоремы Ферма. Но, применяя преобразования, мы можем привести решение к использованию малой теоремы Ферма.

$$3^{12} \bmod 11 \equiv (3^{11} \times 3) \bmod 11 \equiv (3^{11} \bmod 11)(3 \bmod 11) \equiv (3 \times 3) \bmod 11 = 9$$

Задание:

1. С помощью алгоритма Евклида найти НОД чисел:
    1. НОД (130 , 15)
    2. НОД (374 , 32)
    3. НОД (5741 , 76)
    4. НОД (846 , 42)
    5. НОД (5783 , 3420)
2. С помощью расширенного Алгоритма Евклида найти НОД (a,b) и значения s и t:
    1. a = 57 b = 14
    2. a = 328 b = 36
    3. a = 1179 b = 27
    4. a = 502 b = 52
    5. a = 791 b = 13
3. Найдите частное и общие решения следующих линейных диофантовых уравнений:
    1. 25x + 10y = 15
    2. 19x + 13y = 20
    3. 14x + 21y = 77
    4. 40x +16y = 88
4. Определите, сколько из следующих целых чисел пройдут испытание Ферма на простоту чисел: 100, 110, 130, 150, 200, 250, 271, 341, 561. Используйте основание 2.
5. Найдите результаты следующих операций:
    1. 22 mod 7
    2. 140 mod 10
    3. -78 mod 13
    4. 0 mod 15

Содержание отчета: Отчет должен содержать пошаговое решение заданий.

Контрольные вопросы:

1. Определите наибольший общий делитель двух целых чисел. Какой алгоритм может эффективно найти наибольший общий делитель?
2. Что такое линейное диофантово уравнение двух переменных? Сколько решений может иметь такое уравнение? Как может быть найдено решение(я)?
3. Объясните разницу между простым числом и составным целым числом.

Список литературы:

Интернет ресурс: НОУ Интуит : https://intuit.ru/studies/courses/552/408/info

Практическая работа № 2

Тема: Методы криптографического защиты информации

Цель: Изучить классических шифров замены и классических шифров перестановки

Формируемые компетенции: ПК 2.1,ОК 1, ОК 2, ОК 5.

Пояснения к работе:

Аддитивный шифр

Предположим, что исходный текст состоит из маленьких букв (от a до z) и зашифрованный текст состоит из заглавных букв (от A до Z). Чтобы обеспечить применение математических операций к исходному и зашифрованному текстам, мы присвоим каждой букве числовое значение (для нижнего и верхнего регистра), как это показано на рисунке 2.

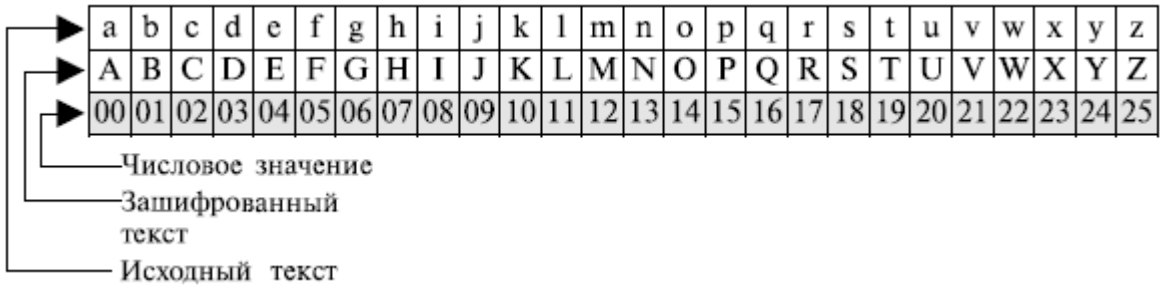


Рис. 2. Представление букв исходного текста и зашифрованного текста в Z26

На рисунке 2 каждому символу (нижний регистр или верхний регистр) сопоставлено целое число из Z_{26}. Ключ засекречивания между Алисой и Бобом — также целое число в Z_n. Алгоритм кодирования прибавляет ключ к символу исходного текста; алгоритм дешифрования вычитает ключ из символа зашифрованного текста. Все операции проводятся в Z_n. Рисунок 3 показывает процесс шифрования и дешифрования.

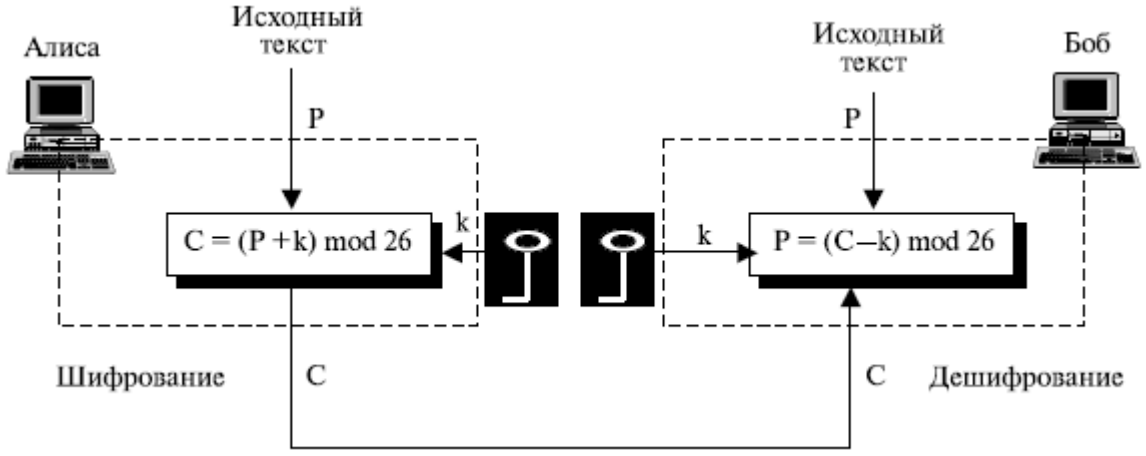


Рис. 3. Аддитивный шифр

Пример

Используйте аддитивный шифр с ключом = 15, чтобы зашифровать сообщение "hello".

Решение

Мы применяем алгоритм кодирования к исходному тексту, буква за буквой:

Исходный текст h -> 07 Шифрование (07 + 15) mod 26 Шифр. Текст 22 -> W
Исходный текст e -> 04 Шифрование (04+ 15) mod 26 Шифр. Текст 19 -> T
Исходный текст l -> 11 Шифрование (11 + 15) mod 26 Шифр. Текст 00 -> A
Исходный текст l -> 11 Шифрование (11 + 15) mod 26 Шифр. Текст 00 -> A
Исходный текст o -> 14 Шифрование (14 + 15) mod 26 Шифр. Текст 03 -> D
Результат — "WTAAD". Обратите внимание, что шифр моноалфавитный, потому что два отображения одной и той же буквы исходного текста (l) зашифрованы как один и тот же символ.

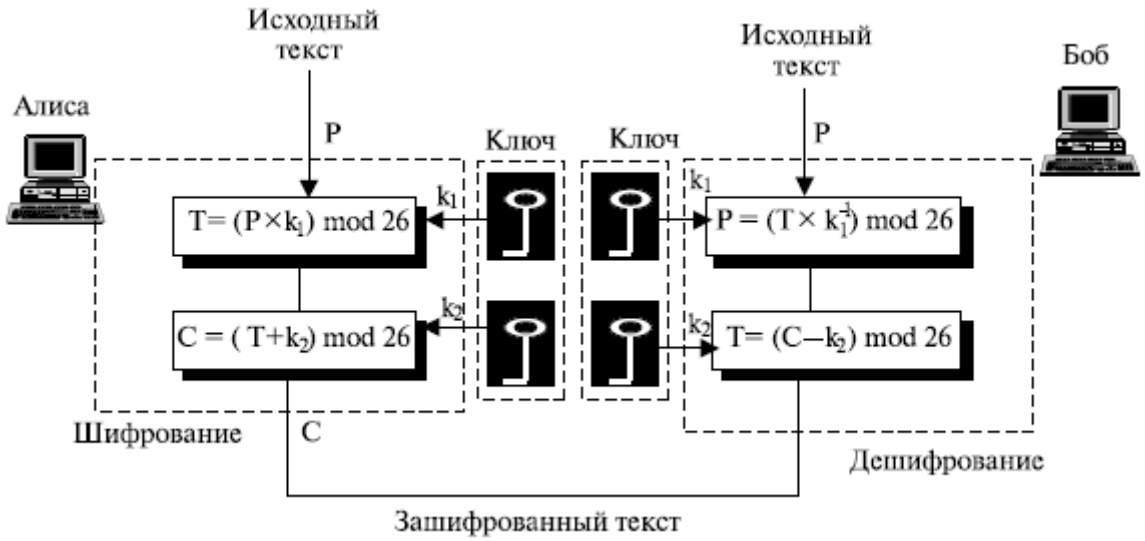


Рис. 4. Аффинный шифр

При аффинном шифре (рисунок 4) отношение между исходным текстом P и шифрованным текстом C определяется, как это показано ниже.

$C = (P \times k_1 + k_2) \bmod 26$ $P = ((C - k_2) \times k_1^{-1}) \bmod 26$
где k_1^{-1} мультипликативная инверсия k_1, a$(-k_2)$–аддитивная инверсия k_2

Пример

Используйте аффинный шифр, чтобы зашифровать сообщение "hello" с ключевой парой (7, 2).

Решение

Мы используем 7 для мультипликативного ключа и 2 для аддитивного ключа. Получаем "ZEBBW".

P: h 07 Шифрование (07 x 07+2) mod 26 C: 25 Z
P: e 04 Шифрование (04 x 07+2) mod 26 C: 04 E
P: l 11 Шифрование (11 x 07+2) mod 26 C: 01 B
P: l 11 Шифрование (11 x 07+2) mod 26 C: 01 B
P: o 14 Шифрование (14 x 07+2) mod 26 C: 22 W

Шифр Виженера

Шифр Виженера использует различную стратегию создания потока ключей. Поток ключей — повторение начального потока ключа засекречивания длины m, где мы имеем $1 < m < 26$. Шифр может быть описан следующим образом: (k_1, k_2, …., k_m) — первоначальный ключ засекречивания, согласованный Алисой и Бобом.

$P = P_1P_2P_3…..$
$C = C_1C_2C_3…..$
$k = [(k_1,k_2), (k_3,k_4),…..]$
Шифрование $C_i = k_i$
Дешифрование $P_i = k_i$

Поток ключей Виженера не зависит от символов исходного текста; он зависит только от позиции символа в исходном тексте. Другими словами, поток ключей может быть создан без знания сути исходного текста.

