МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждения образования

«БЕЛОРУССКИЙ

ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационных систем и технологий

Направление специальности Информационные системы и технологии

**ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №8**

**ПО ДИСЦИПЛИНЕ**

**«ЗАЩИТА ИНФОРМАЦИИ**

**И НАДЕЖНОСТЬ ИНФОРМАЦИОННЫХ СИСТЕМ»**

Выполнил: студент 3 курса 1 группы

Кашперко Василиса Сергеевна

Проверил: ассистент

Нистюк Ольга Александровна

Минск 2023

**Лабораторная работа №8**

**Исследование асимметричных шифров RSA и Эль-Гамаля**

**Цель:** изучение и приобретение практических навыков разработки и использования приложений для реализации асимметричных шифров RSA и Эль-Гамаля.

**Теоретические сведения**

**Математические основы асимметричных шифров**

Асимметричная криптография основана на сложности решения некоторых математических задач. По существу, таких задач две:

– разложение больших чисел на простые сомножители (задача факторизации);

– вычисление дискретного логарифма в конечном поле, а также вычислительные операции над точками эллиптической кривой

Все системы асимметричного зашифрования основаны:

– на проблеме факторизации (RSA)

– на проблеме дискретного логарифма (Эль-Гамаля)

**RSA**

Безопасность RSA основана на *трудности разложения на множители больших чисел*. Открытый и закрытый ключи являются функциями двух больших простых чисел. Предполагается, что восстановление открытого текста по шифртексту и открытому ключу эквивалентно разложению на множители двух больших чисел.

Используются два больших случайных числа *p* и *q* (для максимальной криптостойкости нужно выбирать их равной длины)для генерации тайного и открытого ключа (по сути – двух взаимосвязанных частей одного ключа, т.е. ключа, принадлежащего одному физическому лицу). Рассчитываем *n* = *pq.* Ключ состоит из *n*, *e*, *d*.

Выбираем случайным образом *e,* такое что *e* и (*p* – 1)(*q* – 1) являются взаимно простыми числами.

Затем находим *d*:





В результате получаем ключ из трех чисел, которые образуют две взаимосвязи  
публичный ключ (*e*, *n*); тайный ключ (*d*, *n*).

Зашифрование RSA: если шифруется сообщение *М*, состоящее из r блоков: , , …, , …, , то шифртекст С будет состоять из такого же числа (*r*) блоков, представляемых числами:



Расшифрование RSA:



**Эль-Гамаля**

Как подчеркивалось выше, безопасность алгоритма Эль-Гамаля, как и безопасность алгоритма Диффи – Хеллмана, основана на *трудности вычисления дискретных логарифмов*.

Алгоритм Эль-Гамаля фактически использует схему Диффи – Хеллмана, чтобы сформировать общий секретный ключ для абонентов, передающих друг другу сообщение, и затем сообщение шифруется путем умножения его на этот ключ.

*Генерация ключевой информации:*

Выбирается простое число *р*. Выбирается число (*g*, *g* < *p*), являющееся первообразным корнем числа *р* – очень важный элемент с точки зрения безопасности алгоритма.

Далее выбирается число *х* (*х* < *p*) и вычисляется последний компонент ключевой информации:



Отправка сообщения, зашифрованного ключами *p*, *g*, *y*. Расшифровка сообщения ключами *p*, *g*, *x*. Как видим, тайным является только *x*.

При больших *p* нахождение *x* затруднено.

Алгоритм Эль-Гамаля предназначен для шифрования сообщений и создания цифровой подписи. Этот алгоритм основан на сложности вычисления дискретного логарифма в конечном поле.

Хотя в теории возможно использование алгоритма Эль-Гамаля для расшифрования сообщений, это не является его основной функцией и не является практически осуществимым в общем случае. Для расшифрования сообщений, зашифрованных алгоритмом Эль-Гамаля, необходимо знать закрытый ключ, который был использован для шифрования сообщения.

