МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ "ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ - УЧЕБНО-НАУЧНО-ПРОИЗВОДСТВЕННЫЙ КОМПЛЕКС"
УЧЕБНО-НАУЧНО-ИССЛЕДОВАТЕЛЬСКИЙ ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Кафедра «Электроника, вычислительная техника и информационная безопасность»

В. Т. Еременко, М. Ю. Рытов, П. Н. Рязанцев

УЧЕБНОЕ ПОСОБИЕ

МАТЕМАТИЧЕСКИЕ ОСНОВЫ КРИПТОЛОГИИ.

Дисциплины – Б.3.А.15 «Криптографические методы защиты информации», Б.3.В.2.1 «Основы криптографии»

Направление подготовки 090900.62 Информационная безопасность
Направление подготовки 210700.62 Инфокоммуникационные технологии и систем связи

Орёл 2014

УДК 621.391

Авторы: д.т.н., профессор кафедры «ЭВТИБ» В.Т. Еременко, к.т.н., доцент БГТУ кафедры «Системы информационной безопасности» М. Ю. Рытов, ассистент кафедры «ЭВТИБ» П. Н. Рязанцев

Рецензент: к.т.н., профессор кафедры «ЭВТИБ» В. А. Лобанова

Учебное пособие «Математические основы криптографии» предназначено для бакалавров, обучающихся по направлению подготовки 090900.62 «Информационная безопасность» и 210700.62 «Инфокоммуникационные технологии и систем связи», изучающих дисциплины Б.3.А.15 «Криптографические методы защиты информации», Б.3.В.2.1 «Основы криптографии» и смежных с ними.

Учебное пособие «Математические основы криптографии» рассмотрено и одобрено

на заседании кафедры ЭВТИБ «_____»_____2014 г., протокол №___,

зав. кафедрой ЭВТИБ, д.т.н., проф. _____ В.Т. Еременко;

на заседании УМС УНИИ ИТ «___»_____2014 г., протокол № ____

председатель УМС УНИИ ИТ, д.т.н., проф. _____ К.В.Подмастерьев

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Пособие построено на принципе постепенного усложнения излагаемого материала. В случае недостаточного понимания текущего раздела, необходимо вернуться к повторению материалов предыдущих разделов.

Все примеры решения в пособии предваряются теоретическим введением с подробными комментариями. Непосредственно в решении приведены все промежуточные результаты, а комментарии наоборот - полностью отсутствуют.

Оформление заданий при их выполнении оставляется на усмотрение преподавателя.

Ниже приведены примеры обозначений, используемых при описании алгоритмов и заданий [10]:

C_i, P, G - открытые параметры, которые могут быть известны противнику обозначаются заглавными символами;

k, x, q - секретные параметры, известные только корреспондентам (или только одному корреспонденту) обозначаются строчными символами;

$\tilde{H}$ – пример обозначения величины, подбираемой на данном этапе вычислений случайным или произвольным образом;

m, m' – исходный и восстановленный, например, при расшифровании, параметры;

(mod N) - все операции выполняются по модулю (в остатках от деления) числа N.

Для лучшего понимания в последующем принципов программной реализации криптографических алгоритмов с большими числами все решения производить с последовательным приведением промежуточных значений по заданному модулю. В случае возникновения отрицательного промежуточного результата – тут же приводить его (прибавлять к нему модуль). Желательно, чтобы в решении не приводились отрицательные значения и значения, превышающие модуль по которому производятся вычисления.

Например, вместо:

(357 - 654) · 46 (mod 701) = - 13662 mod 701 = 358

Производим последовательные вычисления:

(357 - 654) · 46 (mod 701) = 404 · 46 (mod 701) = 358

При решении задач рекомендуется использование встроенного калькулятора Windows, а также для реализации некоторых операций - редактора электронных таблиц MS Excel или Open Office Calc. Примеры реализации операций, таких как: вычисление мультипликативного обратного, модульное экспоненцирование, приведены в соответствующих разделах пособия. Все задания строятся на использовании относительно небольших чисел. В случае значительного увеличения значений исходных данных рекомендуется написание программ, реализующих отдельные действия на одном из языков

программирования. Примеры исходных кодов таких программ на языках C# и Pascal приведены в приложении 2.

ЧАСТЬ 1. АРИФМЕТИКА ОСТАТКОВ. ЭЛЕМЕНТАРНЫЕ ШИФРЫ

Шифр Цезаря

Один из наиболее ранних и примитивных шифров простой замены. Изначально шифрование производилось при помощи записи двух алфавитов – открытого текста и шифртекста один под другим со смещением влево на величину сдвига k - ключа (см. рис.1). Считается, что Юлий Цезарь, которому приписывается авторство этого шифра, использовал ключ $k = 3$. При зашифровании символы заменяются сверху вниз, а при расшифровании, соответственно – снизу вверх.

A	**B**	**C**	**D**	**E**	**. . .**	**W**	**X**	**Y**	**Z**
D	**E**	**F**	**G**	**H**	**. . .**	**Z**	**A**	**B**	**C**

Рис.1. Шифр Цезаря

Для построения функций зашифрования и расшифрования этого шифра более удобным будет представление шифратора в виде двух соосных колец (см. рис.2), на каждом из которых нанесены N символов алфавита. Ключом является угол поворота k внутреннего кольца относительно внешнего, выраженный в количестве символов.

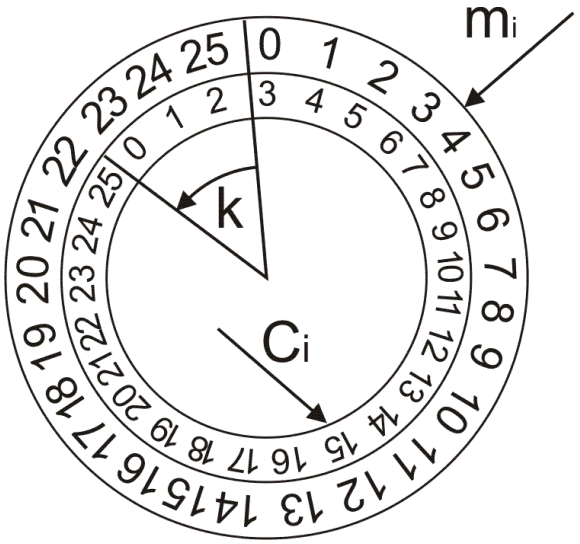

Рис.2. Дисковый шифратор

Для удобства перехода к математической модели вместо символов будем использовать их порядковые номера от 0 до N - 1. И зашифрование и расшифрование будем производить сверху-вниз. Для этого при зашифровании внутреннее кольцо будем

поворачивать на k делений против часовой стрелки, а при расшифровании на k делений по часовой стрелке, или на $N - k$ делений против часовой стрелки, что одно и то же.

Таким образом, количество возможных ключей шифра Цезаря – $(N - 1)$, за исключением сдвига на величину $0 \equiv N$. Математически выразим функции зашифрования и расшифрования для шифра Цезаря. Операцией наложения ключа является сложение по модулю N.

Функция зашифрования : $C_i = m_i + k \,(\mathrm{mod}\, N)$,

Функция расшифрования: $m_i = C_i + \bar{k} \,(\mathrm{mod}\, N)$,

где:

m_i – i-ый символ открытого текста,

C_i – i-ый символ шифртекста, k – ключ зашифрования,

$\bar{k} = N - k$ - ключ расшифрования.

Задача 1.1. Осуществить зашифрование и расшифрование символа m_1 на алфавите N при помощи шифра Цезаря на ключе k. Проверить правильность расшифрования.

Пример решения.

Дано:	Решение:
$k = 7$ $m_1 = 24$ $N = 26$	$C_1 = m_1 + k \,(\mathrm{mod}\, N) = 24 + 7 (\mathrm{mod}\, 26) = 5$ $\bar{k} = N - k = 26 - 7 = 19$ $m_1' = C_1 + \bar{k} \,(\mathrm{mod}\, N) = 5 + 19 (\mathrm{mod}\, 26) = 24$
Найти: C_1, $\bar{k}$, m_1' Проверить: $m_1' = m_1$	$m_1' = m_1$

Недостаток операции сложения заключается в ее линейности (см. столбец 2 табл. 1), что позволяет восстановить ключ k уже по одному известному паросочетанию $m_i \leftrightarrow C_i$ для чего достаточно решить одно линейное сравнение:

$$C_i = m_i + k \ (\mathrm{mod}\ N)$$

$$k = C_i - m_i \ (\mathrm{mod}\ N)$$

Несколько лучшие характеристики перемешивания дает операция модульного умножения (см. столбец 3 табл. 1), но здесь первым недостатком является то, что элемент 0 всегда будет зашифрован как 0. Вторым недостатком считается то, что в качестве ключа по умножению можно использовать не все значения алфавита N, а только взаимно простые с N. Например, при попытке использовать $k = 2$ (см. столбец 5 табл. 1) не удается добиться однозначности расшифрования – одному шифробозначению соответствует две шифрвеличины. Например 6 может быть расшифровано как 3 или 7.

Таблица 1

N=8, k_2=2				
m	$k = k_1 = 3$			$k = 2$
	$m+k$(mod N)	mЧk(mod N)	mЧk_1+k_2(mod N)	mЧk(mod N)
0	3	0	2	0
1	4	3	5	2
2	5	6	0	4
3	6	1	3	6
4	7	4	6	0
5	0	7	1	2
6	1	2	4	4
7	2	5	7	6

Первый недостаток исправляется введением дополнительной операции сложения, а второй – ограничением пространства ключей:

$$k_1 : \mathrm{НОД}(k_1, N) = 1$$

Нахождение наибольшего общего делителя возможно при помощи основной теоремы арифметики [11]:

$$N = \prod_{i=1}^{n} p_i^{e_i}$$

Например для : $p_1 = 2, p_2 = 3, e_1 = 4, e_2 = 1$, то есть k_1 не должен делиться на 2 и 3, но ввиду высокой вычислительной сложности метода на практике применяют алгоритм Евклида.

Количество элементов алфавита, пригодных для шифрования при помощи умножения определяется функцией Эйлера:

$$\varphi(N) = \prod_{i=1}^{n} p_i^{e_i - 1}(p_i - 1)$$

Например: $\varphi(48) = 2^{4-1} \cdot (2-1) \cdot 3^{1-1} \cdot (3-1) = 16$

Особый интерес представляют частные случаи функции Эйлера от простого числа и от произведения двух простых чисел (при известных p и q):

$$\varphi(p) = p - 1$$

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

На основании вышеизложенного приведем модель аффинного шифра.

Аффинный шифр

Шифрование осуществляется посимвольно на алфавите N. Используется два ключа k_1 и k_2 и две операции их наложения – соответственно умножение и сложение по модулю N.

Функция зашифрования: $C_i = m_i \cdot k_1 + k_2 (\mathrm{mod}\ N)$

Функция расшифрования: $m_i = (C_i + \overline{k_2})\overline{k_1} (\mathrm{mod}\ N)$

где:

m_i – i-ый символ открытого текста,

C_i – i-ый символ шифртекста,

k_1 – ключ зашифрования по умножению,

k_2 – ключ зашифрования по сложению,

$\overline{k_1} = k_1^{-1} (\mathbf{mod}\ \square)$ - ключ расшифрования по умножению,

$\overline{k_2} = N - k_2$ - ключ расшифрования по сложению.

Отдельно следует рассмотреть вопрос нахождения мультипликативного обратного. На практике для этого применяют обобщенный (расширенный) алгоритм Евклида.

Обобщенный алгоритм Евклида

На вход алгоритма подаются два натуральных числа a и b. На выходе алгоритма получается следующая линейная композиция $s_m \cdot a + t_m \cdot b = r_m$, где при $r_m = 1$:

$s_m = a^{-1} (\mathrm{mod}\ b)$

$t_m = b^{-1} (\mathrm{mod}\ a)$

Ниже приведен обобщенный алгоритм Евклида в редакции Д.Кнута [7], достаточной для его программной реализации:

Алгоритм X. (*Обобщенный алгоритм Евклида*).

X1. [Начальная установка.] Присвоить

$$(u_1, u_2, u_3) \leftarrow (1, 0, a), (v_1, v_2, v_3) \leftarrow (0, 1, b).$$

X2. [$v_3 = 0$?] Если $v_3 = 0$, то выполнение алгоритма заканчивается.

$$(s_m, t_m, r_m) \leftarrow (u_1, u_2, u_3)$$

X3. [Разделить и вычесть.] Присвоить $q = \lfloor u_3 / v_3 \rfloor$, затем присвоить

$$(t_1, t_2, t_3) \leftarrow (u_1, u_2, u_3) - (v_1, v_2, v_3) \cdot q,$$
$$(u_1, u_2, u_3) \leftarrow (v_1, v_2, v_3),$$
$$(v_1, v_2, v_3) \leftarrow (t_1, t_2, t_3).$$

Возвратиться к шагу X2.

При решении задач, содержащихся в данном пособии более удобно пользоваться шаблоном электронной таблицы, построение которого рассмотрено ниже. Диапазон ячеек A1:C1 соответствует начальному значению вектора u. Диапазон ячеек D1:F1 соответствует начальному значению вектора v. В ячейки A2, D2 и G1 вводим формулы согласно рис.3.