Пример

Посмотрим, как мы можем зашифровать сообщение "She is listening (Она слушает) ", используя ключевое слово на 6 символов "PASCAL". Начальный поток ключей — это (15, 0, 18, 2, 0, 11). Поток ключей — повторение этого начального потока ключей (столько раз, сколько необходимо).

Исходный текст	s	h	e	i	s	l	i	s	t	e	n	i	n	g
Значения P	18	07	04	08	18	11	08	18	19	04	13	08	13	06
Поток ключей	15	00	18	02	00	11	15	00	18	02	00	11	15	00
Значения C	07	07	22	10	18	22	23	18	11	6	13	19	02	06
Шифрованный текст	H	H	W	K	S	W	X	S	L	G	N	T	C	G

Шифр Плейфера

Ключ засекречивания в этом шифре сделан из 25 букв алфавита, размещенных в матрице 5×5 (буквы I и J рассматриваются при шифровании как одинаковые). С помощью различных соглашений о размещении букв в матрице можно создать много различных ключей засекречивания. Одно из возможных соглашений показано на рисунке 5.

Секретный ключ =

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

Рис. 5. Пример секретного ключа Плейфера

Перед шифрованием исходный текст разбивается на пары; если две буквы пары одинаковые, то, чтобы отделить их, вставляется фиктивная буква. После вставки фиктивных букв, если число символов в исходном тексте нечетно, в конце добавляется один дополнительный фиктивный символ, чтобы сделать число символов четным.

Шифр использует три правила для шифрования:

a. если эти две буквы-пары расположены в одной и той же строке таблицы ключа засекречивания, соответствующий зашифрованный символ для каждой буквы — следующий символ справа в той же самой строке (с возвращением к началу строки; если символ исходного текста — последний символ в строке);

b. если эти две буквы-пары расположены в одном и том же столбце таблицы ключа засекречивания, соответствующий зашифрованный символ для каждой буквы — символ ниже этого в том же самом столбце (с возвращением к началу столбца; если символ исходного текста — последний символ в столбце);

c. если эти две буквы-пары не находятся в одной строке или столбце таблицы засекречивания, соответствующий зашифрованный символ для каждой буквы — символ, который находится в его собственной строке, но в том же самом столбце, что и другой символ.

В шифре Плейфера поток ключей и поток шифра — те же самые. Это означает, что вышеупомянутые правила можно представить как правила для создания потока ключей. Алгоритм кодирования берет пару символов из исходного текста и создает пару подключей, следуя указанным правилам. Мы можем сказать, что поток ключей зависит от позиции символа в исходном тексте. Зависимость от позиции имеет здесь различную интерпретацию: подключ для каждого символа исходного текста зависит от следующего или предыдущего "соседа". Рассматривая шифр Плейфера, таким образом, можно сказать, что зашифрованный текст — это фактически поток ключей.

$P = P_1P_2P_3…..$
$C = C_1C_2C_3…..$
$k = [(k_1,k_2), (k_3,k_4),…..]$
Шифрование $C_i = k_i$
Дешифрование $P_i = k_i$

Пример

Пусть нам надо зашифровать исходный текст "hello", использующий ключи на рис. 5. Когда мы группируем буквы по парам, мы получаем "he, ll,o". Мы должны вставить x между двумя l (эль), после чего получим "he, lx, lo". Мы имеем

he -> EC lx -> QZ lo -> BX
Исходный текст: hello Зашифрованный текст: ECQZBX

Мы можем видеть из этого примера, что наш шифр — фактически многоалфавитный шифр: два появления l (эль) зашифрованы как "Q" и "B".

Задание:

1. Зашифруйте сообщение "this is exercise" ("это — упражнение"), используя аддитивный шифр с ключом = 20 и аффинный шифр с ключом = (15, 20). Игнорируйте пробелы между словами. Расшифруйте сообщение, чтобы получить первоначальный исходный текст.
2. Зашифруйте сообщение "the house is being sold tonight" ("дом продан сегодня вечером"), используя шифр Виженера с ключом: "dollars". Игнорируйте пространство между словами. Расшифруйте сообщение, чтобы получить исходный текст.

3. Используйте шифр Виженера с ключевым словом "HEALTH" , чтобы зашифровать сообщение "Life is full surprises" ("Жизнь полна сюрпризов").
4. Используйте шифр Плейфера, чтобы зашифровать сообщение "The key hidden under the door pad" ("ключ спрятан под ковриком у двери"). Ключ засекречивания можно составить, заполняя первую и вторую часть строки со словом "GUIDANCE" и заполняя остальную часть матрицы с остальной частью алфавита.

Содержание отчета: Отчет должен содержать пошаговое решение заданий.

Контрольные вопросы:

1. Поясните отличия между шифром подстановки и шифром перестановки.
2. Поясните отличия между шифром подстановки и шифром перестановки.

Список литературы:

Интернет ресурс: НОУ Интуит : https://intuit.ru/studies/courses/552/408/info

Практическая работа № 3

Тема: Криптоанализ

Цель:

Изучить:

- криптоанализ шифра простой замены методом анализа частотности символов;
- криптоанализ классических шифров методом полного перебора ключей;
- криптоанализ шифра Вижинера.

Формируемые компетенции: ПК 2.1, ПК 2.2, ОК 1, ОК 2, ОК 5.

Пояснения к работе:

Статистическая атака (атака частоты отдельных букв)

Шифр перестановки не изменяет частоту букв в зашифрованном тексте; он только переставляет буквы. Так что первая атака, которая может быть применена, - анализ частоты отдельной буквы. Этот метод может быть полезен, если длина зашифрованного текста достаточно большая. Однако шифры перестановки не сохраняют частоту пар и триграмм. Это означает, что Ева не может использовать такие инструментальные средства. Фактически, если шифр не сохраняет частоту пар и триграмм, но сохраняет частоту отдельных букв, то вероятнее всего, что это шифр перестановки.

Ева, чтобы расшифровать сообщение, может попробовать все возможные ключи. Однако число ключей может быть огромно 1! + 2! + 3! + … + L!, где L — длина зашифрованного текста. Лучший подход состоит в том, чтобы попробовать отгадать число столбцов. Ева знает, что число столбцов делится на L. Например, если длина шифра — 20 символов, то . Это означает, что номером столбцов может быть комбинация этих коэффициентов (1, 2, 4, 5, 10, 20). Однако только один столбец и только одна строка — маловероятные варианты.

Пример

Предположим, что Ева перехватила сообщение зашифрованного текста "EEMYNTAACTTKONSHITZG". Длина сообщения L = 20, число столбцов может быть 1, 2, 4, 5, 10 или 20. Ева игнорирует первое значение, потому что это означает только один столбец и оно маловероятно.

a. Если число столбцов — 2, единственные две перестановки — (1,2) и (2, 1). Первое означает, что перестановки не было. Ева пробует вторую комбинацию. Она делит зашифрованный текст на модули по два символа "EE MY NT AA CT TK ON SH IT ZG". Затем она пробует переставлять каждый модуль из них, получая текст "ee ym nt aa tc kt no hs ti gz", который не имеет смысла.

b. Если номер столбцов — 4, тогда имеется 4! = 24 перестановки. Первая перестановка (1 2 3 4) означает, что не было никакой перестановки. Ева должна попробовать остальные. После испытания всех 23 возможностей Ева находит, что никакой исходный текст при таких перестановках не имеет смысла.

c. Если число столбцов — 5, тогда есть 5! = 120 перестановок. Первая (1 2 3 4 5) означает отсутствие перестановки. Ева должна попробовать остальные. Перестановка (2 5 13 4) приносит плоды — исходный текст "enemyattackstonightz", который имеет смысл после удаления фиктивной буквы z и добавления пробелов.

Криптоанализ шифра Виженера

Шифры Виженера, подобно всем многоалфавитным шифрам, не сохраняют частоту символов. Однако Ева может использовать некоторые методы для того, чтобы расшифровать перехваченный зашифрованный текст. Криптоанализ в данном случае состоит из двух частей: находят длину ключа и потом непосредственно находят ключ.

1. Были изобретены несколько методов, чтобы найти длину ключа. Один метод рассмотрим ниже. В так называемом тесте Казиского (Kasiski) криптоаналитик в зашифрованном тексте ищет повторные сегменты по крайней мере из трех символов. Предположим, что найдены два сегмента, и расстояние между ними d. Криптоаналитик предполагает, что m делит d, где m — длина ключа. Если можно найти больше повторных сегментов с расстоянием d_1, d_2, …., d_n, тогда НОД(d_1, d_2, …., d_n....)/ m. Это предположение логично, потому что если два символа одинаковы и $-k \times m$ (k = 1, 2...) — символы, выделенные в исходном тексте, то одинаковы и $k \times m$ символы, выделенные в зашифрованном тексте. Криптоаналитик использует сегменты по крайней мере из трех символов, чтобы избежать случаев, где символы имеют один и тот же ключ.
2. Зашифрованный текст делится на m различных частей и применяется метод, используемый в криптоанализе аддитивного шифра, включая атаку частоты. Каждая часть зашифрованного текста может быть расшифрована и соединена с другими, чтобы создать целый исходный текст, другие слова. Весь зашифрованный текст не сохраняет частоту отдельной буквы исходного текста, но каждая часть делает это.

Пример

Предположим, что мы перехватили следующий зашифрованный текст:

LIOMWGFEGGDVWGHHCQUCRHRWAGWIOWQLKGZETKKMEVLWPCZVG'TH
VTSGXQOVGCSVETQLTJSUMVWVEUVLXEWSLGFZMVVWLGYHCUSWXQH -
KVGSHEEVFLCFDGVSUMPHKIRZDMPHHBVWVWJWIXGFWLTSHGJOUEEHH-
VUCFVGOWICQLIJSUXGLW

Тест Казиского на повторение сегментов на три символа приводит к результатам, показанным в таблице ниже.

Таблица . Тест Казиского			
Комбинация	Первое расстояние	Второе расстояние	Разность
JSU	68	168	100
SUM	69	117	48
VWV	72	132	60
MPH	119	127	8

Наибольший делитель - 4, что означает длину ключа, пропорциональную 4. Сначала пробуем m = 4. Делим зашифрованный текст на четыре части. Часть C_1 состоит из символов 1, 5, 9...; часть C_2 состоит из символов 2, 6, 10..., и так далее. Используем статистическую атаку каждой части отдельно. Перебираем расшифровывающиеся части по одному символу одновременно, чтобы получить целый исходный текст.