Поэтому алгоритм Эль-Гамаля обычно используется только для шифрования сообщений и создания цифровой подписи, а не для расшифровки сообщений, зашифрованных им.

**Ход работы**

В ходе выполнения лабораторной работы мы разработали приложение, которое реализует и изображает табличную форму зависимости времени вычисления параметра *y* (Рис. 1), функционально заданного выражением вида:



Параметры:

* *а* (десятичные числа от 5 до 35; можно взять 1 или 2 числа),
* *х* (числа, желательно простые, из диапазона от 103 до 10100; для примера взять 5–10 чисел, равномерно распределенных в указанном диапазоне),
* *n* (для примера взять числа, в двоичном виде состоящие из 1024 и 2048 битов).

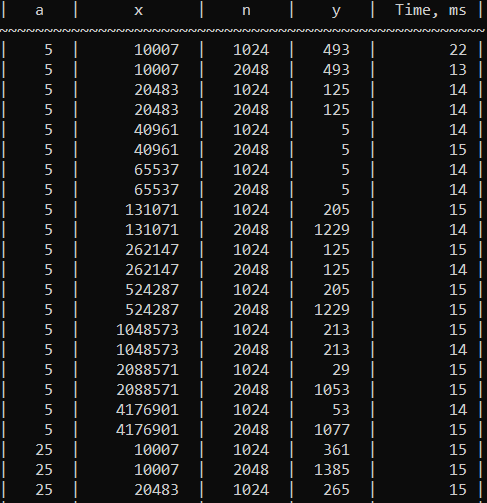


Рисунок 1 – Табличная форма зависимости времени вычисления параметра *y*

Мы выполняем вычисление значения *y* по заданной формуле:

*y = a^x* mod *n*,

где *a*, *x* и *n* - заданные параметры функции.

В этой формуле символ "^" означает возведение в степень, а "mod" - операцию нахождения остатка от деления.

В коде для программы реализован алгоритм вычисления *y* в функции,который можно описать следующим образом:

1. Инициализировать переменную *result* значением 1.
2. Преобразовать *a* в двоичную форму и сохранить результат в массив *bits*.
3. Для каждого бита *b* в массиве *bits* (начиная со старшего бита):
   * Возвести *result* в квадрат (т.е. умножить *result* на само себя).
   * Если *b* равен 1, умножить *result* на *a*.
   * Вычислить остаток от деления *result* на *n* и сохранить результат в *result*.
4. Вернуть *result* в качестве результата функции.

Данный алгоритм использует свойства арифметики остатков, которые позволяют вычислять значение y за время, зависящее от количества бит в x, а не от самого значения x.

После выполнения программы было замечено, что время вычисления параметра y составляет всего 0 мс.

Возможно, причина этого заключается в том, что операция выполняется настолько быстро, что таймер не успевает зафиксировать ее выполнение. Для решения этой проблемы была добавлена задержка в код с помощью метода Thread.Sleep().

Затем было разработано приложение в соответствии с целью лабораторной работы, при этом использовались доступные библиотеки и программный код.

Приложение реализует следующие операции:

• зашифрование и расшифрование на основе алгоритмов RSA (Рис. 2) и Эль-Гамаля (Рис. 3);

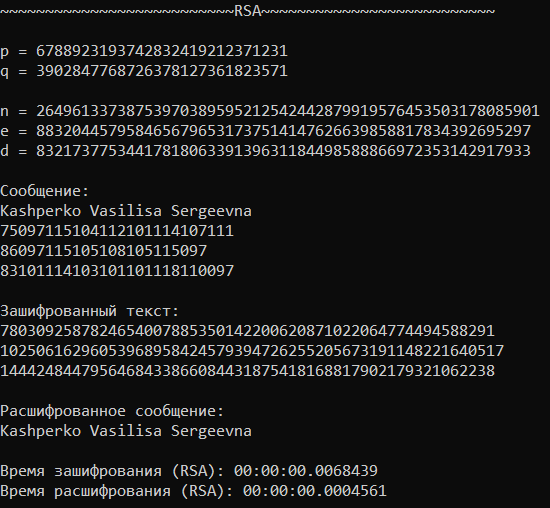


Рисунок 2 – Результат выполнения операций шифрования и дешифрования алгоритмом RSA