	A	B	C	D	E	F	G
1	1	0	113	0	1	7	=ЦЕЛОЕ(C1/F1)
2	=D1			=A1-D1*$G1			

Рис. 3. Исходные данные

Распространяем введенные формулы согласно рис.4.

	A	B	C	D	E	F	G
1	1	0	113	0	1	7	=ЦЕЛОЕ(C1/F1)
2	=D1	=E1	=F1	=A1-D1*$G1	=B1-E1*$G1	=C1-F1*$G1	=ЦЕЛОЕ(C2/F2)

Рис. 4. Распространение формул

При работе с шаблоном значение a вводится в ячейку C1, а значение b в ячейку F1 (см рис.5).

	A	B	C	D	E	F	G
1	1	0	113	0	1	7	16
2	0	1	7	1	-16	1	7

Рис. 5. Общий вид шаблона

Далее выделяем диапазон A2:G2 и распространяем формулы вниз по строкам до появления в столбце G ошибки деления на 0. Значения s_m, t_m и r_m берутся из последней строки столбцов A, B, C соответственно. В случае отрицательной величины s_m или t_m – к ним прибавляются величины b и a соответственно.

	A	B	C	D	E	F	G
1	1	0	113	0	1	7	16
2	0	1	7	1	-16	1	7
3	1	-16	1	-7	113	0	#ДЕЛ/0!

Рис. 6. Работа алгоритма

В данном примере $7^{-1} \bmod 113 = 113 - 16 = 97$. Обязательно следить за тем, чтобы значение r_m из последней строки столбца C было равно единице.

Задача 1.2. Осуществить зашифрование и расшифрование символа m_1 на алфавите N при помощи аффинного шифра на ключевой паре k_1, k_2. Проверить правильность расшифрования.

Пример решения.

Дано:	Решение:
$k_1 = 7$ $k_2 = 36$ $m_1 = 24$ $N = 113$	$C_1 = m_1 \cdot k_1 + k_2 (\mathrm{mod}\ N) =$ $24 \cdot 7 + 36 (\mathrm{mod}\ 113) = 55 + 36 (\mathrm{mod}\ 113) = 91$ $\overline{k_1} = k_1^{-1} (\mathbf{mod}\ N) = \mathbf{7^{-1}} \left(\mathbf{mod\ 113} \right) =$ $= 113 - 16 = 97$ $\overline{k_2} = N - k_2 = 113 - 36 = 77$ $m_1' = \left(C_i + \overline{k_2}\right) \overline{k_1} (\mathrm{mod}\ N) == (91 + 77) \cdot 97 (\mathrm{mod}\ 113) == 55 \cdot 97 (\mathrm{mod}$ $m_1' = m_1$
Найти: C_1, $\overline{k_1}, \overline{k_2}$, m_1’	
Проверить: m_1’ = m_1	

Вскрытие аффинного шифра по двум паросочетаниям

Если вскрытие шифра Цезаря возможно при знании лишь одного паросочетания $m_1 \leftrightarrow C_1$, то вскрытие аффинного шифра возможно по двум паросочетаниям $m_1 \leftrightarrow C_1$ и $m_2 \leftrightarrow C_2$. Для чего необходимо всего лишь решить систему линейных сравнений:

$$\begin{cases} C_1 = m_1 \cdot k_1 + k_2 (\mathrm{mod}\ N) \\ C_2 = m_2 \cdot k_1 + k_2 (\mathrm{mod}\ N) \end{cases}$$

Для этого вычтем из первого сравнения второе, при этом сократится ключ по сложению k_2:

$$(C_1 - C_2) = (m_1 - m_2) \cdot k_1 (\mathrm{mod}\ N)$$

После чего выразим и найдем ключ по умножению k_1:

$$k_1 = (m_1 - m_2) \cdot (C_1 - C_2)^{-1} (\mathrm{mod}\ N)$$

Ключ по сложению восстанавливается из любого исходного сравнения, а другое может служить проверочным выражением:

$$k_2 = C_1 - m_1 \cdot k_1 (\mathrm{mod}\ N)$$

Задача 1.3. Осуществить восстановление ключевой пары k_1, k_2 аффинного шифра на алфавите N по двум известным паросочетаниям $m_1 \leftrightarrow C_1$ и $m_2 \leftrightarrow C_2$. Проверить правильность восстановления ключа.

Пример решения.

Дано:	Решение:
$C_1 = 46$	$\begin{cases} C_1 = m_1 \cdot k_1 + k_2 (\mathrm{mod}\ N) \\ C_2 = m_2 \cdot k_1 + k_2 (\mathrm{mod}\ N) \end{cases}$
$C_2 = 37$	
$m_1 = 19$	$\begin{cases} 46 = 19 \cdot k_1 + k_2 (\mathrm{mod}\ 113) \\ 37 = 104 \cdot k_1 + k_2 (\mathrm{mod}\ 113) \end{cases}$
$m_2 = 104$	
$N = 113$	$(C_1 - C_2) = (m_1 - m_2) \cdot k_1 (\mathrm{mod}\ N)$
Найти: k_1, k_2	$(46 - 37) = (19 - 104) \cdot k_1 (\mathrm{mod}\ 113)$
Проверить:	$9 = 28 \cdot k_1 (\mathrm{mod}\ 113)$
по C_1 или C_2	$k_1 = 9 \cdot 28^{-1} (\mathrm{mod}\ 113) = 9 \cdot 109 (\mathrm{mod}\ 113) = 77$

$k_2 = C_1 - m_1 \cdot k_1 (\mathrm{mod}\ N) = 46 - 19 \cdot 77 (\mathrm{mod}\ 113) == 46 - 107 (\mathrm{mod}\ 113) = 52$

Проверка:

$$C_2 = m_2 \cdot k_1 + k_2 (\mathrm{mod}\ N) = 104 \cdot 77 + 52 (\mathrm{mod}\ 113) =$$
$$= 98 + 52 (\mathrm{mod}\ 113) = 37 .$$

Варианты заданий к первой части

Вариант	Задача 1.1			Задача 1.2				Задача 1.3				
	N	k	m_1	N	k_1	k_2	m_1	N	m_1	C_1	m_2	C_2
1	113	68	34	101	64	45	25	211	13	56	89	23
2	127	69	35	103	65	44	26	199	14	57	88	22
3	131	70	36	107	66	43	27	197	15	58	87	21
4	137	71	37	109	67	42	28	193	16	59	86	20
5	139	72	38	113	68	41	29	191	17	60	85	19
6	149	73	39	127	69	40	30	181	18	61	84	18
7	151	74	40	131	70	39	31	179	19	62	83	17
8	157	75	41	137	71	38	32	173	20	63	82	16
9	163	76	42	139	72	37	33	167	21	64	81	15
10	167	77	43	149	73	36	34	163	22	65	80	14
11	173	78	44	151	74	35	35	157	23	66	79	13
12	179	79	45	157	75	34	36	151	24	67	78	12
13	181	44	46	163	76	33	37	149	25	68	77	11
14	191	43	47	167	77	32	38	139	26	69	76	10
15	193	42	48	173	78	31	39	137	27	70	75	9
16	197	41	49	179	79	30	40	131	28	71	74	8
17	199	40	50	181	80	29	41	127	29	72	73	7
18	211	39	51	191	81	28	42	113	30	73	72	6
19	101	38	52	193	82	27	43	109	31	74	71	5
20	103	37	53	197	83	26	44	107	32	75	70	4
21	107	36	54	199	84	25	45	103	33	76	69	3
22	109	35	55	211	85	24	46	101	34	77	68	2

ЧАСТЬ 2. БАЗОВЫЕ ТЕОРЕТИКО-ЧИСЛОВЫЕ АЛГОРИТМЫ

Китайская теорема об остатках

Исторически формулируется следующим образом [11]: Любое число от 0 до 9 может быть однозначно представлено в виде пары остатков от деления его на 2 и 5. Историческую формулировку теоремы хорошо характеризует приведенная ниже таблица.

Таблица 2.

x	x mod 2	x mod 5
0	0	0
1	1	1
2	0	2
3	1	3
4	0	4
5	1	0
6	0	1
7	1	2
8	0	3
9	1	4

Соответственно все действия над парами остатков однозначно ассоциированы с исходными числами. Например 7 – 3 = 4, или же в остатках (1, 2) – (1, 3) = (0, -1 mod 5) = (0, 4). Данное свойство позволяет организовывать в криптографических приложениях вычисления по типу «разделяй и властвуй». Восстановление же исходных значений по остаткам от деления на два простых числа p и q (частный случай) реализуется при помощи современной формулировки теоремы [10]:

Система сравнений

$$\begin{cases} x = a (\mathrm{mod}\ p) \\ x = b (\mathrm{mod}\ q) \end{cases}$$

При НОД(p, q) = 1 имеет единственное решение по модулю $p \cdot q$, которое определяется по формулам:

$$T = p^{-1} (\mathrm{mod}\ q)$$

$$u = (b - a) T (\mathrm{mod}\ q)$$

$$x = a + u \cdot p$$

Задача 2.1. Восстановить значение x при помощи китайской теоремы об остатках по двум его остаткам от деления a и b, на простые числа p и q соответственно.

Пример решения.

Дано:

$p = 113$

$q = 127$

$a = 65$

$b = 104$

Найти: T, u, x

Решение:

$$\begin{cases} x = a(\mathrm{mod}\ p) \\ x = b(\mathrm{mod}\ q) \end{cases}$$

$$\begin{cases} x = 65(\mathrm{mod}\ 113) \\ x = 104\ (\mathrm{mod}\ 127) \end{cases}$$

$$T = p^{-1}(\mathrm{mod}\ q) = 113^{-1}(\mathrm{mod}\ 127) = 9$$

$$u = (b - a)T(\mathrm{mod}\ q) =$$

$$= (104 - 65) \cdot 9(\mathrm{mod}\ 127) = 39 \cdot 9(\mathrm{mod}\ 127) =$$

$$x = a + u \cdot p = 65 + 97 \cdot 113 = 11026$$

Возведение в квадрат

Рассмотрим приведенную ниже таблицу модульного возведения в квадрат ненулевых элементов простого конечного поля F_7.

Таблица 3

$P = 7$	
x	$x^2 \bmod P$
1	1
2	4
3	2
4	2
5	4
6	1

Как видно из таблицы, если возвести в квадрат можно любой элемент поля, то квадратный корень может быль извлечен только ровно из половины значений. Такие значения называют квадратичными вычетами. Если квадратный корень из значения извлечь нельзя, то такой элемент поля называют квадратичным

невычетом. Также видно, что корни попарно в сумме образуют модуль: $x_1 + x_2 = P$. Возможность извлечения квадратного корня по модулю простого и составного числа оценивают при помощи вычисления соответственно символов Лежандра и Якоби.

Символы Лежандра и Якоби, извлечение квадратного корня

В таблице 4 приведены значения символов Лежандра и Якоби, а также выводы, из них следующие.

Таблица 4

Значение	$L(a, p)$	$J(a, n)$
1	Корень можно извлечь	Корень **может быть** можно извлечь
0	a кратно p	a кратно n
-1 (оно же p - 1 или n – 1)	Корень извлечь нельзя	Корень извлечь нельзя

Как видно из таблицы символ Якоби однозначно дает только отрицательный ответ.

В общем случае символ Лежандра легко вычисляется по формуле: $$L(a,p) = a^{\frac{p-1}{2}} (\mathrm{mod}\ p)$$

Но в реальных приложениях данный способ не используется ввиду своей высокой вычислительной сложности. На практике символы Лежандра и Якоби вычисляют по одинаковой рекуррентной зависимости приведенной ниже:

$$J(a,n) = J(2,n)^e \cdot J(n(\mathrm{mod}\ a_1), a_1) \cdot (-1)^{\frac{(a_1-1)(n-1)}{4}}$$

$$a = 2^e \cdot a_1$$

где: ,

e – максимальная степень двойки на которую можно сократить a,

a_1 – нечетный результат сокращения.

Условия выхода из рекурсии:

$$J(1,n) = 1$$

$$J(2,n) = (-1)^{\frac{n^2-1}{8}}$$

$$J(-1,n) = (-1)^{\frac{n-1}{2}}$$

$$J(0,a_1) = 1$$

Следует отметить, что присутствующие в выражениях операции деления на 2, 4 и 8 легко реализуются программно путем правого битового сдвига на 1, 2 и 3 соответственно.

Извлечение квадратного корня по модулю простого числа $p \equiv 3 (\mathrm{mod}\ 4)$ легко выполняестя по следующей формуле:

$$\sqrt[2]{x} (\mathrm{mod}\ p) = \pm x^{\frac{p+1}{4}} (\mathrm{mod}\ p)$$

В случае же $p \equiv 1 (\mathrm{mod}\ 4)$, вычисление квадратного корня возможно только при помощи алгоритма Шенкса, чем на практике не пользуются ввиду его высокой вычислительной сложности.

Задача 2.2. Определить путем вычисления символа Якоби возможность извлечения квадратного корня из числа *a* по модулю простого числа *p*. Проверить правильность полученного вывода при помощи вычисления символа Лежандра по упрощенной методике. В случае если такая возможность есть – извлечь квадратный корень.

Пример решения.