C_1	LWGW	CRAO	KTEP	GTQC	TJVU	EGVG	UQGE	CVPR	PVJG	TJEU	GCJG
P_1	jueu	apym	ircn	eroa	rhts	thin	ytra	hcie	ixst	hcar	rehe
C_2	IGGG	QHGW	GKVC	TSOS	QSWV	WFVY	SHSV	FSHZ	HWWF	SOHC	OQSL
P_2	usss	ctsl	swho	feae	ceih	cete	soec	atnp	nkhe	rhck	esex

C_3	OFDN	URWQ	ZKLZ	HGVV	LUVL	SZWH	WKHF	DUKD	HVIW	HUHF	WLUW
P_3	lcae	rotn	whiw	edss	irsi	irh	eteh	retl	tiid	eatr	airt
C_4	MEVN	CWIL	EMWV	VXGE	TMEX	LMLC	XVEL	GMIM	BWHL	GEVV	ITX
P_4	iard	yseh	aisr	rtca	piaf	pwte	thec	arha	esft	erec	tpt

Если исходный текст не имеет смысла, попробуем с другим m.

В рассматриваемом случае исходный текст имеет смысл (читаем по столбцам).

Julius Caesar used a cryptosystem in his wars, which is now referred to as Caesar chipper. It is an additive chipper with the key set to three. Each character in the plaintext is shift three characters to create ciphertext.

Перевод этого текста приведен ниже.

Юлий Цезарь использовал в своих войнах криптографическую систему, которая упоминается теперь как шифр Цезаря. Это аддитивный шифр с ключом, установленным на три. Каждый символ в исходном тексте сдвинут на три символа, чтобы создать зашифрованный текст.

Задание:

1. Используйте атаку частоты отдельных букв, чтобы взломать следующий зашифрованный текст. Предположите, что вы знаете, что он был создан с использованием аддитивного шифра.
   OTWEWNGWCBPQABIZVQAPMLJGZWTTQVOBQUMAPMIDGZCABEQVBMZL
   ZIXMLAXZQVOQVLMMXAVWEIVLLIZSNZWABJQZLWNLMTQOPBVIUMLG
   WCBPAEQNBTGTMNBBPMVMAB
2. Используйте тест Казиского и атаку частоты отдельных букв, чтобы нарушить следующий зашифрованный текст. Предположите, что вы знаете, что он был создан шифром Виженера.
   OTWEWNGWCBPQABIZVQAPMLJGZWTTQVOBQUMAPMIDGZCABEQVBMZL
   ZIXMLAXZQVOQVLMMXAVWEIVLLIZSNZWABJQZLWNLMTQOPBVIUMLG
   WCBPAEQNBTGTMNBBPMVMAB

Содержание отчета: Отчет должен содержать пошаговое решение заданий.

Контрольные вопросы:

1. Опишите суть изученных методов криптоанализа.

Список литературы:

Интернет ресурс: НОУ Интуит : https://intuit.ru/studies/courses/552/408/info

Практическая работа № 4

Тема: Поточные шифры и генераторы псевдослучайных чисел
Цель: Изучить применение методов генерации ПСЧ
Формируемые компетенции: ПК 2.4,ОК 1, ОК 2, ОК 5.
Пояснения к работе:

Генераторы псевдослучайных чисел могут работать по разным алгоритмам. Одним из простейших генераторов является так называемый линейный конгруэнтный генератор, который для вычисления очередного числа ki использует формулу ki=(a*ki-1+b)mod c,
где a, b, c — некоторые константы, a ki-1 — предыдущее псевдослучайное число. Для получения k1 задается начальное значение k0. Возьмем в качестве примера a=5,b=3,c=11 и пусть k0= 1. В этом случае мы сможем по приведенной выше формуле получать значения от 0 до 10 (так как c = 11). Вычислим несколько элементов последовательности:
k1 = (5 * 1 + 3) mod 11 = 8;
k2 = (5 * 8 + 3) mod 11 = 10;
k3 = (5 * 10 + 3) mod 11 = 9;
k4 = (5 * 9 + 3) mod 11 = 4;
k5 = (5 * 4 + 3) mod 11 = 1.
Полученные значения (8, 10, 9, 4, 1) выглядят похожими на случайные числа. Однако следующее значение k6 будет снова равно 8:

Метод Фибоначчи с запаздыванием
Известны разные схемы использования метода Фибоначчи с запаздыванием. Один из широко распространённых фибоначчиевых датчиков основан на следующей рекуррентной формуле:

$$k_i = \begin{Bmatrix} k_{i-a} - k_{i-b}, & если\ k_{i-a} \geq k_{i-b} \\ k_{i-a} - k_{i-b} + 1, & если\ k_{i-a} < k_{i-b} \end{Bmatrix}$$

где ki — вещественные числа из диапазона [0,1], a, b — целые положительные числа, параметры генератора. Для работы фибоначчиеву датчику требуется знать max{a,b} предыдущих сгенерированных случайных чисел. При программной реализации для хранения сгенерированных случайных чисел необходим некоторый объем памяти, зависящих от параметров a и b.

Пример.
Вычислим последовательность из первых десяти чисел, генерируемую методом Фибоначчи с запаздыванием начиная с k5 при следующих исходных данных: a = 4, b = 1, k0=0.1; k1=0.7; k2=0.3; k3=0.9; k4=0.5:

k5 = k1 - k4 = 0.7 - 0.5 = 0.2;
k6 = k2 - k5= 0.3 - 0.2 = 0.1;
k7 = k3 - k6 = 0.9 - 0.1 = 0.8;
k8 = k4 - k7 + 1 =0.5 - 0.8 + 1 = 0.7;
k9 = k5- k8 + 1 =0.2 - 0.7 + 1 = 0.5;
k10 = k6 - k9 + 1 =0.1 - 0.5 + 1 = 0.6;
k11 = k7 - k10 = 0.8 - 0.6 = 0.2;
k12 = k8 - k11 = 0.7 - 0.2 = 0.5;
k13 = k9 - k12 + 1 =0.5 - 0.5 + 1 = 1;
k14 = k10 - k13 + 1 =0.6 - 1 + 1 = 0.6.

Видим, что генерируемая последовательность чисел внешне похожа на случайную. И действительно, исследования подтверждают, что получаемые случайные числа обладают хорошими статистическими свойствами.

Широкое распространение получил алгоритм генерации псевдослучайных чисел, называемый алгоритмом BBS (от фамилий авторов — L. Blum, M. Blum, M. Shub) или генератором с квадратичным остатком. Для целей криптографии этот метод предложен в 1986 году.

Вначале выбираются два больших простых числа p и q. Числа p и q должны быть оба сравнимы с 3 по модулю 4, то есть при делении p и q на 4 должен получаться одинаковый остаток 3. Далее вычисляется число M = p* q, называемое целым числом Блюма. Затем выбирается другое случайное целое число x, взаимно простое (то есть не имеющее общих делителей, кроме единицы) с M. Вычисляем $x0= x^2 \bmod M$. x_0 называется стартовым числом генератора.

На каждом n-м шаге работы генератора вычисляется $x_{n+1}= x_n^2 \bmod M$. Результатом n-го шага является один (обычно младший) бит числа x_{n+1}. Иногда в качестве результата принимают бит чётности, то есть количество единиц в двоичном представлении элемента. Если количество единиц в записи числа четное – бит четности принимается равным 0, нечетное – бит четности принимается равным 1.

Пример

Пусть p = 11, q = 19 (убеждаемся, что 11 mod 4 = 3, 19 mod 4 = 3). Тогда M = p* q = 11*19=209. Выберем x, взаимно простое с M: пусть x = 3. Вычислим стартовое число генератора x_0:

$x_0 = x^2 \bmod M = 3^2 \bmod 209 = 9 \bmod 209 = 9$.

Вычислим первые десять чисел x_i по алгоритму BBS. В качестве случайных бит будем брать младший бит в двоичной записи числа x_i:

$x_1=9^2 \bmod 209= 81 \bmod 209= 81$	младший бит:	1
$x_2=81^2 \bmod 209= 6561 \bmod 209= 82$	младший бит:	0
$x_3=82^2 \bmod 209= 6724 \bmod 209= 36$	младший бит:	0
$x_4=36^2 \bmod 209= 1296 \bmod 209= 42$	младший бит:	0
$x_5=42^2 \bmod 209= 1764 \bmod 209= 92$	младший бит:	0
$x_6=92^2 \bmod 209= 8464 \bmod 209= 104$	младший бит:	0
$x_7=104^2 \bmod 209= 10816 \bmod 209= 157$	младший бит:	1
$x_8=157^2 \bmod 209= 24649 \bmod 209= 196$	младший бит:	0
$x_9=196^2 \bmod 209= 38416 \bmod 209= 169$	младший бит:	1
$x_{10}=169^2 \bmod 209= 28561 \bmod 209= 137$	младший бит:	1

Задание:

1) Определите последовательность из первых десяти чисел и период линейного конгруэнтного генератора ПСЧ для различных параметров a, b и c (k0 принять равным 0):

   a =5, b =7 и c = 17;

a = 6, b =3 и c = 23.

2) Вычислите последовательность из десяти чисел, генерируемую методом Фибоначчи с запаздыванием начиная с ka при следующих исходных данных:

   a = 3, b = 1, k0=0.6; k1=0.3; k2=0.5;

   a = 4, b = 2, k0=0.9; k1=0.3; k2=0.5; k3=0.9.

3) Значения k0, k1, k2, k3, полученные с помощью линейного конгруэнтного генератора, равны: k0 = 1, k1 = 12, k2 = 3, k3 = 6. Найдите параметры a, b и c генератора ПСЧ.

4) Вычислить x11 по методу генерации псевдослучайных чисел BBS, если p = 11, q = 19, x = 3.

Содержание отчета: Отчет должен содержать пошаговое решение заданий.

Контрольные вопросы:

1. Какие числа называют "псевдослучайными"?
2. Какими свойствами должен обладать генератор псевдослучайных чисел для использования в криптографических целях?
3. Какие генераторы псевдослучайных чисел Вы можете назвать?
4. Перечислите основные характеристики, достоинства и недостатки каждого из рассмотренных в данной лекции генераторов псевдослучайных чисел.

Список литературы:

Интернет ресурс: НОУ Интуит : https://intuit.ru/studies/courses/691/547/lecture/12383

Практическая работа №5

Тема: Кодирование информации. Компьютеризация шифрования.