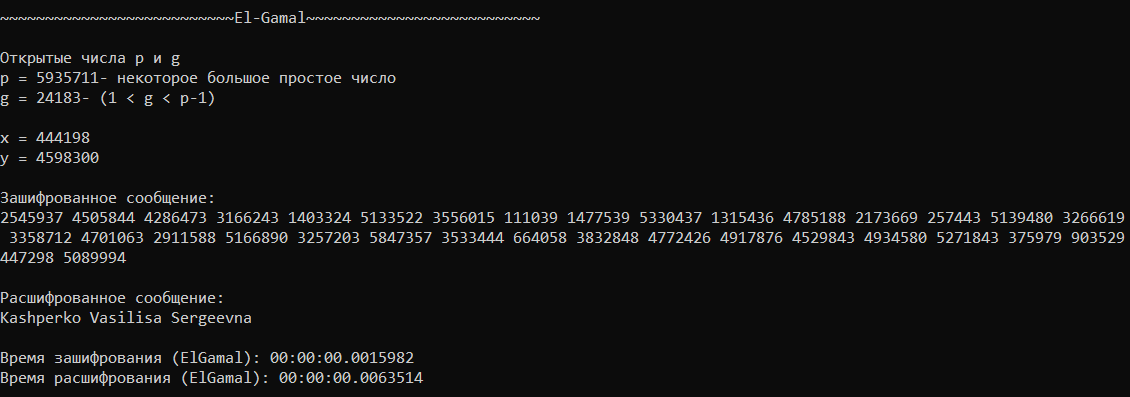


Рисунок 3 – Результат выполнения операций шифрования и дешифрования алгоритмом Эль-Гамаля

• определение времени выполнения операций.

После оценки времени выполнения операций шифрования и дешифрования, можно отметить, что оба алгоритма имеют приблизительно одинаковые значения, однако на шифрование RSA уходит больше времени, чем Эль-Гамаля.

Обратная ситуация при расшифровке: расшифрование методом Эль-Гамаля проходит медленнее, чем RSA.

**Исходное (23 знаков):** Kashperko Vasilisa Sergeevna

**RSA (166 знаков):**

7803092587824654007885350142200620871022064774494588291

10250616296053968958424579394726255205673191148221640517

1444248447956468433866084431875418168817902179321062238

**Эль-Гамаля:**

2545937 4505844 4286473 3166243 1403324 5133522 3556015 111039 1477539 5330437 1315436 4785188 2173669 257443 5139480 3266619 1478976 3209134 4285137 1054896 4211770 1552139 3344902 68185 1566516 2826326 1670823 3358712 4701063 2911588 5166890 3257203 5847357 3533444 664058 3832848 4772426 4917876 4529843 4934580 5271843 375979 903529 19557 1762668 418197 2817941 4728111 2749188 921479 3973565 1955002 4630872 2999917 1447298 5089994

Длина закодированного алгоритмом RSA сообщения приблизительно в 7 раз больше исходного.

Длина закодированного алгоритмом Эль-Гамаля сообщения приблизительно в 13 раз больше исходного.

Также мы установили и проверили приложение OpenSSL. OpenSSL – это система защиты и сертификации данных; SSL (Secure Socket Layer – система безопасных сокетов). Внутри OpenSSL имеются отдельные компоненты (утилиты), отвечающие за то или иное действие. OpenSSL поддерживает в том числе много различных стандартов шифрования. К их числу относится RSA. На рисунках 4 – 12 представлен процесс установки и интерфейса данной программы.