Дано:	Решение:
$a = 26$ $p = 127$	$J(26,127) = J(2,127)^1 \cdot J(127 (\mathrm{mod}\ 13), 13) \cdot (-1)^{\frac{(13-1)(127-1)}{4}} =$ $= J(2,127) \cdot J(10,13) \cdot 1$
Найти: $J(a, p)$	
Проверить: по $L(a, p)$ при $J(a, p) = 1$ найти $x_{1,2} = \sqrt[2]{a} (\mathrm{mod}\ p)$	$J(2,127) = (-1)^{\frac{127^2-1}{8}} = (-1)^{2016} = 1$
	$J(10,13) = J(2,13)^1 \cdot J(13 (\mathrm{mod}\ 5), 5) \cdot (-1)^{\frac{(5-1)(13-1)}{4}} =$ $= J(2,13) \cdot J(3,5) \cdot 1$

$$J(2,13) = (-1)^{\frac{13^2-1}{8}} = (-1)^{21} = -1$$

$$J(3,5) = J(2,5)^0 \cdot J(2,3) \cdot (-1)^{\frac{(3-1)(5-1)}{4}} = 1 \cdot J(2,3) \cdot 1$$

$$J(2,3) = (-1)^{\frac{(3^2-1)}{8}} = (-1)^1 = -1;$$

$$J(3,5) = 1 \cdot (-1) \cdot 1 = -1$$

$$J(10,13) = (-1) \cdot (-1) \cdot 1 = 1; \quad J(26,127) = 1 \cdot 1 \cdot 1 = 1$$

Проверка:

$$L(a,p) = a^{\frac{p-1}{2}} (\mathrm{mod}\ p) = L(26,127) = 26^{\frac{127-1}{2}} (\mathrm{mod}\ 127) = = 26^{63} (\mathrm{mod}\ 127) = 1$$

Извлечение корней:

$$x_{1,2} = \sqrt[3]{a} (\mathrm{mod}\ p) = \pm a^{\frac{p+1}{4}} (\mathrm{mod}\ p)$$

$$x_1 = a^{\frac{p+1}{4}} (\mathrm{mod}\ p) = 26^{\frac{127+1}{4}} (\mathrm{mod}\ 127) =$$

$$= 26^{32} (\mathrm{mod}\ 127) = 36$$

$$x_2 = p - x_1 = 127 - 36 = 91$$

Возведение в степень и нахождение порождающего элемента группы

Приведенный ниже шаблон реализует так называемое «наивное» модульное возведение в степень в конечном поле. Для примера взято поле F_{19}. Заносим его элементы в первую строку таблицы, а в ячейку A2 вводим формулу согласно рис. 7.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
2	=ОСТАТ(A1*A$1;19)																	

Рис. 7. Исходная формула

Распространяем формулу на диапазон ячеек A2:R18

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
2	1	4	9	16	6	17	11	7	5	5	7	11	17	6	16	9	4	1
3	1	8	8	7	11	7	1	18	7	12	1	18	12	8	12	11	11	18
4	1	16	5	9	17	4	7	11	6	6	11	7	4	17	9	5	16	1
5	1	13	15	17	9	5	11	12	16	3	7	8	14	10	2	4	6	18
6	1	7	7	11	7	11	1	1	11	11	1	1	11	7	11	7	7	1
7	1	14	2	6	16	9	7	8	4	15	11	12	10	3	13	17	5	18
8	1	9	6	5	4	16	11	7	17	17	7	11	16	4	5	6	9	1
9	1	18	18	1	1	1	1	18	1	18	1	18	18	18	18	1	1	18
10	1	17	16	4	5	6	7	11	9	9	11	7	6	5	4	16	17	1
11	1	15	10	16	6	17	11	12	5	14	7	8	2	13	3	9	4	18
12	1	11	11	7	11	7	1	1	7	7	1	1	7	11	7	11	11	1
13	1	3	14	9	17	4	7	8	6	13	11	12	15	2	10	5	16	18
14	1	6	4	17	9	5	11	7	16	16	7	11	5	9	17	4	6	1
15	1	12	12	11	7	11	1	18	11	8	1	18	8	12	8	7	7	18
16	1	5	17	6	16	9	7	11	4	4	11	7	9	16	6	17	5	1
17	1	10	13	5	4	16	11	12	17	2	7	8	3	15	14	6	9	18
18	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Рис. 8. Таблица Кэли с выделенными примитивными элементами

В результате получилась таблица Кэли для возведения в степень по модулю 19. Особое внимание обратим на столбцы, выделенные серым цветом. В них многократное применение операции умножения к заглавным элементам столбцов образует полную перестановку элементов поля. Такие элементы называют порождающими или примитивными. Кроме того, следует обратить внимание на последнюю строку таблицы (все единицы). Она хорошо иллюстрирует малую теорему Ферма:

$$x^{p-1} \equiv 1 (\mathrm{mod}\ p)$$

На практике наивное возведение в степень оказывается неэффективным. Приведем алгоритм бинарного модульного возведения в степень. В основе алгоритма лежит схема Горнера. Проиллюстрируем ее на конкретном примере [8]:

$$x^{27} = \left(x^{13}\right)^2 \cdot x = \left((x^6)^2 \cdot x\right)^2 \cdot x = \left(\left((x^3)^2\right)^2 \cdot x\right)^2 \cdot x == \left(\left((x^2 \cdot x)^2\right)^2 \cdot x\right)^2 \cdot x$$

Что в итоге дает вместо 26 умножений – 4 возведения в квадрат + 3 умножения. Для больших чисел экономия становится еще более значительной. Заметим, что последовательность возведений в квадрат и удвоений соотносится с двоичным представлением показателя степени:

$$27_{10} = 11011_2$$

Начиная со второго старшего двоичного разряда 1 – возвести в квадрат и домножить, 0 – возвести в квадрат.

Алгоритм бинарного модульного экспоненцирования $x^y \pmod N$ [1]:
Представить показатель степени в двоичном виде:

$$y = y_0 \cdot 2^r + \cdots + y_{r-1} \cdot 2^1 + y_r$$

Присвоить $x_0 = x$.

Для i от 1 до r: $x_i = x_{i-1}^2 \cdot x^{y_i} \pmod N$;

$x_r = x^y \pmod N$

Задача 2.3. Возвести число x в степень y по модулю числа N, используя метод бинарного модульного экспоненцирования.

Пример решения.

Дано:
$x = 134$
$y = 344$
$N = 365$

Найти:
$x^y \bmod N$

Решение:

$$y = 344_{10} = 1\overrightarrow{0101100}_2$$

y_i	$x_i = x_{i-1}^2 \cdot x^{y_i} \pmod N$
0	$134^2 \pmod{365} = 71$
1	$71^2 \cdot 134 \pmod{365} = 296 \cdot 134 \pmod{365} = 244$
0	$244^2 \pmod{365} = 41$
1	$41^2 \cdot 134 \pmod{365} = 221 \cdot 134 \pmod{365} = 49$
1	$49^2 \cdot 134 \pmod{365} = 211 \cdot 134 \pmod{365} = 169$
0	$169^2 \pmod{365} = 91$
0	$91^2 \pmod{365} = 251$
0	$251^2 \pmod{365} = 221$

Генерация простых чисел

Детерминированной зависимости распределения простых чисел не существует. Также не существует детерминированного алгоритма их генерации. Наилучшие результаты дают вероятностные тесты числа на простоту из которых на практике наиболее применим тест Рабина-Миллера [13]:

Выбрать для тестирования число p. Вычислить b – число делений $p - 1$ на 2 (то есть 2^b – это наибольшая степень числа 2, на которую делится $p - 1$). Затем вычислить m, такое, что $p = 1 + 2^b \cdot m$.

1) Выбрать случайное число a, меньшее p.
2) Установить $j = 0$ и $z = a^m \bmod p$.
3) Если $z = 1$ или если $z = p - 1$, то p проходит тест и может быть простым числом.
4) Если $j > 0$ и $z = 1$, то p не относится к простым числам.

5) Установить $j = j + 1$. Если $j < b$ и $z \neq (p - 1)$, установить $z = z^2 \bmod p$ и вернуться к шагу 4). Если $z = p - 1$, то p проходит тестирование и может быть простым числом.

6) Если $j = b$ и $z \neq (p - 1)$, то p не относится к простым числам.

Считается, что число, прошедшее тест пять раз с вероятностью более 95% является простым.

Задача 2.4. Провести тест простоты Рабина-Миллера для числа p при помощи заданного числа a.

Пример решения.

Дано:	Решение:
$p = 241$ $a = 99$	$p - 1 = 240 = 2^4 \cdot 3 \cdot 5$ $p = 1 + 2^b \cdot m = 1 + 2^4 \cdot 15$
Повести тест простоты	$j = 0;\ z = a^m \bmod p = 99^{15} \bmod 241 = 126$ $j = 1;\ z = z^2 \bmod p = 126^2 \bmod 241 = 211$ $j = 2;\ z = z^2 \bmod p = 211^2 \bmod 241 = 177$ $j = 3;\ z = z^2 \bmod p = 177^2 \bmod 241 = 240 = p - 1; p - \text{простое}$

Варианты заданий ко второй части

Вариант	Задача 2.1				Задача 2.2		Задача 2.3			Зад. 2.4
	p	*q*	*a*	*b*	*a*	*p*	*x*	*y*	*N*	*p*
1	37	53	9	31	56	149	564	423	721	211
2	41	59	10	32	57	139	563	424	722	223
3	43	61	11	33	58	137	562	425	723	227
4	47	67	12	34	59	131	561	426	724	229
5	29	71	13	35	60	127	560	427	725	233
6	31	73	14	36	61	113	559	428	726	239
7	37	79	15	37	62	109	558	429	727	241
8	41	83	16	38	63	107	557	430	728	251
9	43	89	17	39	64	103	556	431	729	257
10	47	97	18	40	65	101	555	432	730	263
11	53	101	19	41	66	211	554	433	731	269
12	59	103	20	42	67	199	553	434	732	271
13	61	107	21	43	68	197	552	435	733	277

14	67	109	22	44	69	193	551	436	734	281
15	71	113	23	45	70	191	550	437	735	283
16	73	127	24	46	71	181	549	438	736	293
17	79	131	25	47	72	179	548	439	737	307
18	61	137	26	48	73	173	547	440	738	311
19	67	139	27	49	74	167	546	441	739	313
20	71	149	28	50	75	163	545	442	740	317
21	73	151	29	51	76	157	544	443	741	331
22	79	157	30	52	77	151	543	444	742	337

ЧАСТЬ 3. АСИММЕТРИЧНЫЕ КРИПТОГРАФИЧЕСКИЕ ПРОТОКОЛЫ И СИСТЕМЫ ШИФРОВАНИЯ С ОТКРЫТЫМ КЛЮЧОМ

Базой асимметричных криптографических протоколов и алгоритмов является использование однонаправленных функций и однонаправленных функций с лазейкой (секретом) [12].

Основные кандидаты на однонаправленные функции:

1) Перемножение двух больших простых чисел. Обратное преобразование называют факторизацией – разложением на множители.
2) Модульное возведение в степень (экспоненцирование) в конечном поле при фиксированном основании. Обратное преобразование называют дискретным логарифмированием.

Основным кандидатом на однонаправленную функцию с лазейкой (секретом) является модульное экспоненцирование с фиксированным показателем в конечном кольце Z_N, где N является произведением двух больших простых чисел.

Рассмотрим вопросы извлечения корня произвольной степени по модулю простого и составного чисел.

В основе операции извлечения корня в конечном поле лежит Малая теорема Ферма:

$$x^{p-1} \equiv 1 (\mathrm{mod}\ p)$$

Для начала приведем доказательство следующего утверждения [9]:

$$x^y (\mathrm{mod}\ p) \equiv x^{y \bmod (p-1)} (\mathrm{mod}\ p)$$

Любое натуральное y может быть представлено в виде: $y = k(p-1) + r$, отсюда

$$x^y (\mathrm{mod}\ p) \equiv x^{(k(p-1)+r) \bmod (p-1)} (\mathrm{mod}\ p) \equiv (x^{p-1})^k \cdot x^r (\mathrm{mod}\ p) \equiv$$

{применяя малую теорему Ферма}

$$\equiv 1^k \cdot x^r (\mathrm{mod}\ p) \equiv x^{y \bmod (p-1)} (\mathrm{mod}\ p)$$

Из вышеуказанного следует, что извлечение корня степени y по модулю p возможно лишь для y: $\mathrm{НОД}(y, p-1) = 1$

Таким образом: $\sqrt[y]{x} (\mathrm{mod}\ p) = x^{y^{-1} \bmod (p-1)} (\mathrm{mod} p)$

В основе операции извлечения корня в конечном поле лежит обобщение Эйлера для малой теоремы Ферма:

$$x^{\varphi(N)} \equiv 1 (\mathrm{mod}\ N)$$

Опустив промежуточные выкладки по аналогии с предыдущим случаем получаем, что извлечение корня степени y по модулю составного числа N возможно лишь для

$$y: \mathrm{НОД}(y, \varphi(N)) = 1$$

и реализуется как:

$$\sqrt[y]{x} (\mathrm{mod}\ N) = x^{y^{-1} \mathrm{mod}(\varphi(N))} (\mathrm{mod} N),$$

при этом обязательно необходимо знать разложение $N = p \cdot q$ – секрет, без которого невозможно эффективное вычисление значения функции Эйлера $\varphi(N)$.