Цель: Освоить программную реализацию классических шифров

Формируемые компетенции: ПК 2.1, ПК 2.4, ОК 1, ОК 2, ОК 5, ОК 6.

Пояснения к работе:
Шифры Файстеля

Файстель проектировал очень интеллектуальный и интересный шифр, который использовался в течение многих десятилетий. Шифр Файстеля может иметь три типа компонентов: самообратимый, обратимый и необратимый.

Шифр Файстеля содержит в блоках все необратимые элементы и использует один и тот же модуль в алгоритмах дешифрования и шифрования. Вопрос в том, как алгоритмы шифрования и дешифрования позволяют инвертировать открытый и закрытый тексты друг в друга, если каждый содержит необратимый модуль. Файстель показал, что они могут быть сбалансированы.

Чтобы лучше понять шифр Файстеля, давайте посмотрим, как мы можем использовать один и тот же необратимый компонент в алгоритмах дешифрования и шифрования. Эффекты необратимого компонента в алгоритме шифрования могут быть отменены в алгоритме дешифрования, если мы используем операцию ИСКЛЮЧАЮЩЕЕ ИЛИ, как показано на рис. 6.

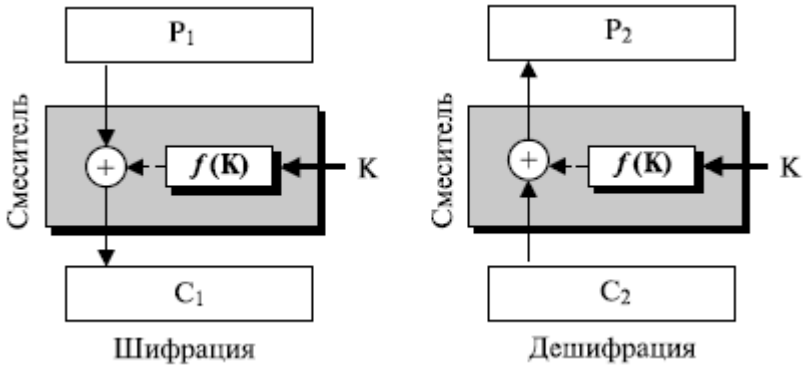


Рис. 6. Первая идея в разработке шифра Файстеля

В шифровании ключ поступает на вход необратимой функции f (K), которая является одним из слагаемых оператора ИСКЛЮЧАЮЩЕГО ИЛИ с исходным текстом. Результат становится зашифрованным текстом. Мы будем называть комбинацию функции и операции ИСКЛЮЧАЮЩЕЕ ИЛИ смесителем (из-за отсутствия другого названия). Смеситель играет важную роль в более поздних вариантах шифра Файстеля.

Поскольку ключ один и тот же в шифровании и дешифровании, мы можем доказать, что два алгоритма инверсны друг другу. Другими словами, если $C_2 = C_1$ (любое изменение в зашифрованном тексте в течение передачи), то $P_2 = P_1$.

$$\text{Шифрование}: C_1 = P_1 \oplus f(K) \text{Дешифрование}: P_2 = C_2 \oplus f(K) = C_1 \oplus f(K) = P_1 \oplus f(K) \oplus f(K) = P_1 \oplus (00\ldots0) = P_1$$

Обратите внимание, что использовались два свойства операции ИСКЛЮЧАЮЩЕЕ ИЛИ (существование инверсии и существование нулевого кода).
Уравнения, показанные выше, доказывают, что хотя смеситель имеет неконвертируемый элемент, сам смеситель является самоконвертируемым.

Усовершенствование. Попробуем улучшить нашу первую идею, чтобы приблизиться к шифру Файстеля. Мы знаем, что должны применить вход к неконвертируемому элементу (функции), но мы не будем использовать только ключ. Мы задействуем также вход к функции, чтобы применить ее для шифрования части исходного текста и дешифрования части зашифрованного текста. Ключ может использоваться как второй вход к функции. Этим способом наша функция становится сложным элементом с некоторыми неключевыми элементами и некоторыми ключевыми элементами. Чтобы достичь цели, разделим исходный текст и зашифрованный текст на два блока равной длины – левый (L) и правый (R). Правый блок вводится в функцию, а левый блок складывается с помощью операции ИСКЛЮЧАЮЩЕЕ ИЛИ с выходом функции. Мы должны запомнить, что входы к функции должны точно совпадать в шифровании и дешифровании. Это означает, что правая секция исходного текста до шифрования и правая секция зашифрованного текста после дешифрования будут совпадать. Другими словами, секция должна войти в шифрование и выйти из дешифрования неизмененной. Рисунок 7 иллюстрирует идею.

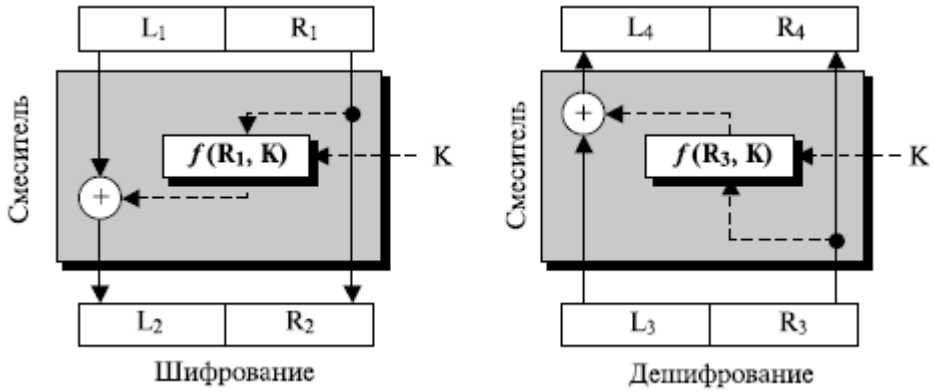


Рис. 7. Усовершенствование предыдущей схемы Файстеля

Алгоритмы шифрование и дешифрования инверсны друг другу. Предположим, что $L_3 = L_2$ и $R_3 = R_2$ (в зашифрованном тексте в течение передачи не произошло изменений).

$$R_4 = R_3 = R_2 = R_1 L_4 = L_3 \oplus f(R_3, K) = L_2 \oplus f(R_2, K) = L_1 \oplus f(R_1, K) \oplus f(R_1, K) = L_1$$

Исходный текст, используемый в алгоритме шифрования, — это текст, правильно восстановленный алгоритмом дешифрования.

Окончательный вариант. Предыдущее усовершенствование имеет один недостаток: правая половина исходного текста никогда не изменяется. Ева может немедленно найти правую половину исходного текста, разбивая на части зашифрованный текст и распаковывая его правую половину. Проект нуждается в дальнейших шагах усовершенствования.

Первое: увеличим число раундов. Второе: добавим новый элемент в каждый раунд — устройство замены. Эффект устройства замены в раунде шифрования компенсируется эффектом устройства замены в раунде дешифрования. Однако это позволяет нам менять левые и правые половины в каждом раунде. Рисунок 8 иллюстрирует новый вариант шифра Файстеля с двумя раундами.

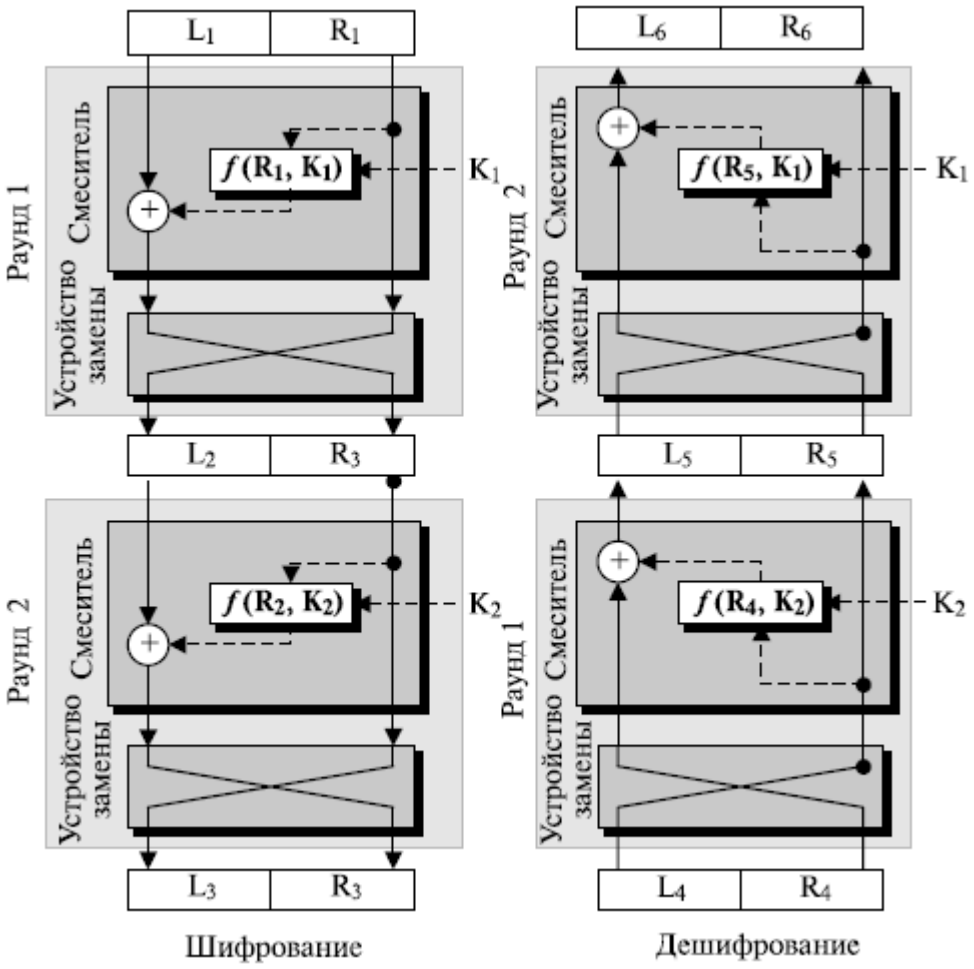


Рис. 8. Окончательный вариант шифра Файстеля с двумя раундами

Обратите внимание, что есть два ключа раундов: K_1 и K_2. Ключи используются в обратном порядке в шифровании и дешифровании.

Задание: Реализовать шифр Файстеля на любом из языков программирования.

Содержание отчета: Текст программной реализации классических шифров на любом языке программирования с подробными комментариями и тестовыми данными.

Контрольные вопросы: Укажите различие между блочным шифром Файстеля и не-Файстеля.