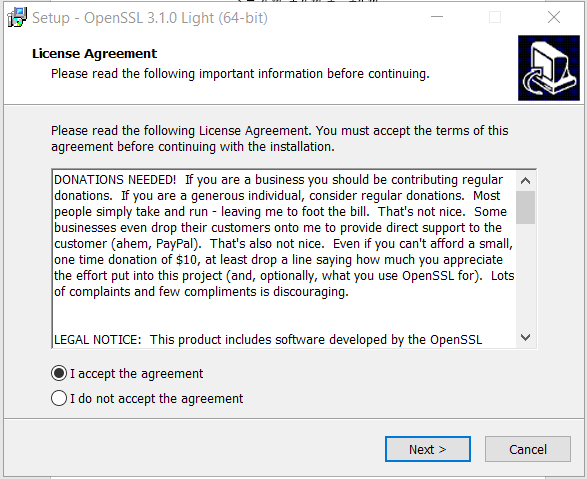


Рисунок 4 – Установка OpenSSL

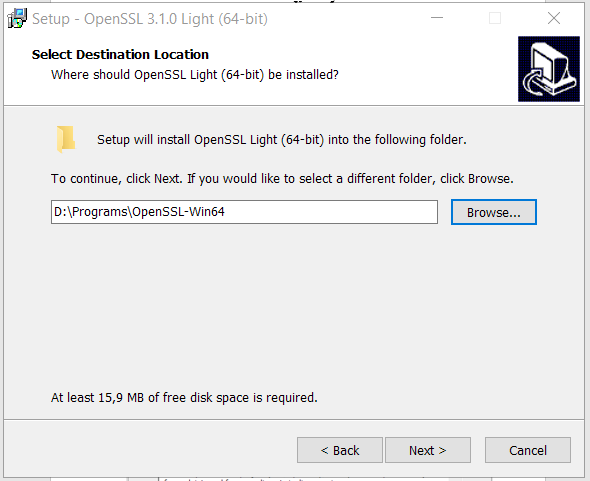


Рисунок 5 – Установка OpenSSL

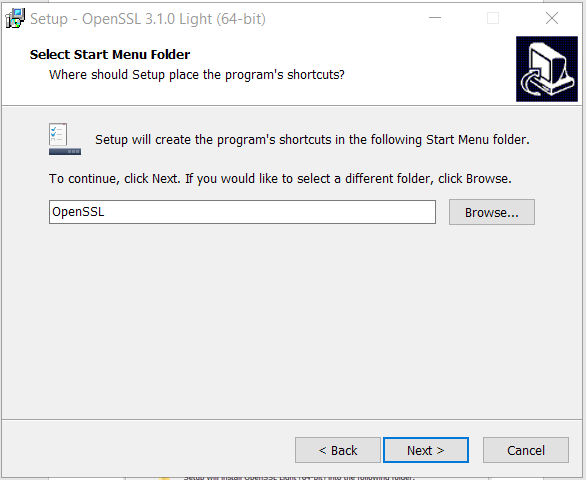


Рисунок 6 – Установка OpenSSL

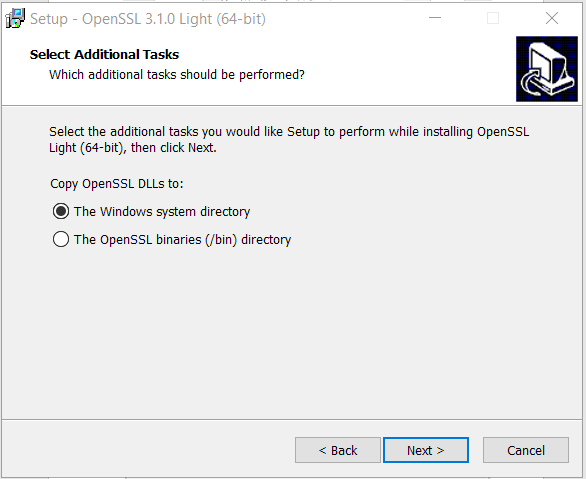


Рисунок 7 – Установка OpenSSL