Далее рассмотрим несколько базовых асимметричных протоколов и алгоритмов шифрования.

Протокол Диффи-Хеллмана

Протокол предназначен для совместной выработки секретного ключа с использованием открытого канала связи [2]. В основе протокола лежит трудность задачи дискретного логарифмирования в конечном поле. Общими (и общедоступными) параметрами системы являются простое число P и примитивный элемент конечного поля G.(На практике размер значения P должен быть не менее 1024 бит.)

Каждый из двух абонентов A и B соответственно выбирают секретные ключи x и y.

При этом $x: 1 < x < P - 1$ и $y: 1 < y < P - 1$.

Далее абоненты обмениваются открытыми сообщениями:

A передает B: $G^x \mathrm{mod} P$

B передает A: $G^y \mathrm{mod} P$

Противник, зная G, но, не зная x и y, вынужден решать вычислительно сложную задачу дискретного логарифмирования.

Далее абонент A вычисляет: $k = (G^y)^x \mathrm{mod} P$

Абонент B вычисляет: $k' = (G^x)^y \mathrm{mod} P$

Причем в силу коммутативности операции модульного экспоненцирования: $k = k'$.

Соблюдение требований $x: 1 < x < P - 1$ и $y: 1 < y < P - 1$ является обязательным, так как при выборе секретного ключа равного 1 на следующем шаге будет передано $G^1 = G$, а второй абонент передаст секретный ключ в открытом виде. При выборе секретного ключа равного $P - 1$ на следующем шаге будет передана единица (см.малую теорему Ферма) и секретный ключ $k = k'$ также будет равен единице.

Задача 3.1. Для заданного простого числа P, определить минимальный порождающий элемент G и выбрав самостоятельно секретные ключи x и y провести совместную выработку сеансового ключа k (k'). Проверить совпадение сеансовых ключей обоих абонентов.

Пример решения.

Дано:	Решение:
$P = 113$	$x: 1 < x < P - 1;\ \ x = 26$
Найти: x, y, G_{min}, G^x, G^y, k, k'	$y: 1 < y < P - 1;\ \ y = 45$ $G_{min} = 3$ $G^x \bmod P = 3^{26} \bmod 113 = 36$ $G^y \bmod P = 3^{45} \bmod 113 = 55$
Проверить: $k = k'$	

$k = (G^y)^x \bmod P = (55)^{26} \bmod 113 = 31$

$k' = (G^x)^y \bmod P = (36)^{45} \bmod 113 = 31$

$k = k'$

Трехпроходный протокол Шамира

Данный протокол [9] в отличие от предыдущего, где ключ получается в процессе выполнения, позволяет передать заранее сгенерированный секретный ключ k. В основе протокола лежат свойства ассоциативности и коммутативности операций модульного экспоненцирования и извлечения корней в конечном поле. Общим параметром системы является простое число P. На практике значение P занимает не менее 1024 бит.

Каждый из двух абонентов А и Б соответственно выбирают секретные ключи x_A и $x_Б$.

При этом $x_А: \text{НОД}(x_А, P - 1) = 1$ и $x_Б: \text{НОД}(x_Б, P - 1) = 1$.

Далее каждый из двух абонентов А и Б соответственно вычисляют секретные ключи y_A и $y_Б$:

$y_А = x_А^{-1} \mathrm{mod}(P-1)$

$y_Б = x_Б^{-1} \mathrm{mod}(P-1)$

1 проход. Абонент А генерирует секретный сеансовый ключ k и передает абоненту Б: $k^{x_А} \mathrm{mod} P$

2 проход. Абонент Б передает абоненту А: $(k^{x_А})^{x_Б} \mathrm{mod} P$

3 проход. Абонент А передает абоненту Б: $\left((k^{x_А})^{x_Б}\right)^{y_А} \mathrm{mod} P$, что фактически равно $k^{x_Б} \mathrm{mod} P$

После этого абоненту Б остается восстановить ключ k':

$$k' = \left(\left((k^{x_А})^{x_Б}\right)^{y_А}\right)^{y_Б} \mathrm{mod}\, P = (k^{x_А})^{y_Б} \mathrm{mod} P$$

Соблюдение требований: $x_А : \mathrm{НОД}(x_А, P-1) = 1$ и $x_Б : \mathrm{НОД}(x_Б, P-1) = 1$ является обязательным, так как без этого корректное восстановление k невозможно.

Противник, зная только модуль, по которому производились вычисления, но, не зная, ни основания, ни показателя степени не может решить, ни задачу логарифмирования, ни задачу извлечения корня. Фактически, протокол Шамира уже является асимметричной системой шифрования, но отнести его к алгоритмам шифрования мешает его «трехпроходность».

Задача 3.2. Для заданного простого числа P и сеансового ключа k, определить самостоятельно пары секретных ключей обоих абонентов и осуществить передачу сеансового ключа при помощи трехпроходного протокола Шамира. Проверить совпадение сеансовых ключей обоих абонентов.

Пример решения.

Дано:	Решение:
$P = 113$ $k = 37$	$x_А : \mathrm{НОД}(x_А, P-1) = \mathrm{НОД}(x_А, 112) = 1; x_А = 17$ $y_А = x_А^{-1} \mathrm{mod}(P-1) = 17^{-1} \mathrm{mod}(112) = 33$ $x_Б : \mathrm{НОД}(x_Б, P-1) = \mathrm{НОД}(x_Б, 112) = 1; x_Б = 45$ $y_Б = x_Б^{-1} \mathrm{mod}(P-1) = 37^{-1} \mathrm{mod}(112) = 5$ $k^{x_А} \mathrm{mod} P = 37^{17} \mathrm{mod} 113 = 80$ $(k^{x_А})^{x_Б} \mathrm{mod} P = 80^{45} \mathrm{mod} 113 = 59$ $\left((k^{x_А})^{x_Б}\right)^{y_А} \mathrm{mod} P = 59^{33} \mathrm{mod} 113 = 70$
Найти: $x_А, y_А, x_Б, y_Б, k^{x_А}$,	

$$\left(\left(k^{x_A}\right)^{x_Б}\right)^{y_A} \bmod P = 59^{33} \bmod 113 = 70$$

Проверить:
$k = k'$

$$k' = \left(\left(\left(k^{x_A}\right)^{x_Б}\right)^{y_A}\right)^{y_Б} \bmod P = 70^{55} \bmod 113 = 37 \quad ; \quad k = k'$$

Криптосистема RSA

Первая полноценная асимметричная система шифрования. В основе системы – трудность извлечения корней по модулю составного числа без знания разложения этого числа на два простых множителя. В отличие от предыдущих протоколов система не имеет общих параметров. Наоборот, для обеспечения стойкости необходимо, чтобы каждый абонент имел свой модуль для вычислений.

Абонент, желающий получать сообщения, зашифрованные на открытом ключе, выбирает два простых числа p и q и вычисляет:

$$N = p \cdot q$$

$$\varphi(N) = (p-1)(q-1)$$

Открытая экспонента шифрования выбирается из условия $E: \text{НОД}(E, \varphi(N)) = 1$. Соблюдение данного требования обязательно по условию возможности расшифрования.

Закрытая экспонента находится при помощи обобщенного алгоритма Евклида: $d = E^{-1} \bmod \varphi(N)$

Открытым ключом является пара: $K_o = (E, N)$

Закрытым ключом является набор: $k_з = (d, p, q)$

Зашифрование сообщения m производится как: $C = m^E \bmod N$

Для расшифрования производят извлечение корня: $m' = C^d \bmod N$

Так как для удобства абонентов, выполняющих зашифрование на открытом ключе, предлагается выбирать малые экспоненты E, значение d может иметь значительно больший размер, поэтому удобнее выполнять извлечение корня по методу «разделяй и властвуй» по модулям простых чисел p и q:

$$m_p = C^d \bmod p = (C \bmod p)^{d \bmod (p-1)} (\bmod p)$$

$$m_q = C^d \bmod q = (C \bmod q)^{d \bmod (q-1)} (\bmod q)$$

Значение *m'* затем восстанавливается при помощи китайской теоремы об остатках из системы сравнений:

$$\begin{cases} m' = m_p \left(\bmod p\right) \\ m' = m_q \left(\bmod q\right) \end{cases}$$

Подобным образом и следует выполнять расшифрование при решении нижеследующей задачи, так как калькулятор Windows может дать переполнение регистра.

Главным недостатком канонической RSA является то, что один и тот же открытый текст m всегда зашифровывается одинаковым *C*, что открывает возможности анализа трафика [10,13].

Задача 3.3. Для заданной пары простых чисел *p* и *q* и открытого текста *m*, определить самостоятельно остальные параметры шифрсистемы RSA и осуществить зашифрование и расшифрование. Проверить совпадение исходного *m* и полученного *m'* значений открытого текста.

Пример решения.

Дано:	Решение:
$p = 113$	$N = p \cdot q = 113 \cdot 127 = 14351$
$q = 127$	$\varphi(N) = (p-1)(q-1) = (113-1)(127-1) == 112 \cdot 126 = 14112$
$m = 59$	$E: НОД(E, \varphi(N)) = НОД(E, 14112) = 1; E = 17$
Найти: *N*, $\varphi(N)$, *E*, *d*, *C*, *m'*	$d = E^{-1} \bmod \varphi(N) = 17^{-1} \bmod 14112 = 6641$
	$C = m^E \bmod N = 59^{17} \bmod 14351 = 4473$
Проверить:	$m' = C^d \bmod N = 4473^{6641} \bmod 14355$
m' = *m*	

дает переполнение

$$m_p = C^d \bmod p = (C \bmod p)^{d \bmod (p-1)} (\bmod p) == (4473 \bmod 113)^{6641 \bmod 112} (\bmod 1$$

$$= 66^{38} (\mathrm{mod}\ 113) = 59$$

$$m_q = C^d \mathrm{mod} q = (C \ \mathrm{mod}\ q)^{d \ \mathrm{mod}(q-1)} (\mathrm{mod}\ q) == (4473 \ \mathrm{mod}\ 127)^{6641 \ \mathrm{mod} 126} (\mathrm{mod}\ 1$$
$$= 28^{89} (\mathrm{mod}\ 127) = 59$$

$$\begin{cases} m' = m_p \left(\mathrm{mod}\ p\right) \\ m' = m_q \left(\mathrm{mod}\ q\right) \end{cases}; \quad \begin{cases} m' = 59 \left(\mathrm{mod}\ 113\right) \\ m' = 59 \left(\mathrm{mod}\ 127\right) \end{cases}$$

$m' = m = 59$

Криптосистема Эль-Гамаля

В данной системе исправлен недостаток RSA по однозначному шифрованию. В самой схеме заложено присутствие случайного элемента – эфемерного сеансового ключа *k*, передаваемого в сообщении неявным образом. Кроме того, система позволяет использовать общие для всех абонентов параметры, аналогичные протоколу Диффи-Хеллмана – простое число *P* и примитивный элемент *G*.

Каждый абонент системы генерирует секретный ключ $x: 1 < x < P - 1$

Открытый ключ Y вычисляют как: $Y = G^x \bmod P$

Абонент, желающий зашифровать сообщение *m* выбирает сеансовый ключ $k: 1 < k < P - 1$

Обязательность соблюдения условий $x: 1 < x < P - 1$ и $k: 1 < k < P - 1$ аналогична протоколу Диффи-Хеллмана.

Особенностью (и недостатком) шифрования является то, что сообщение при шифровании удваивается по длине и представляет собой пару: $C = (C_1, C_2)$

$C_1 = G^k \bmod P$ - первая часть сообщения, не зависящая от открытого текста. C_1 служит для неявной передачи эфемерного ключа и, ввиду независимости от шифруемого сообщения, может быть выработано заранее в фоновом режиме.

$C_2 = m \cdot Y^k \bmod P$ - вторая часть сообщения, служащая собственно для передачи *m*.

Расшифрование сообщения *m'* производится как:

$$m' = C_2 \cdot \left(C_1^{\,x}\right)^{-1} \bmod P$$

Ниже приведено краткое доказательство работоспособности, в котором для удобства опущены операции приведения по модулю:

$$\frac{C_2}{C_1^{\,x}} = \frac{m \cdot Y^k}{(G^k)^x} = \frac{m \cdot (G^x)^k}{(G^k)^x} = m$$

Недостатком системы также является необходимость помимо возведения в степень при расшифровании также применить

обобщенный алгоритм Евклида, что отрицательно сказывается на скорости вычислений.

Задача 3.4. Для заданного простого числа P и открытого текста m, определить самостоятельно минимальный порождающий элемент G и остальные параметры шифрсистемы Эль-Гамаля и осуществить зашифрование и расшифрование. Проверить совпадение исходного m и полученного m' значений открытого текста.