Список литературы:
Интернет ресурс: НОУ Интуит : https://intuit.ru/studies/courses/552/408/info

Практическая работа № 6

Тема: Асимметричные системы шифрования
Цель: Изучить применение различных асимметричных алгоритмов и изучить программную реализации асимметричного алгоритма RSA
Формируемые компетенции: ПК 2.1, ПК 2.4, ОК 1, ОК 2, ОК 5, ОК 6.
Пояснения к работе:

Алгоритм RSA
Алгоритм шифрования с открытым ключом RSA был предложен одним из первых в конце 70-х годов XX века. Его название составлено из первых букв фамилий авторов: Р.Райвеста (R.Rivest), А.Шамира (A.Shamir) и Л.Адлемана (L.Adleman). Алгоритм RSA является, наверно, наиболее популярным и широко применяемым асимметричным алгоритмом в криптографических системах.
Алгоритм основан на использовании того факта, что задача разложения большого числа на простые сомножители является трудной. Криптографическая система RSA базируется на следующих двух фактах из теории чисел:

1) задача проверки числа на простоту является сравнительно легкой;
2) задача разложения чисел вида n = pq (p и q — простые числа); на множители является очень трудной, если мы знаем только n, а p и q — большие числа.

Алгоритм RSA представляет собой блочный алгоритм шифрования, где зашифрованные и незашифрованные данные должны быть представлены в виде целых чисел между 0 и n - 1 для некоторого n.
Шифрование
Итак, рассмотрим сам алгоритм. Пусть абонент А хочет передать зашифрованное сообщение абоненту Б. В этом случае абонент Б должен подготовить пару (открытый ключ; закрытый ключ) и отправить свой открытый ключ пользователю А.
Первым этапом является генерация открытого и закрытого ключей. Для этого вначале выбираются два больших простых числа P и Q. Затем вычисляется произведение N:
$N = PQ$.
После этого определяется вспомогательное число f: $f = (P - l)(Q - 1)$.
Затем случайным образом выбирается число $d < f$ и взаимно простое с f.
Далее необходимо найти число e, такое, что $ed \bmod f = 1$.
Числа d и N будут открытым ключом пользователя, а значение e – закрытым ключом.
Таким образом, на этом этапе у пользователя должна быть информация, указанная в следующей таблице:

	Открытый ключ	Закрытый ключ
Пользователь системы	N, d	e

Так как пользователь Б хочет получить зашифрованное сообщение от пользователя А, значит пользователь Б должен отправить свой открытый ключ (d, N) пользователю А. Числа P и Q больше не нужны, однако их нельзя никому сообщать; лучше всего их вообще забыть.
На этом этап подготовки ключей закончен и можно использовать основной протокол RSA для шифрования данных.
Второй этап – шифрование данных. Если абонент А хочет передать некоторые данные абоненту Б, он должен представить свое сообщение в цифровом виде и разбить его на блоки $m_1, m_2, m_3, \ldots$, где $m_i < N$. Зашифрованное сообщение будет состоять из блоков c_i.
Абонент А шифрует каждый блок своего сообщения по формуле
$c_i = m_i^d \bmod N$ используя открытые параметры пользователя Б, и пересылает зашифрованное сообщение $C=(c_1, c_2, c_3, \ldots)$ по открытой линии.
Абонент Б, получивший зашифрованное сообщение, расшифровывает все блоки полученного сообщения по формуле $m_i = c^e \bmod N$
Все расшифрованные блоки будут точно такими же, как и исходящие от пользователя А.

Злоумышленник, перехватывающий все сообщения и знающий всю открытую информацию, не сможет найти исходное сообщение при больших значениях P и Q.

Пример

Пусть пользователь А хочет передать пользователю Б сообщение. В этом случае вначале пользователь Б должен подготовить открытый и закрытый ключи. Пусть им выбраны, например, следующие параметры:

P = 3, Q = 11, N = 3x11 = 33.

Тогда f = (P - l)(Q - 1) = (3-1)(11-1) = 20.

Затем пользователь Б выбирает любое число d, не имеющее общих делителей с f (это необходимо для того, чтобы зашифрованное сообщение можно было потом однозначно восстановить). Пусть d = 13. Это число будет одним из компонентов открытого ключа.

Далее необходимо найти число e, которое можно будет использовать в качестве закрытого ключа для расшифрования сообщения. Значение e должно удовлетворять соотношению

ed mod f = 1.

Для малых значений f число e можно найти подбором. В общем случае для поиска e можно использовать обобщенный алгоритм Евклида, приведенный в "Основные положения теории чисел, используемые в криптографии с открытым ключом" . В нашем случае подходит e=17. (Проверяем: 13*17 mod 20 = 221 mod 20 = 1.)

Теперь пользователь Б должен запомнить свой закрытый ключ 17, отправить открытый ключ (13, 33) пользователю А и уничтожить числа P = 3 и Q = 11.

Пользователь А, получивший открытый ключ (13, 33), увидев, что N=33, разбивает исходное сообщение на три блока, причем значение каждого меньше N. Например, пусть имеется три блока m_1=8, m_2=27, m_3,=5. Затем пользователь А шифрует каждый блок:

$c_1=8^{13} \bmod 33 = 17$
$c_2 = 27^{13} \bmod 33 = 15$
$c_3 = 5^{13} \bmod 33 = 26$

Зашифрованное сообщение, состоящее из трех блоков (17, 15, 26), передается пользователю Б, который, используя свой закрытый ключ e = 17 и N=33, расшифровывает сообщение:

$m_1 = 17^{17} \bmod 33 = 8$
$m_2 = 15^{17} \bmod 33 = 27$
$m_3 = 26^{17} \bmod 33 = 5$

Таким образом, абонент Б расшифровал сообщение от абонента А.

Задание: Реализовать асимметричный алгоритм RSA на любом из языков программирования.

Содержание отчета: Текст программной реализации асимметричного алгоритма RSA на любом языке программирования с подробными комментариями и тестовыми данными.

Контрольные вопросы:

1) Для каких целей может применяться алгоритм RSA?
2) Опишите процесс шифрования с использованием алгоритма RSA.

Список литературы

Интернет ресурс: НОУ Интуит : https://intuit.ru/studies/courses/552/408/info

Практическая работа № 7

Тема: Аутентификация данных. Электронная подпись

Цель:

Изучить:

- применение различных функций хеширования, анализ особенностей хешей;
- программно-аппаратные средства, реализующих основные функции ЭП

Формируемые компетенции: ПК 2.1, ПК 2.4, ОК 1, ОК 2, ОК 5, ОК 6.

Пояснения к работе:

Обзор алгоритмов хеш-функций

В настоящее время предложены и практически используются различные специальные алгоритмы для вычисления хеш-функции. Наиболее известными алгоритмами являются MD5, SHA-1, SHA-2 и другие версии SHA, а также отечественный алгоритм, изложенный в ГОСТ Р 34.11-94.

Алгоритм MD5 появился в начале 90-х годов XX века в результате усовершенствования алгоритма формирования хеш-функции MD4. Символы в названии "MD" означают Message Digest – краткое изложение сообщения. Автор алгоритмов MD4 и MD5 – Р. Ривест (R.Rivest). В результате использования MD5 для произвольного сообщения формируется 128-битное хеш-значение. Входные данные обрабатываются блоками по 512 бит. В алгоритме используются элементарные логические операции (инверсия, конъюнкция, сложение по модулю 2, циклические сдвиги и др.), а также обыкновенное арифметическое сложение. Комплексное повторение этих элементарных функций алгоритма обеспечивает то, что результат после обработки хорошо перемешан. Поэтому маловероятно, чтобы два сообщения, выбранные случайно, имели одинаковый хеш-код. Алгоритм MD5 имеет следующее свойство: каждый бит полученного хеш-значения является функцией от каждого бита входа. Считается, что MD5 является наиболее сильной хеш-функцией для 128-битного хеш-значения.

Алгоритм SHA (Secure Hash Algorithm – Безопасный хеш-алгоритм) был разработан национальным институтом стандартов и технологии (NIST) США и опубликован в качестве американского федерального информационного стандарта в 1993 году. SHA-1, как и MD5, основан на алгоритме MD4. SHA-1 формирует 160-битное хеш-значение на основе обработки исходного сообщения блоками по 512 бит. В алгоритме SHA-1 также используются простые логические и арифметические операции. Наиболее важным отличием SHA-1 от MD5 является то, что хеш-код SHA-1 на 32 бита длиннее, чем хеш-код MD5. Если предположить, что оба алгоритма одинаковы по сложности для криптоанализа, то SHA-1 является более стойким алгоритмом. Используя атаку методом грубой силы (лобовую атаку), труднее создать произвольное сообщение, имеющее данный хеш-код, а также труднее создать два сообщения, имеющие одинаковый хеш-код.

В 2001 году национальный институт стандартов и технологии США принял в качестве стандарта три хеш-функции с большей длиной хеш-кода, чем у SHA-1. Часто эти хеш-функции называют SHA-2 или SHA-256, SHA-384 и SHA-512 (в названии указывается длина создаваемого алгоритмами хеш-кода). Эти алгоритмы отличаются не только длиной создаваемого хеш-кода, но и используемыми внутренними функциями и длиной обрабатываемого блока (у SHA-256 длина блока – 512, а у SHA-384 и SHA-

512 длина блока – 1024 бита). Постепенные усовершенствования алгоритма SHA ведут к увеличению его криптостойкости. Несмотря на отличия рассматриваемых алгоритмов друг от друга, все они являются дальнейшим развитием SHA-1 и MD4 и имеют похожую структуру.

В России принят ГОСТ Р34.11-94, который является отечественным стандартом для хеш-функций. Его структура довольно сильно отличается от структуры алгоритмов SHA-1,2 или MD5, в основе которых лежит алгоритм MD4. Длина хеш-кода, создаваемого алгоритмом ГОСТ Р 34.11-94, равна 256 битам. Алгоритм последовательно обрабатывает исходное сообщение блоками по 256 бит справа налево. Параметром алгоритма является стартовый вектор хеширования – произвольное фиксированное значение длиной также 256 бит. В алгоритме ГОСТ Р 34.11-94 используются операции перестановки, сдвига, арифметического сложения, сложения по модулю 2. В качестве вспомогательной функции в ГОСТ 34.11-94 используется алгоритм по ГОСТ 28147-89 в режиме простой замены.

Электронная подпись на основе алгоритма RSA

Схема использования алгоритма RSA при большом модуле N практически не позволяет злоумышленнику получить закрытый ключ и прочитать зашифрованное сообщение. Однако она дает возможность злоумышленнику подменить сообщение от абонента А к абоненту Б, так как абонент А шифрует свое сообщение открытым ключом, полученным от Б по открытому каналу связи. А раз открытый ключ передается по открытому каналу, любой может получить его и использовать для подмены сообщения. Избежать этого можно, используя более сложные протоколы, например, следующий.

Пусть, как и раньше, пользователь А хочет передать пользователю Б сообщение, состоящее из нескольких блоков m_i. Перед началом сеанса связи абоненты генерируют открытые и закрытые ключи, обозначаемые, как указано в следующей таблице:

	Открытый ключ	Закрытый ключ
Пользователь А	N_A, d_A	e_A
Пользователь Б	$N_Б$, $d_Б$	$e_Б$

В результате каждый пользователь имеет свои собственные открытый (состоящий из двух частей) и закрытый ключи. Затем пользователи обмениваются открытыми ключами. Это подготовительный этап протокола.

Основная часть протокола состоит из следующих шагов.

1. Сначала пользователь А вычисляет числа $c_i = m_i^{e_A} \, mod \, N_A$, то есть шифрует сообщение своим закрытым ключом. В результате этих действий пользователь А подписывает сообщение.
2. Затем пользователь А вычисляет числа $g_i = c_i^{d_Б} \, mod \, N_Б$, то есть шифрует то, что получилось на шаге 1 открытым ключом пользователя Б. На этом этапе сообщение шифруется, чтобы никто посторонний не мог его прочитать.
3. Последовательность чисел g_i передается к пользователю Б.
4. Пользователь Б получает g_i и вначале вычисляет последовательно числа $c_i = g_i^{e_Б} \, mod \, N_Б$, используя свой закрытый ключ. При этом сообщение расшифровывается.

5. Затем Б определяет числа $m_i = c_i^{d_A} \, mod \, N_A$, используя открытый ключ пользователя А. За счет выполнения этого этапа производится проверка подписи пользователя А.

В результате абонент Б получает исходное сообщение и убеждается в том, что его отправил именно абонент А. Данная схема позволяет защититься от нескольких видов возможных нарушений, а именно:

- пользователь А не может отказаться от своего сообщения, если он признает, что секретный ключ известен только ему;
- нарушитель без знания секретного ключа не может ни сформировать, ни сделать осмысленное изменение сообщения, передаваемого по линии связи.

Данная схема позволяет избежать многих конфликтных ситуаций. Иногда нет необходимости зашифровывать передаваемое сообщение, но нужно его скрепить электронной подписью. В этом случае из приведенного выше протокола исключаются шаги 2 и 4, то есть текст шифруется закрытым ключом отправителя, и полученная последовательность присоединяется к документу. Получатель с помощью открытого ключа отправителя расшифровывает прикрепленную подпись, которая, по сути, является зашифрованным повторением основного сообщения. Если расшифрованная подпись совпадает с основным текстом, значит, подпись верна.

Существуют и другие варианты применения алгоритма RSA для формирования ЭЦП. Например, можно шифровать (то есть подписывать) открытым ключом не само сообщение, а хеш-код от него.

Возможность применения алгоритма RSA для получения электронной подписи связана с тем, что секретный и открытый ключи в этой системе равноправны. Каждый из ключей, d или e, могут использоваться как для шифрования, так и для расшифрования. Это свойство выполняется не во всех криптосистемах с открытым ключом.

Алгоритм RSA можно использовать также и для обмена ключами.

Задание:

1) Реализовать на любом из языков программирования любую из описанных выше хеш-функции.
2) Подготовить доклад о программно-аппаратном средстве, реализующем основные функции ЭЦП.

Содержание отчета:

1) Текст программной реализации хеш-функции на любом языке программирования с подробными комментариями и тестовыми данными.
2) Текст доклада о программно-аппаратном средстве, реализующем основные функции ЭЦП.

Контрольные вопросы:

1) Что в криптографии называется хеш-функцией?
2) Для каких целей используются хеш-функции?
3) Перечислите основные требования, предъявляемые к хеш-функциям.
4) Назовите примеры криптографических хеш-функций.

5) Каков российский стандарт на алгоритм формирования криптографической хеш-функции?
6) Каким образом можно использовать блочный алгоритм шифрования для формирования хеш-функции?

Список литературы:

Интернет ресурс: НОУ Интуит : https://intuit.ru/studies/courses/691/547/info

Практическая работа № 8

Тема: Алгоритмы обмена ключей и протоколы аутентификации

Цель: Изучение принципов работы протоколов аутентификации с использованием доверенной стороны на примере протокола Kerberos.

Формируемые компетенции: ПК 2.1, ПК 2.2, ОК 1, ОК 2, ОК 5, ОК 6.

Пояснения к работе:

Протокол Kerberos
Протокол Kerberos был разработан в Массачусетском технологическом институте в середине 1980-х годов и сейчас является фактическим стандартом системы централизованной аутентификации и распределения ключей симметричного шифрования. Поддерживается операционными системами семейства Unix, Windows (начиная с Windows'2000), есть реализации для Mac OS.
В сетях Windows (начиная с Windows'2000 Serv.) аутентификация по протоколу Kerberos v.5 (RFC 1510) реализована на уровне доменов. Kerberos является основным протоколом аутентификации в домене, но в целях обеспечения совместимости с с предыдущими версиями, также поддерживается протокол NTLM.
Перед тем, как рассмотреть порядок работы Kerberos, разберем зачем он изначально разрабатывался. Централизованное распределение ключей симметричного шифрования подразумевает, что у каждого абонента сети есть только один основной ключ, который используется для взаимодействия с центром распределения ключей (сервером ключей). Чтобы получить ключ шифрования для защиты обмена данными с другим абонентом, пользователь обращается к серверу ключей, который назначает этому пользователю и соответствующему абоненту сеансовый симметричный ключ.
Протокол Kerberos обеспечивает распределение ключей симметричного шифрования и проверку подлинности пользователей, работающих в незащищенной сети. Реализация Kerberos - это программная система, построенная по архитектуре "клиент-сервер". Клиентская часть устанавливается на все компьютеры защищаемой сети, кроме тех, на которые устанавливаются компоненты сервера Kerberos. В роли клиентов Kerberos могут, в частности, выступать и сетевые серверы (файловые серверы, серверы печати и т.д.).
Серверная часть Kerberos называется центром распределения ключей (англ. Key Distribution Center, сокр. KDC) и состоит из двух компонент:

- сервер аутентификации (англ. Authentication Server, сокр. AS);
- сервер выдачи разрешений (англ. Ticket Granting Server, сокр. TGS).

Каждому субъекту сети сервер Kerberos назначает разделяемый с ним ключ симметричного шифрования и поддерживает базу данных субъектов и их секретных ключей. Схема функционирования протокола Kerberos представлена на рис. 9.

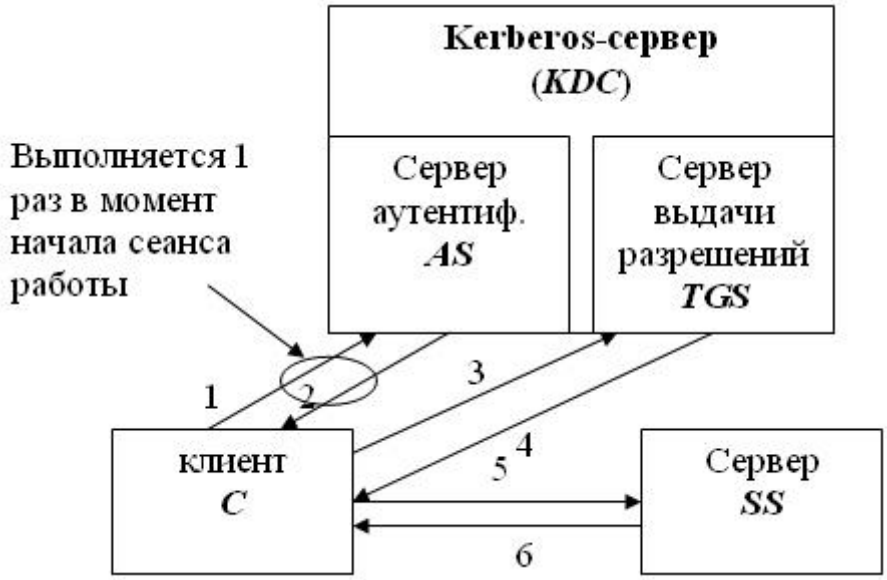


Рис. 9. Протокол Kerberos

Пусть клиент C собирается начать взаимодействие с сервером SS (англ. Service Server - сервер, предоставляющий сетевые сервисы). В несколько упрощенном виде, протокол предполагает следующие шаги:

1. C->AS: {c}.
   Клиент C посылает серверу аутентификации AS свой идентификатор c (идентификатор передается открытым текстом).
2. AS->C: {{TGT}K_{AS_TGS}, K_{C_TGS}}K_C,
   где:
   - K_C - основной ключ C ;
   - K_{C_TGS} - ключ, выдаваемый C для доступа к серверу выдачи разрешений TGS ;
   - {TGT} - Ticket Granting Ticket - билет на доступ к серверу выдачи разрешений

   {TGT}={c,tgs,t_1,p_1, K_{C_TGS}}, где tgs - идентификатор сервера выдачи разрешений, t_1 - отметка времени, p_1 - период действия билета.

   Запись $\{\cdot\}K_X$ здесь и далее означает, что содержимое фигурных скобок зашифровано на ключе K_X.

   На этом шаге сервер аутентификации AS, проверив, что клиент C имеется в его базе, возвращает ему билет для доступа к серверу выдачи разрешений и ключ для взаимодействия с сервером выдачи разрешений. Вся посылка зашифрована на ключе клиента C. Таким образом, даже если на первом шаге взаимодействия идентификатор c послал не клиент C, а нарушитель X, то полученную от AS посылку X расшифровать не сможет.

   Получить доступ к содержимому билета TGT не может не только нарушитель, но и клиент C, т.к. билет зашифрован на ключе, который распределили между собой сервер аутентификации и сервер выдачи разрешений.
3. C->TGS: {TGT}K_{AS_TGS}, {Aut_1} K_{C_TGS}, {ID}
   где {Aut_1} - аутентификационный блок - Aut_1 = {c,t_2}, t_2 - метка времени; ID - идентификатор запрашиваемого сервиса (в частности, это может быть идентификатор сервера SS).