Пример решения

Дано:	Решение:
$P = 131$ $m = 59$	$G_{min} = 2$ $x: 1 < x < P - 1;\ \ x = 71$
Найти: G_{min}, x, Y, k, C_1, C_2, m' Проверить: $m' = m$	$Y = G^x \bmod P = 2^{71} \bmod 131 = 67$ $k: 1 < k < P - 1;\ \ k = 92$ $C_1 = G^k \bmod P = 2^{92} \bmod 131 = 25$

$$C_2 = m \cdot Y^k \bmod P = m \cdot Y^k \cdot (\bmod P) = 59 \cdot [67]^{92} \pmod{131} =$$

$$m' = C_2 \cdot \left(C_1^{\,x}\right)^{-1} \bmod P = 91 \cdot (25^{71})^{-1} \bmod 131 =$$

$$= 91 \cdot 117^{-1} \bmod 131 = 91 \cdot 28 \bmod 131 = 59$$

$$m' = m = 59$$

Криптосистема Рабина

Данная система не получила широкого применения, но приведена здесь для иллюстрации возможности использовать при асимметричном шифровании операции модульного возведения в квадрат. Система основана на трудности извлечения квадратного корня по модулю составного числа без знания разложения модуля на множители.

Каждый абонент выбирает собственную пару простых чисел p и q, с соблюдением следующего требования: $p \equiv q \equiv 3 (\bmod 4)$, что позволит извлекать квадратный корень не прибегая к алгоритму Шенкса.

Затем вычисляют: $N = p \cdot q$ и выбирают $B: 0 \le B < N - 1$

Открытым ключом является пара $K_0 = (B, N)$

Закрытым ключом является пара $k_з = (p, q)$

Особенностью системы является очень простая и быстрая операция зашифрования :

$$C = m(m + B) (\bmod N)$$

Для расшифрования необходимо вычислить:

$$m' = \sqrt{B^2 \cdot 4^{-1} + C} - B \cdot 2^{-1} (\mathrm{mod}\ N)$$

Ниже приведено краткое доказательство работоспособности, в котором для удобства опущены операции приведения по модулю:

$$\sqrt{\frac{B^2}{4} + C} - \frac{B}{2} = \sqrt{\frac{B^2}{4} + \frac{4m\,(m+B)}{4}} - \frac{B}{2} = \sqrt{\frac{B^2 + 4m\,(m+B)}{4}} - \frac{B}{2} =$$

Для удобства вычислений принимаем: $s' = \sqrt{B^2 \cdot 4^{-1} + C}\ (\mathrm{mod}\ N)$
и соответственно: $t' = B^2 \cdot 4^{-1} + C\ (\mathrm{mod}\ N)$
Извлечение корней производится по модулю простых чисел p и q:

$$s_{p1-2} = \sqrt{t'}\ \mathrm{mod} p$$

$$s_{q1-2} = \sqrt{t'}\ \mathrm{mod} q$$

Каждое из сравнений дает по два корня и, соответственно вариантов значения s_i получается всего четыре (что является основным недостатком системы). Найдем их попарно, при помощи КТО и свойства корней:

$$\begin{cases} s_1 = s_{p_1} (\mathrm{mod}\ p) \\ s_1 = s_{q_1} (\mathrm{mod}\ q) \end{cases}$$

$$s_2 = N - s_1$$

$$\begin{cases} s_3 = s_{p_1} (\mathrm{mod}\ p) \\ s_3 = s_{q_2} (\mathrm{mod}\ q) \end{cases}$$

$$s_4 = N - s_3$$

Четыре варианта расшифрования получим, выполнив вычисления:

$$m'_i = s_i - B \cdot 2^{-1} \left(\mathrm{mod} N\right)$$

Для распознавания правильного варианта расшифрования, как правило вводят некоторую обратимую функцию избыточности $F(m)$, позволяющую с высокой вероятностью выбрать правильный вариант. В приведенном ниже примере $F(m) = 44||m$.

Задача 3.5. Для заданной пары простых чисел p и q и открытого текста m, определить самостоятельно остальные параметры шифрсистемы Рабина и осуществить зашифрование и

расшифрование. Проверить совпадение исходного m и одного из четырех полученных m_j’ значений открытого текста.

Пример решения

Дано:	Решение:
$p = 127$ $q = 131$ $m = 4410 = 44\|\|10$	$12_7 \equiv 13_1 \equiv 3 (\mathrm{mod}\ 4)$ $N = p \cdot q = 127 \cdot 131 = 16637$ $B: 0 \le B < N - 1; B = 12345$ $C = m(m + B) (\mathrm{mod}\ N) =$ $= 4410 \cdot (4410 + 12345) (\mathrm{mod}\ 16637) = = 4410 \cdot 118 = 463$
Найти: $N, B, C, m_{1\text{-}4}$’ Проверить: $\exists\ m_i$’ $= m$	

$$t' = B^2 \cdot 4^{-1} + C \pmod N = 12345^2 \cdot 12478 + 4633 \pmod{16637} =$$

$$= 41055 \cdot 12478 + 4633 \pmod{16637} =$$

$$= 13504 + 4633 \pmod{16637} = 1500$$

$$s_{p1-2} = \sqrt{t'}\ \mathrm{mod} p; s_{p_1} = (1500\ \mathrm{mod}\ 127)^{\frac{127+1}{4}}\ \mathrm{mod} 127 = = 103^{32} \mathrm{mod} 127 = 22$$

$$s_{p_2} = p - s_{p_1} = 127 - 22 = 105$$

$$s_{q1-2} = \sqrt{t'}\ \mathrm{mod} q = \left(1500 \bmod 131\right)^{\frac{131+1}{4}}\ \mathrm{mod} 131 = = 59^{33} \mathrm{mod} 131 = 94$$

$$s_{q_2} = q - s_{q_1} = 131 - 94 = 37$$

$$\begin{cases} s_1 = s_{p_1} (\mathrm{mod}\ p) \\ s_1 = s_{q_1} (\mathrm{mod}\ q) \end{cases} \qquad \begin{cases} s_2 = s_{p_1} (\mathrm{mod}\ p) \\ s_2 = s_{q_2} (\mathrm{mod}\ q) \end{cases}$$

$$\begin{cases} s_1 = 22 (\mathrm{mod}\ 127) \\ s_1 = 94 (\mathrm{mod}\ 131) \end{cases} \qquad \begin{cases} s_2 = 22 (\mathrm{mod}\ 127) \\ s_2 = 37 (\mathrm{mod}\ 131) \end{cases}$$

$$T = 12_7^{-1} (\mathrm{mod}\ 131) = 98 \qquad T = 98$$

$$u = (94 - 22) 98 (\mathrm{mod}\ 131) = \qquad u = (37 - 22) 98 (\mathrm{mod}\ 131) =$$

$$= 72 \cdot 98 (\mathrm{mod}\ 131) = 113 \qquad = 15 \cdot 98 (\mathrm{mod}\ 131) = 29$$

$s_1 = 22 + 113 \cdot 127 = 14373$ $s_2 = 22 + 29 \cdot 127 = 3705$

$s_2 = N - s_1 = 16637 - 14373 = 2264$ $s_3 = 16637 - 3705 = 12932$

$B \cdot 2^{-1} (\mathrm{mod}\ N) = 12345 \cdot 8319 (\mathrm{mod}\ 16637) = 14491$

$m_i' = s_i - B \cdot 2^{-1} \left(\mathrm{mod} N\right)$

$m_1' = 14373 - 14491 (\mathrm{mod}\ 16637) = 16519$

$m_2' = 2264 - 14491 (\mathrm{mod}\ 16637) = 4410$

$m_3' = 3705 - 14491 (\mathrm{mod}\ 16637) = 5851$

$m_4' = 12932 - 14491 (\mathrm{mod}\ 16637) = 15078$

$m_2' = m$

Варианты заданий к третьей части

Вар-т	Зад. 3.1	Зад. 3.2		Задача 3.3			Зад. 3.4		Задача 3.5		
	P	P	k	p	q	m	P	m	p	q	m
1	211	271	10	101	191	21	283	15	19	127	13
2	223	277	11	103	193	22	293	16	23	131	14
3	227	281	12	107	197	23	307	17	31	139	15
4	229	283	13	109	199	24	311	18	43	151	16
5	233	293	14	113	211	25	313	19	47	163	17
6	239	307	15	127	223	26	317	20	59	167	18
7	241	311	16	131	227	27	331	21	67	179	19
8	251	313	17	137	229	28	337	22	71	191	20
9	257	317	18	139	233	29	347	23	79	199	21
10	263	331	19	149	239	30	349	24	83	211	22
11	269	337	20	151	241	31	353	25	103	223	23
12	271	211	21	157	251	10	359	26	107	227	24
13	277	223	22	163	257	11	367	27	127	239	25
14	281	227	23	167	263	12	373	28	131	251	26
15	283	229	24	173	269	13	379	29	139	263	27
16	293	233	25	179	271	14	383	30	151	271	28
17	307	239	26	181	277	15	389	31	163	283	29
18	311	241	27	191	281	16	397	10	167	307	30
19	313	251	28	193	283	17	401	11	179	31	31
20	317	257	29	197	293	18	409	12	191	43	10
21	331	263	30	199	307	19	419	13	199	47	11
22	337	269	31	211	181	20	421	14	211	59	12

ЧАСТЬ 4. АСИММЕТРИЧНЫЕ СХЕМЫ ЭЛЕКТРОННО-ЦИФРОВОЙ ПОДПИСИ

Электронно-цифровая подпись является фактически обращением асимметричной схемы шифрования. «Шифрование» осуществляется на закрытом ключе, расшифрование на открытом. Данная схема подтверждает, что зашифрование произведено владельцем секретного ключа. В практически реализованных схемах электронно-цифровой подписи подписывают не все сообщение (слишком велики вычислительные затраты), а только его свертка – хэш. В данном пособии вопросы хэширования не затрагиваются, поэтому все задачи приведены для случая аутентификации. В процессе аутентификации сервер передает клиенту случайную последовательность M, клиент подписывает ее на закрытом ключе, а сервер проверяет подпись клиента на открытом ключе и предоставляет ему соответствующий доступ. Ниже рассмотрим основные базовые схемы электронно-цифровой подписи.

Цифровая подпись RSA

Реализует так называемую схему с восстановлением сообщения и является фактически той же самой системой асимметричного шифрования с инвертированным порядком применения шифрующих экспонент. Для обеспечения стойкости, как и при шифровании необходимо, чтобы каждый абонент имел свой модуль для вычислений. Кроме того, если RSA используется и для шифрования и для подписывания сообщений, для решения каждой из этих задач обязательно использования разных наборов ключей.

Абонент, желающий подписывать сообщения на секретном ключе ключе, выбирает два простых числа p и q и вычисляет:

$$N = p \cdot q$$

$$\varphi(N) = (p-1)(q-1)$$

Открытая экспонента, как и при шиффровании выбирается из условия $E: \text{НОД}(E, \varphi(N)) = 1$. Соблюдение данного требования обязательно по условию возможности проверки подписи.

Закрытая экспонента находится при помощи обобщенного алгоритма Евклида: $d = E^{-1} \bmod \varphi(N)$

Открытым ключом является пара: $K_{\text{о}} = (E, N)$

Закрытым ключом является набор: $k_{\text{з}} = (d, p, q)$

Подписание сообщения *M* производится как: $S = M^d \bmod N$

Так как для удобства абонентов, выполняющих проверку подписи на открытом ключе, предлагается выбирать малые экспоненты *E*, значение *d* может иметь значительно больший размер, поэтому удобнее выполнять возведение в степень по методу «разделяй и властвуй» по модулям простых чисел *p* и *q*:

$$S_p = M^d \bmod p = (M \bmod p)^{d \bmod (p-1)} (\bmod p)$$

$$S_q = M^d \bmod q = (M \bmod q)^{d \bmod (q-1)} (\bmod q)$$

Значение *S* затем восстанавливается при помощи китайской теоремы об остатках из системы сравнений:

$$\begin{cases} S = S_p (\bmod p) \\ S = S_q (\bmod q) \end{cases}$$

Проверка подписи осуществляется как: $M' = S^E \bmod N$

Подпись считается правильной, при корректном восстановлении исходного сообщения $M = M'$

Задача 4.1. Для заданной пары простых чисел *p* и *q* и открытого текста *M*, определить самостоятельно остальные параметры схемы электронно-цифровой подписи RSA и осуществить подписание сообщения. Проверить правильность цифровой подписи. (При возможном переполнении регистра калькулятора – воспользоваться КТО аналогично задаче 3.3.)

Пример решения

Дано:	Решение:
$p = 113$	$N = p \cdot q = 113 \cdot 131 = 14803$
$q = 131$	$\varphi(N) = (p-1)(q-1) = (113-1)(131-1) = 112 \cdot 130 = 14560$
$M = 765$	$E: \text{НОД}(E, \varphi(N)) = \text{НОД}(E, 14560) = 1; E = 19$
Найти: *N*,	$d = E^{-1} \bmod \varphi(N) = 19^{-1} \bmod 14560 = 2299$
$\varphi(N)$, *E*, *d*,	$S = M^d \bmod N = 765^{2299} \bmod 14803 = 11917$

S, M' Проверить: *M' = M*	$M' = S^E \bmod N = 1191^{19} \bmod 14803 = 765$

$$M' = M$$

Цифровая подпись Эль-Гамаля

Данная система электронно-цифровой подписи реализует схему с дополнением, поэтому существуют отличия от шифрования Эль-Гамаля, основное из которых – дополнительные требования, предъявляемые к эфемерному сеансовому ключу *k*. Как и при шифровании Эль-Гамаля, система позволяет использовать общие для всех абонентов параметры, аналогичные протоколу Диффи-Хеллмана – простое число *P* и примитивный элемент *G*.