   Клиент C на этот раз обращается к серверу выдачи разрешений TGS. Он пересылает полученный от AS билет, зашифрованный на ключе K_{AS_TGS}, и аутентификационный блок, содержащий идентификатор c и метку времени, показывающую, когда была сформирована посылка.Сервер выдачи разрешений расшифровывает билет TGT и получает из него информацию о том, кому был выдан билет, когда и на какой срок,

ключ шифрования, сгенерированный сервером AS для взаимодействия между клиентом C и сервером TGS. С помощью этого ключа расшифровывается аутентификационный блок. Если метка в блоке совпадает с меткой в билете, это доказывает, что посылку сгенерировал на самом деле C (ведь только он знал ключ K_{C_TGS} и мог правильно зашифровать свой идентификатор). Далее делается проверка времени действия билета и времени отправления посылки 3). Если проверка проходит и действующая в системе политика позволяет клиенту C обращаться к клиенту SS, тогда выполняется шаг 4).

4. TGS->C: $\{\{TGS\}K_{TGS_SS},K_{C_SS}\}K_{C_TGS}$,
   где K_{C_SS} - ключ для взаимодействия C и SS, {TGS} - Ticket Granting Service - билет для доступа к SS (обратите внимание, что такой же аббревиатурой в описании протокола обозначается и сервер выдачи разрешений). $\{TGS\} =\{c,ss,t_3,p_2, K_{C_SS}\}$.
   Сейчас сервер выдачи разрешений TGS посылает клиенту C ключ шифрования и билет, необходимые для доступа к серверу SS. Структура билета такая же, как на шаге 2): идентификатор того, кому выдали билет; идентификатор того, для кого выдали билет; отметка времени; период действия; ключ шифрования.
5. C->SS: $\{TGS\}K_{TGS_SS}$, $\{Aut_2\}\ K_{C_SS}$
   где $Aut_2=\{c,t_4\}$.
   Клиент C посылает билет, полученный от сервера выдачи разрешений, и свой аутентификационный блок серверу SS, с которым хочет установить сеанс защищенного взаимодействия. Предполагается, что SS уже зарегистрировался в системе и распределил с сервером TGS ключ шифрования K_{TGS_SS}. Имея этот ключ, он может расшифровать билет, получить ключ шифрования K_{C_SS} и проверить подлинность отправителя сообщения.
6. SS->C: $\{t_4+1\}K_{C_SS}$
   Смысл последнего шага заключается в том, что теперь уже SS должен доказать C свою подлинность. Он может сделать это, показав, что правильно расшифровал предыдущее сообщение. Вот поэтому, SS берет отметку времени из аутентификационного блока C, изменяет ее заранее определенным образом (увеличивает на 1), шифрует на ключе K_{C_SS} и возвращает C.

Если все шаги выполнены правильно и все проверки прошли успешно, то стороны взаимодействия C и SS, во-первых, удостоверились в подлинности друг друга, а во-вторых, получили ключ шифрования для защиты сеанса связи - ключ K_{C_SS}.

Нужно отметить, что в процессе сеанса работы клиент проходит шаги 1) и 2) только один раз. Когда нужно получить билет на доступ к другому серверу (назовем его SS1), клиент C обращается к серверу выдачи разрешений TGS с уже имеющимся у него билетом, т.е. протокол выполняется начиная с шага 3).

При использовании протокола Kerberos компьютерная сеть логически делится на области действия серверов Kerberos. Kerberos-область - это участок сети, пользователи и серверы которого зарегистрированы в базе данных одного сервера Kerberos (или в одной базе, разделяемой несколькими серверами). Одна область может охватывать сегмент локальной сети, всю локальную сеть или объединять несколько связанных локальных сетей. Схема взаимодействия между Kerberos-областями представлена на рис. 5.2.

Для взаимодействия между областями, должна быть осуществлена взаимная регистрация серверов Kerberos, в процессе которой сервер выдачи разрешений одной области регистрируется в качестве клиента в другой области (т.е. заносится в базу сервера аутентификации и разделяет с ним ключ).

После установки взаимных соглашений, клиент из области 1 (пусть это будет K_{11}) может установить сеанс взаимодействия с клиентом из области 2 (например, K_{21}). Для этого K_{11} должен получить у своего сервера выдачи разрешений билет на доступ к Kerberos-серверу, с клиентом которого он хочет установить взаимодействие (т.е. серверу Kerberos KDC2). Полученный билет содержит отметку о том, в какой области зарегистрирован владелец билета. Билет шифруется на ключе, разделенном между серверами KDC1 и KDC2. При успешной расшифровке билета, удаленный Kerberos-

сервер может быть уверен, что билет выдан клиенту Kerberos-области, с которой установлены доверительные отношения. Далее протокол работает как обычно.

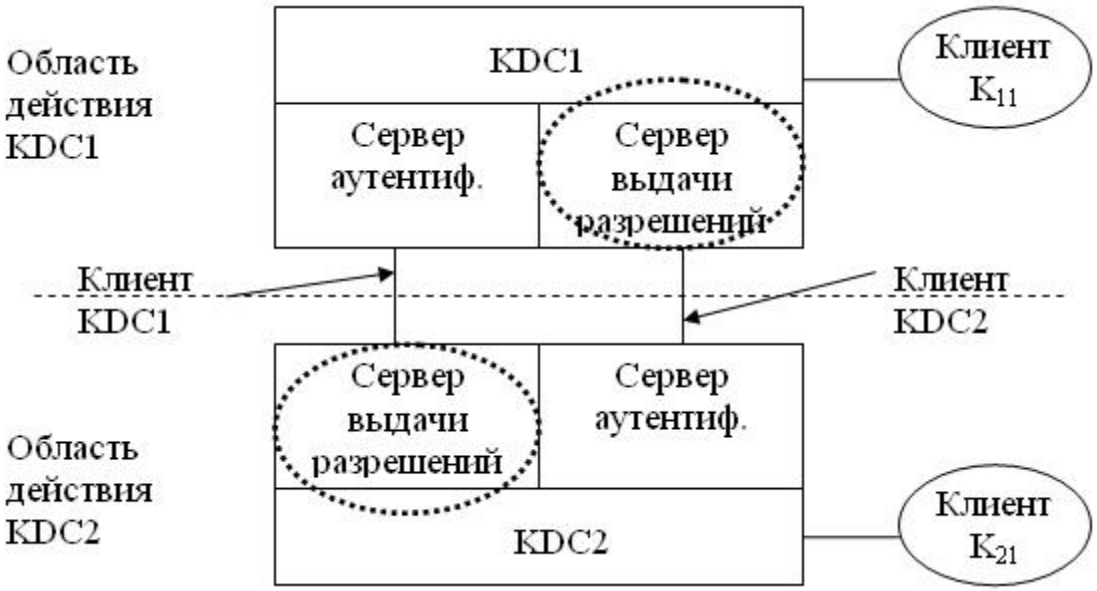


Рис. 10. Взаимодействие между Kerberos-областями

Кроме рассмотренных, Kerberos предоставляет еще ряд дополнительных возможностей. Например, указанный в структуре билета параметр p (период времени) задается парой значений "время начала действия" - "время окончания действия", что позволяет получать билеты отложенного действия.

Имеется тип билета "с правом передачи", что позволяет, например, серверу выполнять действия от имени обратившегося к нему клиента.

Два слова об аутентификации. Если Kerberos - протокол распределения ключей, корректно ли использовать его для аутентификации?! Но как отмечалось выше, если все стадии протокола прошли успешно, взаимодействующие стороны не только распределили ключ, но и убедились в подлинности друг друга, иными словами - аутентифицировали друг друга.

Что касается реализации протокола Kerberos в Windows, то надо отметить следующее.

1. Ключ пользователя генерируется на базе его пароля. Таким образом, при использовании слабых паролей эффект от надежной защиты процесса аутентификации будет сведен к нулю.
2. В роли Kerberos-серверов выступают контроллеры домена, на каждом из которых должна работать служба Kerberos Key Distribution Center (KDC). Роль хранилища информации о пользователях и паролях берет на себя служба каталога Active Directory. Ключ, который разделяют между собой сервер аутентификации и сервер выдачи разрешений формируется на основе пароля служебной учетной записи krbtgt - эта запись автоматически создается при организации домена и всегда заблокирована.
3. Microsoft в своих ОС использует расширение Kerberos для применения криптографии с открытым ключом. Это позволяет осуществлять регистрацию в домене и с помощью смарт-карт, хранящих ключевую информацию и цифровой сертификат пользователя.
4. Использование Kerberos требует синхронизации внутренних часов компьютеров, входящих в домен Windows.

Для увеличения уровня защищенности процесса аутентификации пользователей, рекомендуется отключить использование менее надежного протокола NTLM и оставить только Kerberos (если использования NTLM не требуют устаревшие клиентские ОС).

Кроме того, рекомендуется запретить администраторским учетным записям получать билеты "с правом передачи" (это убережет от некоторых угроз, связанных автоматическим запуском приложений от имени таких записей).

Задание:

1) Найти уязвимость в схеме протокола
2) Определить, атака какого типа позволит воспользоваться выявленной уязвимостью протокола
3) Предложить способ модифицировать протокол, чтобы устранить уязвимость

Описание схемы

Смарт-карты как на программном, так и на аппаратном уровне поддерживают множество алгоритмов симметричного шифрования и шифрования с открытым ключом. Смарт-карта оснащена собственным криптопроцессором и может выполнять операции шифрования и дешифрации без участия хост-компьютера, что и позволяет осуществлять безопасный обмен данными через Internet. Информация, которую необходимо предоставить пользователям, накапливается в базе данных сервера, а на компьютере пользователя установлено специальное клиентское ПО для доступа к этой информации.

Пользователь должен ввести PIN -код для взаимной аутентификации Internet -сервера и карты. Так как карта была эмитирована учебным заведением, в ней хранится открытый ключ сервера, а серверу известен открытый ключ карты. Сам процесс аналогичен взаимной аутентификации карты и хост-компьютера: карта передает серверу случайное число, зашифрованное его открытым ключом. Сервер должен его расшифровать и передать назад. Если число расшифровано правильно, карта может доверять серверу (рисунок 11).

Аналогично происходит аутентификация карты сервером. Такая процедура позволяет обеспечить доступ к данным легитимным пользователям. После того как подлинность карты и сервера по отношению друг к другу установлена, происходит обмен сессионными ключами. Например, ключи генерируются на сервере, шифруются открытым ключом карты и передаются карте, которая их расшифровывает и сохраняет. Далее смарт-карта шифрует запрос пользователя сессионным ключом и передает серверу. Сервер расшифровывает запрос, формирует ответ, шифрует их другим сессионным ключом и передает назад. Карта расшифровывает полученный ответ, который затем отображается на компьютере клиента. При передаче следующей порции данных будет использован новый сессионный ключ.