Каждый абонент системы генерирует секретный ключ

$$x: 1 < x < P - 1$$

Открытый ключ Y вычисляют как: $Y = G^x \bmod P$

Абонент, желающий подписать сообщение *M*, выбирает сеансовый ключ $k: 1 < k < P - 1 : \text{НОД}(k, P - 1) = 1$, последнее требование необходимо к выполнению для успешной проверки подписи.

Особенностью подписания является то, что сообщение при подписании удваивается по длине и представляет собой пару: (R, S)

$R = G^k \bmod P$ - первая часть подписи, не зависящая от подписываемого сообщения. *R* служит для неявной передачи эфемерного ключа и, ввиду независимости от подписываемого сообщения, может быть выработано заранее в фоновом режиме.

$S = (M - x \cdot R)k^{-1} (\bmod P - 1)$ - вторая часть подписи, связанная с сообщением *M*. Самой затратной операцией при формировании второй части является вычисление мультипликативного обратного при помощи обобщенного алгоритма Евклида, но оно может выполняться заранее вместе с генерацией *k*. Непосредственно же подписание сводится к выполнению двух операций сложения и одной вычитания.

Для проверки подписи, проверяют сравнение:

$$Y^R \cdot R^S \equiv G^M (\bmod P).$$

если оно выполняется, подпись считается правильной. Для проверки приходится выполнять три вычислительно затратных операции модульного экспоненцирования, что является несомненным недостатком схемы. Данный недостаток несколько компенсируется тем, что сервер обладает как правило большей вычислительной мощностью по сравнению с клиентом.

Ниже приведено краткое доказательство работоспособности, в котором для удобства опущены операции приведения по модулю:

$$Y^R \cdot R^S = (G^x)^{Gk} \cdot (G^k)^{(M-x\cdot R)k^{-1}} = (G^x)^{Gk} \cdot G^{(M-x\cdot Gk)} =$$

$$= G \cdot G = G$$

Ввиду своих недостатков данная схема в чистом виде на практике применяется редко, но она послужила родоначальником целого семейства схем электронно-цифровой подписи на ее основе.

Задача 4.2. Для заданного простого числа P и открытого текста M, определить самостоятельно минимальный порождающий элемент G и остальные параметры схемы электронно-цифровой подписи Эль-Гамаля и осуществить подписание сообщения. Проверить правильность цифровой подписи.

Пример решения.

Дано:	Решение:
$P = 131$	$G_{min} = 2$
$M = 92$	$x: 1 < x < P-1;\ x = 66$
Найти: G_{min}, x, Y, k, R, S	$Y = G^x \bmod P = 2^{66} \bmod 131 = 129$
Проверить подпись	$k: 1 < k < P-1:$
	$НОД(k, P-1) = НОД(k, 130) = 1;\ k = 43$
	$k^{-1} \bmod (P-1) = 43^{-1} \bmod 130 = 127$

$R = G^k \bmod P = 2^{43} \bmod 131 = 17$

$S = (M - x \cdot R)k^{-1} (\bmod P - 1) =$

$= (92 - 66 \cdot 1_7) \cdot 12_7 (\mathrm{mod}\ 13_0$

$= (92 - 8_2) \cdot 12_7 (\mathrm{mod}\ 13_0) = 10 \cdot 12_7 (\mathrm{mod}\ 13_0) =$

Проверка:

$$Y^R \cdot R^S (\mathrm{mod}\ P) = 12_9^{1_7} \cdot 1_7^{10_0} (\mathrm{mod}\ 13_1) =$$
$$= 59 \cdot 10_7 (\mathrm{mod}\ 13_1) =$$

$$G^M (\mathrm{mod}\ P) = 2^{9_2} (\mathrm{mod}\ 13_1) = 2_5$$

Генерация сильно простого числа и порождающего элемента

На практике простые числа, используемые в качестве параметров системы должны обладать следующим свойством:

$$P = Q \cdot D + 1$$

где Q – достаточно большое простое число (более 160 бит в двоичном представлении), а D – случайное (произвольное последовательным перебором) натуральное четное число, длина двоичного представления которого обеспечивает необходимую длину числа P. Такие числа P называют сильно простыми. Для вычислений по модулю таких чисел трудно организовать факторизацию и дискретное логарифмирование.

Кроме того на практике порождающий (примитивный элемент конечного поля) легко вычисляется при помощи подбора случайного (произвольного) натурального числа H, как [10]:

$$G = H^{\frac{P-1}{Q}} (\mathrm{mod} P) = H^{D} (\mathrm{mod} P) \neq 1$$

Проверку правильности подбора осуществляют вычислением:

$$G^{Q} \equiv 1 \ (\mathrm{mod} P)$$

Случай, когда G^Q отлично от 1, означает ослабление алгоритма в G^Q раз.

Задача 4.2'. Решить задачу 4.2 с вычислением примитивного элемента G, вместо назначения G_{min} при помощи шаблона электронной таблицы.

Комментарий к задаче.

Для вычисления G необходимо разложить число $(P - 1)$ на простые множители. За Q принять наибольший простой делитель $(P - 1)$. Далее подбирается произвольное $\tilde{H}$ до тех пор, пока не будет выполнено условие:

$$G = H^{\frac{P-1}{Q}} (\mathrm{mod} P) = H^{D} (\mathrm{mod} P) \neq 1$$

Кроме того, возможен также произвольный подбор G с последующей проверкой условия

$$G^{Q} \equiv 1 \ (\mathrm{mod} P).$$

Задача 4.3. По заданному простому числу Q осуществить генерацию сильно простого числа $P \leq 1000$, пользуясь таблицей простых чисел (см. Приложение 1). Затем сгенерировать порождающий элемент G.

Пример решения.

Дано:	Решение:
Q = 101	$P = Q \cdot \tilde{D} + 1$
Найти: P, G Проверить: по $G^Q = 1$	$\tilde{D} = 2;\ P = 101 \cdot 2 + 1 = 203 - \text{составное}$ $\tilde{D} = 4;\ P = 101 \cdot 4 + 1 = 405 - \text{составное}$

$\tilde{D} = 6; P = 101 \cdot 6 + 1 = 607 - \text{простое}; D = 6$

$G = H^{D} (\mathrm{mod} P)$

$\tilde{H} = 67; G = 67^{6} (\mathrm{mod}\ 607) = 182 \neq 1$

Проверка:

$G^{Q} (\mathrm{mod}\ P) = 182^{101} \bmod 607 = 1$

Цифровая подпись DSA

Данная система является дальнейшим развитием схемы Эль-Гамаля. Главное достоинство системы в том, что использование в качестве модуля сильно простого числа позволяет ряд вычислений производить по модулю порядка циклической мультипликативной подгруппы *Q*. Данное обстоятельство снижает потребную вычислительную мощность и позволяет также сильно уменьшить размер собственно подписи (с 2 по 1024 до 2 по 160 бит). Стойкость же подписи остается на уровне 1024 числа *P*.

Система использует общие для всех абонентов параметры: *P*, *Q* и примитивный элемент *G*.

Каждый абонент системы генерирует секретный ключ

$$x: 1 < x < Q - 1$$

Открытый ключ Y вычисляют как: $Y = G^x \bmod P$

Абонент, желающий подписать сообщение *M*, выбирает сеансовый ключ $k: 1 < k < Q - 1,$ дополнительных требований к *k* не выдвигается, так как *Q* – простое число.

Подпись представляет собой пару: (R,S)

$R = (G^k \bmod P) \bmod Q$ - первая часть подписи, не зависящая от подписываемого сообщения. *R* служит для неявной передачи эфемерного ключа и, ввиду независимости от подписываемого сообщения, может быть выработано заранее в фоновом режиме.

$S = (M + x \cdot R)k^{-1} \bmod Q$ - вторая часть подписи, связанная с сообщением *M*.

Для проверки подписи необходимо вычислить два промежуточных параметра:

$A = M \cdot S^{-1} \pmod Q$

$B = R \cdot S^{-1} \pmod Q$

Подпись считается верной при выполнении равенства:

$R' = (G^A \cdot Y^B \bmod P) \bmod Q$

Данная схема стала основой стандарта цифровой подписи DSS.

Задача 4.4. Для заданного сильно простого числа *P* и соответствующего простого числа *Q*, открытого текста *M* и порождающего элемента *G*, самостоятельно определить остальные параметры схемы электронно-цифровой подписи DSA и

осуществить подписание сообщения. Проверить правильность цифровой подписи.

Пример решения.

Дано:	Решение:
$Q = 101$ $P = 607$ $G = 182$ $M = 45$	$x: 1 < x < Q - 1;\ \ x = 83$ $Y = G^x \bmod P = 182^{83} \bmod 607 = 195$ $k: 1 < k < Q - 1;\ k = 75$ $k^{-1} \bmod Q = 75^{-1} \bmod 101 = 66$
Найти: x, Y, k, R, S, A, B Проверить: $R' = R$	$R = (G^k \bmod P) \bmod Q = = (182^{75} \bmod 607) \bmod 101$ $= 565 \bmod 101 = 6$

$$S = (M + x \cdot R) k^{-1} \bmod Q = (45 + 83 \cdot 60) \cdot 66 \bmod 101 = (45 + 31) \cdot 66 \bmod 101 =$$

$$A = M \cdot S^{-1} \bmod Q = 45 \cdot 67^{-1} \bmod 101 =$$

$$= 45 \cdot 98 \bmod 101 = 67$$

$$B = R \cdot S^{-1} \bmod Q = 60 \cdot 67^{-1} \bmod 101 =$$

$$= 60 \cdot 98 \bmod 101 = 22$$

Проверка:

$$R' = (G^A \cdot Y^B \bmod P) \bmod Q =$$

$$= (182^{67} \cdot 195^{22} \bmod 607) \bmod 101 =$$

$$= (532 \cdot 559 \bmod 607) \bmod 101 = 565 \bmod 101 = 6$$

$$R' = R$$

Задача 4.4’. Решить задачу 4.4 с самостоятельным нахождением по данному Q сильно простого числа P и порождающего элемента G.

Варианты заданий к четвертой части

Вариант	Задача 4.1			Зад. 4.2		Зад. 4.3	Зад 4.4'	
	p	*q*	*m*	*P*	*M*	*Q*	*Q*	*M*
1	101	191	50	283	33	41	11	5
2	103	193	51	293	34	43	13	6
3	107	197	52	307	35	47	17	7
4	109	199	53	311	36	53	19	8
5	113	211	54	313	37	59	23	9
6	127	223	55	317	38	61	29	4
7	131	227	56	331	39	67	31	5
8	137	229	57	337	40	71	37	6
9	139	233	58	347	41	73	41	7
10	149	239	59	349	42	79	43	8
11	151	241	60	353	43	83	47	9
12	157	251	61	359	44	89	53	10
13	163	257	62	367	45	97	59	11
14	167	263	63	373	46	101	61	12
15	173	269	64	379	47	11	67	13
16	179	271	65	383	48	13	71	14
17	181	277	66	389	49	17	73	15
18	191	281	67	397	50	19	79	16
19	193	283	68	401	51	23	83	17
20	197	293	69	409	52	29	89	18
21	199	307	70	419	53	31	97	19
22	211	181	71	421	54	37	101	20

ЧАСТЬ 5. ЭЛЛИПТИЧЕСКИЕ КРИВЫЕ НАД КОНЕЧНЫМ ПОЛЕМ

В данном пособии рассматриваются только задачи, решаемые в конечной аддитивной группе точек эллиптической кривой над простым конечным полем характеристики $p>3$.

Конечной группой точек эллиптической кривой называют совокупность точек с целочисленными координатами удовлетворяющих короткой аффинной форме однородного уравнения Вейерштрасса над конечным полем характеристики p [3]:

$$E: y^2 = x^3 + ax + b (\mathrm{mod}\ p)$$

Кривая также характеризуется:

дискриминантом $\Delta = -16(4a^3 + 27b^2)$

и j-инвариантом $j(E) = -\dfrac{1728(4a)^3}{\Delta}$.

В данной аддитивной группе точек определены две операции – сложение и удвоение, которые выполняются в соответствии с групповым законом:

Пусть $P_1(x_1, y_1)$, $P_2(x_2, y_2)$ и, если $P_3(x_3, y_3) = P_1 + P_2 \neq O$, то координаты x_3, y_3 вычисляются как:

$$x_3 = \lambda^2 - x_1 - x_2$$

$$y_3 = (x_1 - x_3)\lambda - y_1$$

где при $x_1 \neq x_2$

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1}$$

а при $x_1 = x_2,\ y_1 \neq 0$

$$\lambda = \frac{3x_1^2 + a}{2y_1}$$

При создании криптосистем на основе эллиптических кривых важно знать порядок группы точек эллиптической кривой. Он характеризует стойкость системы и является одним из ее параметров.

На практике порядок группы точек эллиптической кривой и ее циклической подгруппы находится при помощи алгоритма

Шуфа [5], который в данном пособии не приводится. Определим порядок группы точек методом прямого перебора на примере кривой над полем характеристики $p = 23 \equiv 3 (\mathrm{mod}\ 4)$, характеризуемой уравнением [11]:

$E: y^2 = x^3 + x + 1 (\mathrm{mod}\ p)$

Ее дискриминант:

$\Delta = -16(4a^3 + 27b^2) = 7 \cdot (4 \cdot 1^3 + 4 \cdot 1^2) (\mathrm{mod}\ 23) = 10 \neq 0$, - кривая не аномальная и не суперсингулярная.

Для каждого элемента поля F_p вычислим y^2:

При помощи вычисления символа Лежандра по упрощенной зависимости:

$L(a,p) = a^{\frac{p-1}{2}} (\mathrm{mod}\ p)$

определим возможность извлечения квадратного корня.

В случае $L(a,p) = -1 = 22 (\mathrm{mod}\ 23)$ – на кривой нет целочисленных точек с таким x.

В случае $L(a,p) = 0$ – мы нашли нулевую точку (x, 0).

В случае $L(a,p) = 1$ произведем извлечение квадратного корня по упрощенной методике и определим:

$y_{1,2} = \pm a^{\frac{p+1}{4}} (\mathrm{mod}\ p)$

Результаты вычислений сведены в таблицу 5.

Таблица 5

x	$a = y^2 \bmod p == x^3 + x + 1 (\mathrm{mod}\ p)$	$L(a,p) == a^{\frac{p-1}{2}} (\mathrm{mod}\ p)$	$y_{1,2} = \pm a^{\frac{p+1}{4}} (\mathrm{mod}\ p)$	
			y_1	y_2
0	1	1	1	22
1	3	1	7	16
2	11	22(-1)	-	-
3	8	1	10	13
4	0	0	0	-
5	16	1	4	19
6	16	1	4	19
7	6	1	11	12
8	15	22(-1)	-	-
9	3	1	7	16
10	22	22(-1)	-	-
11	9	1	3	20

12	16	1	4	19
13	3	1	7	16
14	22	22(-1)	-	-
15	10	22(-1)	-	-
16	19	22(-1)	-	-
17	9	1	3	20
18	9	1	3	20
19	2	1	5	18
20	17	22(-1)	-	-
21	14	22(-1)	-	-
22	22	22(-1)	-	-

Из таблицы видно, что количество точек с целочисленными координатами $Q = 27$. 27 – составное число, поэтому в группе существует несколько циклических подгрупп.

Задача 5.1. Для заданной эллиптической кривой E над конечным полем и ее точек P_1 и P_2 , осуществить удвоение точки P_1 и сложить результат с точкой P_2. Проверить принадлежность результата к группе точек эллиптической кривой.

Пример решения

Дано:

$E: y^2 = x^3 + x + 1 \pmod{23}$

$P_1(18, 20)$, $P_2(7,11)$

Найти:

$Q = [2]P_1 + P_2$

Проверить:

принадлежность Q эллиптической группе E

Решение:

$$P_3(x_3, y_3) = [2]P_1(x_1, y_1)$$

$$x_1 = x_2,\ y_1 \neq 0$$

$$\lambda = \frac{3x_1^2 + a}{2y_1} =$$

$$= (3 \cdot 18^2 + 1) \cdot (2 \cdot 20)^{-1} \pmod{23} =$$

$$= (3 \cdot 2 + 1) \cdot 17^{-1} \pmod{23} =$$

$$= 7 \cdot 19 \pmod{23} = 18$$

$$x_3 = \lambda^2 - 2x_1 = 18^2 - 2 \cdot 18 \pmod{23} =$$

$= 2 - 13 (\mathrm{mod}\ 23) = 1$

$y_3 = (x_1 - x_3)\lambda - y_1 =$
$= (18 - 12) \cdot 18 - 20 (\mathbf{mod}\ 23) = 19$

$Q(x_4, y_4) = P_3(x_3, y_3) + P_2(x_2, y_2); \quad x_3 \neq x_2$

$\lambda = \frac{y_2 - y_3}{x_2 - x_3} = (11 - 19) \cdot (7 - 12) (\mathbf{mod}\ 23) = \mathbf{15 \cdot 18} (\mathbf{mod}\ 23) = 17$

$x_4 = \lambda^2 - x_2 - x_3 = 17^2 - 7 - 12 \left(\mathbf{mod}\ 23\right) = \mathbf{13 - 19} \left(\mathbf{mod}\ 23\right) = 17$

$y_4 = (x_2 - x_4)\lambda - y_2 = (7 - 17) \cdot 17 - 11 \left(\mathbf{mod}\ 23\right) = \mathbf{13 \cdot 17 - 11} (\mathbf{mod}\ 23) = 3$

$Q(17, 3) \in E$

Умножение точки эллиптической кривой на скаляр $[x]P$ производится с применением схемы Горнера аналогично бинарному модульному экспоненцированию в задаче 2.3.

Задача 5.2. Для заданной эллиптической кривой E над конечным полем, ее точки P и числа x , осуществить умножение $[x]$P, воспользовавшись схемой Горнера. Проверить принадлежность результата к группе точек эллиптической кривой.

Дано:	Решение:
$E: y^2 = x^3 + x + 1$ (mod 23) P (1, 16); $x = 9$	$Q(x_3, y_3) = [9]P(x_1, y_1) = = [2]([2]([2](1,16))) + (1,16)$
Найти: $Q = [9]P$ Проверить: принадлежность Q группе E	$x_1 = x_2,\ y_1 \neq 0$ $\lambda = (3 \cdot 1^2 + 1) \cdot (2 \cdot 16)^{-1} (\mathrm{mod}\ 23) = 3$ $x_3 = 3^2 - 2 \cdot 1 (\mathrm{mod}\ 23) = 7$ $y_3 = (1 - 7) \cdot 3 - 16 (\mathbf{mod}\ 23) = 12$

$$x_1 = x_2,\ y_1 \neq 0$$
$$\lambda = (3 \cdot 7^2 + 1) \cdot (2 \cdot 12)^{-1} (\mathrm{mod}\ 23) = 10$$
$$x_3 = 10^2 - 2 \cdot 7 (\mathrm{mod}\ 23) = 17$$
$$y_3 = (7 - 17) \cdot 10 - 12 (\mathbf{mod}\ 23) = 3$$

$$x_1 = x_2,\ y_1 \neq 0$$
$$\lambda = (3 \cdot 17^2 + 1) \cdot (2 \cdot 3)^{-1} (\mathrm{mod}\ 23) = 22$$
$$x_3 = 22^2 - 2 \cdot 17 (\mathrm{mod}\ 23) = 13$$
$$y_3 = (17 - 13) \cdot 22 - 3 (\mathbf{mod}\ 23) = 16$$
$$Q(x_3, y_3) = (13, 16) + (1,16) = (9,7)$$
$$\lambda = (16 - 16) \cdot (13 - 1)^{-1} = 0$$
$$x_3 = 0^2 - 13 - 1 = 9$$
$$y_3 = (13 - 1) \cdot 0 - 16 = 7$$
$$Q(9,7) \in E$$

Ниже приведены примеры реализации криптосистем на основе эллиптических кривых.

Протокол Диффи-Хеллмана на эллиптических кривых

Протокол предназначен для совместной выработки секретного ключа с использованием открытого канала связи [6]. В основе протокола лежит трудность задачи дискретного логарифмирования на эллиптических кривых над конечным полем. Общими (и общедоступными) параметрами системы являются кривая E над конечным полем характеристики порядка 160 бит, порядок группы точек Q и точка эллиптической кривой P.

Каждый из двух абонентов A и B соответственно выбирают секретные ключи x и y.

При этом $x: 1 < x < Q - 1$ и $y: 1 < y < Q - 1$

Далее абоненты обмениваются открытыми сообщениями:

А передает В: $[x]P$ В передает А: $[y]P$

Противник, зная P, но, не зная x и y, вынужден решать вычислительно сложную задачу дискретного логарифмирования на эллиптической кривой.

Далее абонент А вычисляет: $k = x\text{-координата}([x][y]P)$

Абонент В вычисляет: $k' = x\text{-координата}([y][x]P)$

Причем, в силу коммутативности операции умножения точки эллиптической кривой на скаляр: $k = k'$.

Задача 5.3. Для заданной кривой E над конечным полем, ее точки P и порядка группы Q , самостоятельно определить остальные параметры протокола Диффи-Хеллмана и осуществить совместную выработку симметричного сеансового ключа. Проверить совпадение ключей обоих абонентов. (При решении рекомендуется использовать программу на любом из языков программирования, реализующую сложение и удвоение точек кривой).

Дано:	Решение:
$E: y^2 = x^3 + x + 3 \pmod{199}$ $Q = 197$, $P(1, 76)$ $x = 29$, $y = 153$	$x: 1 < x < Q - 1; x = 29$ $[x]P = [29](1,76) = (113,191)$ $y: 1 < y < Q - 1; y = 153$ $[y]P = [153](1,76) = (185,35)$ $[x][y]P = [29](185,35) = (172,81)$ $[y][x]P = [153](113,191) = (172,81)$ $k' = k = x\text{-коорд.}(172,81) = 172$
Найти: $[x]P$,$[y]P$, k, k' Проверить: $k' = k$	

Цифровая подпись EC-DSA

В целом схема аналогична классической DSA, но вместо вычислительно затратной операции модульного экспоненцирования в конечном поле большой характеристики использует умножение точки на скаляр в аддитивной группе точек эллиптической кривой.

Каждый абонент системы генерирует секретный ключ

$x: 1 < x < Q - 1$

Открытый ключ Y представляет собой точку эллиптической кривой:

$$Y = [x]P$$

Абонент, желающий подписать сообщение *M*, выбирает сеансовый ключ:

$$k: 1 < k < Q - 1$$

Подпись представляет собой пару: (R, S)

$R = x$-координата$([k]P)$ - первая часть подписи, не зависящая от подписываемого сообщения. *R* служит для неявной передачи эфемерного ключа и, ввиду независимости от подписываемого сообщения, может быть выработано заранее в фоновом режиме.

$S = (M + x \cdot R)k^{-1} (\mathrm{mod} Q)$ - вторая часть подписи, связанная с сообщением *M*.

Для проверки подписи необходимо вычислить два промежуточных параметра:

$$A = M \cdot S^{-1} (\mathrm{mod}\ Q)$$

$$B = R \cdot S^{-1} (\mathrm{mod}\ Q)$$

Подпись считается верной при выполнении равенства:

$$R' = x\text{-координата}([A]P + [B]Y)$$

Задача 5.4. Для заданной кривой *E* над конечным полем, ее точки *P*, порядка группы *Q* и открытого текста *M*, самостоятельно определить остальные параметры схемы электронно-цифровой подписи EC-DSA и осуществить подписание сообщения. Проверить правильность цифровой подписи. (При решении рекомендуется использовать программу на любом из языков программирования, реализующую сложение и удвоение точек кривой).

Пример решения [10]

Дано:	Решение:
$E: y^2 = x^3 + x$ $(\mathrm{mod}\ 199)$ $Q = 197$ $P(x, y) = P(1, 76)$ $M = 68$, $x = 29$	$x: 1 < x < Q - 1; x = 29$ $Y = [x]P = [29](1, 76) = (113, 191)$ $k: 1 < k < Q - 1; k = 153$ $R = x\text{-коор}_{\text{д.}}([k]P) = = x\text{-коор}_{\text{д.}}([153](1, 76)) =$ $= x\text{-коор}_{\text{д.}}((185, 35)) = 185$
Найти: *x*, *Y*, *k*, *R*,	

S, A, B, R' Проверить: $R' = R$	$S = (M + x \cdot R)k^{-1}(\mathrm{mod}\ Q) = = (68 + 29 \cdot 185)153^{-1}(\mathrm{mod}\ 197) = (68 +$

$114 \cdot 94(\mathrm{mod}\ 197) = 78$

$A = M \cdot S^{-1}(\mathrm{mod}\ Q) = 68 \cdot 78^{-1}(\mathrm{mod}\ 197) =$

$= 68 \cdot 48(\mathrm{mod}\ 197) = 112$

$B = R \cdot S^{-1}(\mathrm{mod}\ Q) = 185 \cdot 78^{-1}(\mathrm{mod}\ 197) =$

$= 185 \cdot 48(\mathrm{mod}\ 197) = 15$

$R' = x\text{-коор}_{\text{д.}}([A]P + [B]Y) = = x\text{-коор}_{\text{д.}}([112](1, 76) + [15](113, 191)) =$

$= x\text{-коор}_{\text{д.}}\left((111, 60) + \left(122, 140\right)\right) = x\text{-коор}_{\text{д.}}(185, 35) = 185$

$R' = R$

Примеры листингов программ для решения задач 5.3, 5.4 приведены в приложениях 2 и 3.