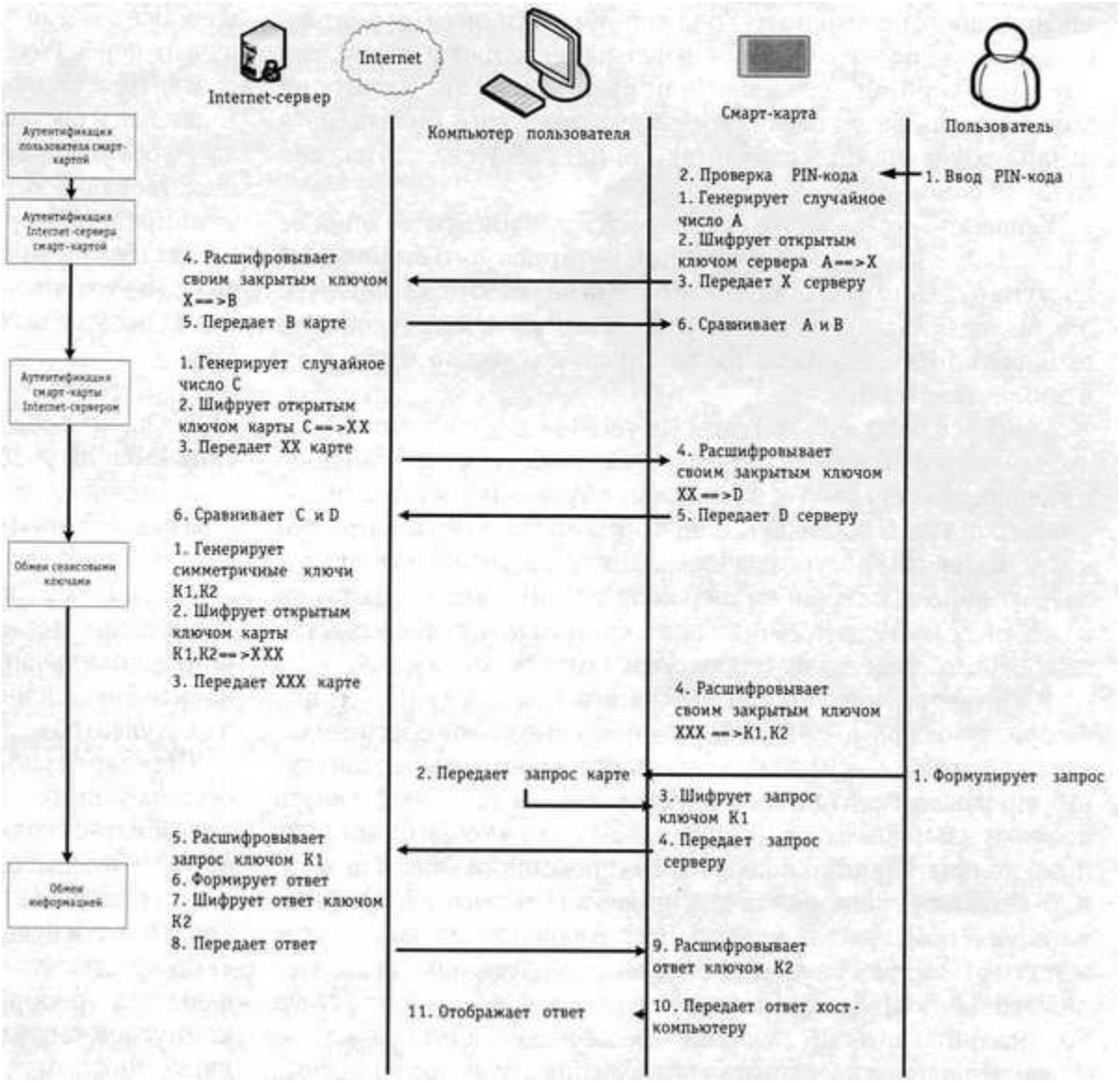


Рис. 11. Схема доступа смарт-карты к Internet

Пояснение

Протокол подвержен атаке типа "человек посередине". На этапе "Обмен сеансовыми ключами " сервер не предоставляет никакого подтверждения того, что именно он сгенерировал зашифрованные на открытом ключе смарт-карты. Сообщение от сервера может быть изъято злоумышленником и заменено на другие ключи симметричные K3, K4, зашифрованные на открытом ключе карты (согласно принципу асимметричной криптографии, открытый ключ известен всем, в т.ч. и потенциальному злоумышленнику). После этого смарт-карта передаст запрос серверу, зашифровав его на ключе K3. Злоумышленник сможет перехватить и расшифровать его, чтобы сформировать ответ и зашифровать на ключе K4. Таким образом, смарт-карта будет взаимодействовать не с сервером, а со злоумышленником. Избежать этого можно, если на этапе "Обмен сеансовыми ключами " в дополнение к передаваемому сообщению XXX сервер передаст результат шифрования XXX на своем закрытом ключе XXXX (по сути, это будет служить цифровой подписью ключей). Тогда эквивалентность XXX и результата расшифрования XXXX на открытом ключе сервера будет для смарт-карты гарантией того, что ключ был получен от сервера, а не от злоумышленника.

Содержание отчета:

1) Описание уязвимости в схеме протокола
2) Описание типа атаки
3) Описание способа модификации протокола

Контрольные вопросы:

1) Опишите основные этапы работы протокола Kerberos
2) Где применяется протокол Kerberos

Список литературы:

Интернет ресурс: НОУ Интуит : https://intuit.ru/studies/courses/691/547/info

Практическая работа № 9

Тема: Защита информации в электронных платежных системах

Цель: Изучение применения аутентификации по одноразовым паролям и изучение реализации алгоритмов создания одноразовых паролей

Формируемые компетенции: ПК 2.1, ПК 2.2, ПК 2.4, ОК 1, ОК 2, ОК 5, ОК 6.

Пояснения к работе:

Наиболее известным программным генератором одноразовых паролей является система S/KEY компании Bellcore. Идея этой системы состоит в следующем. Пусть имеется односторонняя функция f (то есть функция, вычислить обратную которой за приемлемое время не представляется возможным). Эта функция известна и пользователю, и серверу аутентификации. Пусть, далее, имеется секретный ключ K, известный только пользователю.

На этапе начального администрирования пользователя функция f применяется к ключу K n раз, после чего результат сохраняется на сервере. После этого процедура проверки подлинности пользователя выглядит следующим образом:

- сервер присылает на пользовательскую систему число (n-1);
- пользователь применяет функцию f к секретному ключу K (n-1) раз и отправляет результат по сети на сервер аутентификации;
- сервер применяет функцию f к полученному от пользователя значению и сравнивает результат с ранее сохраненной величиной. В случае совпадения подлинность пользователя считается установленной, сервер запоминает новое значение (присланное пользователем) и уменьшает на единицу счетчик (n).

На самом деле реализация устроена чуть сложнее (кроме счетчика, сервер посылает затравочное значение, используемое функцией f), но для нас сейчас это не важно. Поскольку функция f необратима, перехват пароля, равно как и получение доступа к серверу аутентификации, не позволяют узнать секретный ключ K и предсказать следующий одноразовый пароль.

Система S/KEY имеет статус Internet-стандарта (RFC 1938).

Другой подход к надежной аутентификации состоит в генерации нового пароля через небольшой промежуток времени (например, каждые 60 секунд), для чего могут использоваться программы или специальные интеллектуальные карты (с практической точки зрения такие пароли можно считать одноразовыми). Серверу аутентификации должен быть известен алгоритм генерации паролей и ассоциированные с ним параметры; кроме того, часы клиента и сервера должны быть синхронизированы.

Задание: Реализовать алгоритм создания одноразовых паролей на любом из языков программирования.

Содержание отчета: Текст программной реализации создания одноразовых паролей на любом из языков программирования с подробными комментариям кода и тестовыми данными.

Контрольные вопросы:

1. Для чего нужны одноразовые пароли и где они применяются?

Список литературы:

Интернет ресурс: НОУ Интуит : https://intuit.ru/studies/professional_retraining/952/info

Практическая работа № 10

Тема: Компьютерная стеганография

Цель: Реализация простейших стеганографических алгоритмов

Формируемые компетенции: ПК 2.1, ПК 2.2, ПК 2.4, ОК 1, ОК 2, ОК 5, ОК 6.

Пояснения к работе:

F5 - один из алгоритмов стеганографии для встраивания данных в изображения JPEG.

В упрощенном виде алгоритм F5 имеет следующую структуру:

1. Начать сжатие JPEG. Остановиться после квантования коэффициентов.
2. Инициализировать криптографически стойкий генератор случайных чисел с помощью ключа, полученного из пароля.
3. Создать перестановку с двумя параметрами – генератор случайных чисел и количество коэффициентов (включая нулевые коэффициенты).
4. Определить параметр k, который зависит от емкости несущей среды, а также от длины секретного сообщения.
5. Вычислить длину кодового слова $n=2k-1$.
6. Вставить секретное сообщение, используя (1,n,k) кодирование матрицы.
    1. Заполнить буфер n ненулевыми коэффициентами.
    2. Хешировать буфер (сгенерированное значение хеш-функции состоит из k бит).
    3. Добавить следующие k битов сообщения к значению хеша (побитно с помощью операции XOR).
    4. Если сумма равна 0, то буфер остается неизменным. В противном случае, сумма указывает на элемент буфера, абсолютное значение которого нужно уменьшить на единицу.
    5. При получении нуля проверяется сокращение. Если сокращение имеет место, то настроить буфер (исключить 0, прочитав еще один ненулевой коэффициент, т. е. повторить шаг 6.1, начиная с того же коэффициента). Если сокращения не произошло, осуществляется переход к новым коэффициентам фактического буфера. Если еще есть данные сообщения, перейти к шагу 6.1.
7. Продолжить сжатие JPEG (кодирование Хаффмана и т.д.).

Задание: Реализовать стеганографический алгоритм на любом из языков программирования.

Содержание отчета: Текст программной реализации стеганографического алгоритма на любом из языков программирования с подробными комментариям кода и тестовыми данными.

Контрольные вопросы:

Список литературы:

Интернет ресурс: НОУ Интуит : https://intuit.ru/studies/professional_retraining/952/info