Варианты заданий для пятой части

Вариант	Задача 5.1		Задача 5.2		Задача 5.3		Зад. 5.4	
	P_1	P_2	x	P	x	y	M	x
1	(0, 1)	(18, 20)	8	(5, 4)	144	30	69	4
2	(1, 7)	(19, 5)	7	(5, 19)	143	31	70	5
3	(1, 22)	(19, 18)	6	(6, 4)	142	32	71	6
4	(3, 10)	(0, 22)	5	(6, 19)	141	33	72	7
5	(3, 13)	(0, 1)	9	(7, 11)	140	34	73	8
6	(5, 4)	(1, 7)	8	(7, 12)	139	35	74	9
7	(5, 19)	(1, 22)	7	(9, 7)	138	36	75	10
8	(6, 4)	(3, 10)	6	(9, 16)	137	37	76	11
9	(6, 19)	(3, 13)	5	(11, 3)	136	38	77	12
10	(7, 11)	(5, 4)	4	(11, 20)	135	39	78	13
11	(7, 12)	(5, 19)	9	(12, 4)	134	25	79	14
12	(9, 7)	(6, 4)	8	(12, 19)	98	26	80	15
13	(9, 16)	(6, 19)	7	(13, 7)	97	27	81	16
14	(11, 3)	(7, 11)	6	(13, 16)	96	41	82	17
15	(11, 20)	(7, 12)	5	(17, 3)	95	42	83	18
16	(12, 4)	(9, 7)	7	(17, 20)	94	43	84	19
17	(12, 19)	(9, 16)	6	(18, 3)	93	44	85	20
18	(13, 7)	(11, 3)	5	(18, 20)	92	45	86	21
19	(13, 16)	(11, 20)	4	(19, 5)	91	46	87	22
20	(17, 3)	(12, 4)	9	(19, 18)	90	47	88	23
21	(17, 20)	(12, 19)	8	(0, 22)	89	48	89	24
22	(18, 3)	(13, 7)	7	(0, 1)	88	49	90	25

Примечания:

Для задач 5.1, 5.2: $E: y^2 = x^3 + x + 1 (\mathrm{mod}\ 23)$,

Для задач 5.3, 5.4: $E: y^2 = x^3 + x + 3 (\mathrm{mod}\ 199)$, $Q = 197$, $P(1, 76)$

ЗАКЛЮЧЕНИЕ

Учебное пособие «Криптографические методы защиты информации» содержит материалы, которые следует использовать при изучении дисциплин криптографической направленности.

Изучение криптографии является одним из важнейших аспектов в специальной подготовке специалистов по защите информации [2]. Успешное освоение данного курса позволяет изучить современные методы криптографической защиты информации. Такие знания необходимы специалисту при разработке систем защиты информации на различных уровнях обеспечения информационной безопасности личности, организации и в целом Российской Федерации. Построение систем защиты информации в современных условиях становится все более актуальным ввиду того, что уровень развития средств несанкционированного добывания информации очень высок, доступность программных средств шпионского толка превращает задачу нелегального добывания информации из уникальной и рискованной операции в достаточно простую задачу. Это значительно увеличивает риск получения больших объемов информации злоумышленниками на безопасном расстоянии от её источников путем её перехвата при обработке и хранении. Практика показывает, что эффективная защита информации с учетом этих тенденций возможна при активном использовании криптографических методов и средств защиты информации.

Содержание предлагаемого учебного пособия не претендует на обобщение всего многообразия вопросов криптографической защиты информации, но авторы надеются, что их работа будет с благодарностью оценена всеми, кому приходится сталкиваться с решением задач информационной безопасности в рамках профессиональной деятельности.

СПИСОК ИСПОЛЬЗОВАННОЙ И РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ.

1. Аверченков В.И. Криптографические методы защиты информации: учебное пособие./ В.И. Аверченков, М.Ю. Рытов, С.А. Шпичак – Брянск: Изд-во БГТУ, 2011. – 216 с.
2. Основы криптографии: учебное пособие./ А.П. Алферов, А.Ю. Зубов, А.С. Кузьмин, А.В. Черемушкин. – М.: Гелиос-АРВ, 2001. – 480 с.
3. Элементарное введение в эллиптическую криптографию: Алгебраические и алгоритмические основы./ А.А. Болотов, С.Б. Гашков, А.Б. Фролов, А.А. Часовских. – М.: КомКнига, 2006. – 328 с.
4. Элементарное введение в эллиптическую криптографию: Протоколы криптографии на эллиптических кривых./ А.А. Болотов, С.Б. Гашков, А.Б. Фролов, А.А. Часовских – М.: КомКнига, 2006. – 280 с.
5. Василенко О.Н. Теоретико-числовые алгоритмы в криптографии./ О.Н. Василенко – М.: МЦНМО, 2003. – 328 с.
6. Запечников С.В. Криптографические протоколы и их применение в финансовой и коммерческой деятельности: учебное пособие./ С.В. Запечников. - М.: Горячая линия-Телеком, 2007. – 320 с.
7. Кнут. Д. Искусство программирования. Т.2. Получисленные алгоритмы./Д. Кнут. Пер. с англ. – М.: Вильямс – 2001 г.638 с.
8. Маховенко Е.Б. Теоретико-числовые методы в криптографии: учебное пособие./ Е.Б. Маховенко. – М.: Гелиос-АРВ, 2006. – 320 с.
9. Рябко Б.Я. Криптографические методы защиты информации: учебное пособие./ Б.Я. Рябко, А.Н. Фионов. – М.: Горячая линия-Телеком, 2005. – 229 с.
10. Смарт. Н. Криптография./Н. Смарт. Пер с англ. – М.: Техносфера, 2005. – 528 с.
11. Столлингс В. Криптография и защита сетей. Принципы и практика. 2-е изд./В. Столингс. Пер с англ. – М.: Вильямс, 2001. – 672 с.
12. Черемушкин А.В. Лекции по арифметическим алгоритмам в криптографии./ А.В. Черемушкин. – М.:МЦНМО, 2002. – 104 с.
13. Шнайер Б. Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си./ Б. Шнайер. Пер. с англ. – М.: ТРИУМФ, 2003. – 816 с.

Приложение 1

Таблица простых чисел* p *до 997

2	3	5	7	11	13	17	19	23	29	31	37
41	43	47	53	59	61	67	71	73	79	83	89
97	101	103	107	109	113	127	131	137	139	149	151
157	163	167	173	179	181	191	193	197	199	211	223
227	229	233	239	241	251	257	263	269	271	277	281
283	293	307	311	313	317	331	337	347	349	353	359
367	373	379	383	389	397	401	409	419	421	431	433
439	443	449	457	461	463	467	479	487	491	499	503
509	521	523	541	547	557	563	569	571	577	587	593
599	601	607	613	617	619	631	641	643	647	653	659
661	673	677	683	691	701	709	719	727	733	739	743
751	757	761	769	773	787	797	809	811	821	823	827
829	839	853	857	859	863	877	881	883	887	907	911
919	929	937	941	947	953	967	971	977	983	991	997

Примечание: темным цветом выделены $p \equiv 3 \pmod 4$

Приложение 2

Листинг программы для решения задач 5.3, 5.4 на языке C#

Программа реализует обобщенный алгоритм Евклида, сложение и удвоение точек.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication1
{
   public class Operations
   {
      public int Mod(int a,int p)
      {
         if (a < 0) { a = a + p; }
         return a;
      }
      public int Evc(int h, int k) //Алгоритм Евклида
      {
         int[,] mt = new int[20, 7];

         for (int i = 0; i < 20; i++)
         {
            for (int j = 0; j < 7; j++) mt[i, j] = -1;
         }
         mt[0, 0] = 1;
         mt[0, 1] = 0;
         mt[0, 2] = h;
         mt[0, 3] = 0;
         mt[0, 4] = 1;
         mt[0, 5] = k;
         mt[0, 6] = (h - (h % k)) / k;
         int n = 0;
         while (mt[n, 5] != 0)

  {
```

```
            n++;
            int m = 0;
            for (m = 0; m < 3; m++)
            {
               mt[n, m] = mt[n - 1, m + 3];
            }
            for (m = 3; m < 6; m++)
            {
               mt[n, m] = mt[n - 1, m - 3] - mt[n - 1, m] * mt[n - 1, 6];
            }
            if (mt[n, 5] != 0)
            {
               mt[n, 6] = (mt[n, m - 4] - (mt[n, 2] % mt[n, 5])) / mt[n, 5];
            }
         }
         k = mt[n - 1, 4];
         if (k < 0)  {k += h; }
         return (k);
      }
   }
   class Program
   {
      static void Main(string[] args)
      {
         Operations op1 = new Operations();
         int x=0;
         int y=0;
         int k;
         int p = 199;
         Console.WriteLine("Если требуется удвоение, нажмите 0,
если сложение 1");
         k = Convert.ToInt32(Console.ReadLine());
         if (k==0) //Удвоение
         {
            Console.WriteLine("Введите координаты, для удвоения");
            int x1;
            Console.Write("x1=");
            x1 = Convert.ToInt32(Console.ReadLine());
            int y1;
```

```
    Console.Write("y1=");
    y1 = Convert.ToInt32(Console.ReadLine());
    int a;
    int a1 = 2*y1 % p;
    a1 = op1.Evc(p, a1);
    a = (3 * x1 * x1) % p;
    a = a + 1;
    a = (a * a1) % p;
    x = (a * a) % p;
    x = x - ((2 * x1)%p);
    x = op1.Mod(x, p);
    y = x1 - x;
    y = op1.Mod(y, p);
    y = (y * a) % p;
    y = y - y1;
    y = op1.Mod(y, p);
}
else //Сложение
{
    Console.WriteLine("Введите координаты, для сложения");
    int x1;
    Console.Write("x1=");
    x1 = Convert.ToInt32(Console.ReadLine());
    int y1;
    Console.Write("y1=");
    y1 = Convert.ToInt32(Console.ReadLine());
    int x2;
    Console.Write("x2=");
    x2 = Convert.ToInt32(Console.ReadLine());
    int y2;
    Console.Write("y2=");
    y2 = Convert.ToInt32(Console.ReadLine());
    int a;
    int a1 = x2-x1;
    a1 = op1.Mod(a1, p);
    a1 = op1.Evc(p, a1);
    a = y2 - y1;
    a = op1.Mod(a, p);
    a = (a * a1) % p;
```

```
                x = (a * a) % p;
                x = x - x1;
                x = op1.Mod(x, p);
                x = x - x2;
                x = op1.Mod(x, p);
                x = x % p;
                y = (x1 - x);
                y = op1.Mod(y, p);
                y = (y * a) % p;
                y = y - y1;
                y = op1.Mod(y, p);
            }
            Console.WriteLine("x=" + x);
            Console.WriteLine("y=" + y);
            Console.ReadKey();
        }
    }
}
```

Приложение 3

Листинг программы для решения задач 5.3, 5.4 на языке Pascal

Программа реализует обобщенный алгоритм Евклида, сложение и удвоение точек, а также умножение точки на скаляр с использованием схемы Горнера.

```
var
 a:array[1..10]of boolean;
 xn,yn,p,k,xk,yk,i,j,x,y:integer;
function ee(pe,ae:integer):integer;
 var u1,u2,u3,v1,v2,v3,t1,t2,t3,q:integer;
 begin
  u1:=1; u2:=0; u3:=pe;
  v1:=0; v2:=1; v3:=ae;
  while v3>0 do
   begin
    q:=trunc(u3/v3);
    t1:=u1-v1*q;
    t2:=u2-v2*q;
    t3:=u3-v3*q;
    u1:=v1;
    u2:=v2;
    u3:=v3;
    v1:=t1;
    v2:=t2;
    v3:=t3;
   end;
  if u3=1
   then if u2>0
         then ee:=u2
         else ee:=pe+u2
   else ee:=0
 end;
procedure dub(x1,y1,pm:integer; var x3,y3:integer);
 var lam:integer;
 begin
  lam:=((3*x1*x1+1)mod pm)*ee(pm,(2*y1)mod pm);
  lam:=lam mod pm;
```

```
  x3:=(lam*lam)mod pm  - (2*x1)mod pm;
  while x3<0 do x3:=x3+pm;
  x3:=x3 mod pm;
  y3:=(x1-x3)*lam - y1;
  while y3<0 do y3:=y3+pm;
  y3:=y3 mod pm;
 end;
procedure ad(x1,y1,x2,y2,pm:integer; var x3,y3:integer);
 var lam,chz,znam:integer;
 begin
  chz:=y2 - y1;
  while chz<0 do chz:=chz+pm;
  znam:=x2 - x1;
  while znam<0 do znam:=znam+pm;
  lam:=(chz)*ee(pm,znam);
  lam:=lam mod pm;
  x3:=(lam*lam)mod pm  - x1 - x2;
  while x3<0 do x3:=x3+pm;
  y3:=(x1-x3)*lam - y1;
  while y3<0 do y3:=y3+pm;
  y3:=y3 mod pm;
 end;
begin
 write('x=');readln(xn);
 write('y=');readln(yn);
 {write('p=');readln(p);}p:=199;
 write('k=');readln(k);
 j:=0;
 while k>0 do
  begin
   j:=j+1;
   if odd(k) then a[j]:=true
             else a[j]:=false;
   k:=k div 2
  end;
 x:=xn;y:=yn;
 for i:=j-1 downto 1 do
  begin
   if a[i]
```

```
    then
     begin
      dub(x,y,p,xk,yk);x:=xk;y:=yk;
      ad(x,y,xn,yn,p,xk,yk)
     end
    else dub(x,y,p,xk,yk);
   x:=xk;y:=yk;
  end;
 {dub(xn,yn,p,x,y);
 for i:=2 to k-1 do
  begin
   ad(x,y,xn,yn,p,xk,yk);
   x:=xk;
   y:=yk;
  end;}
 writeln(x,' ',y)
end.
```