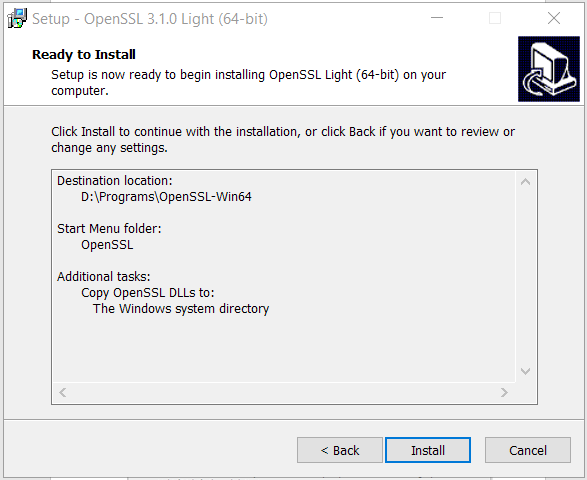


Рисунок 8 – Установка OpenSSL

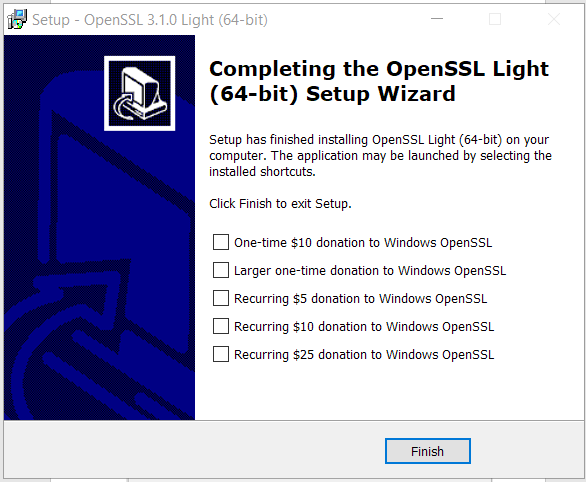


Рисунок 9 – Установка OpenSSL

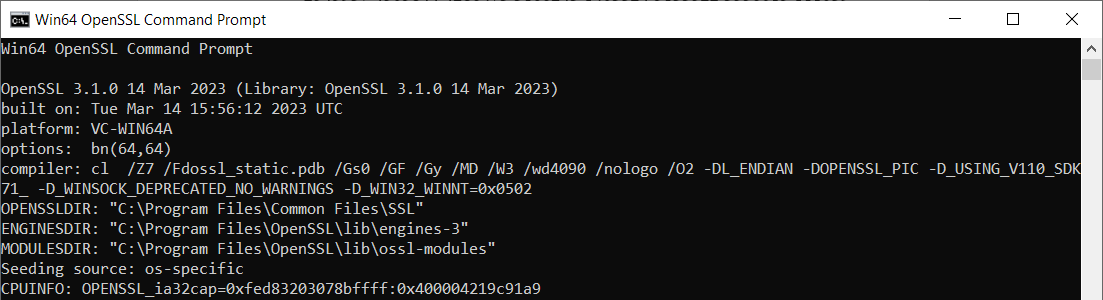


Рисунок 10 – Установка OpenSSL

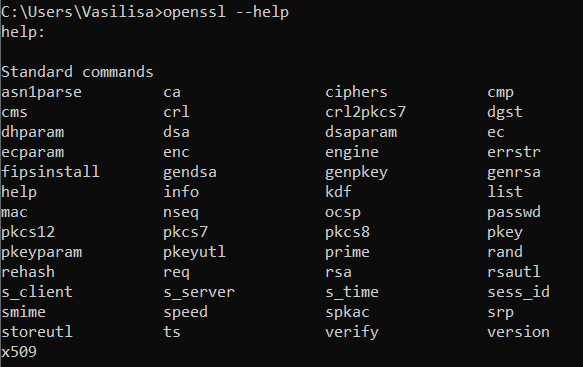


Рисунок 11 – Установка OpenSSL

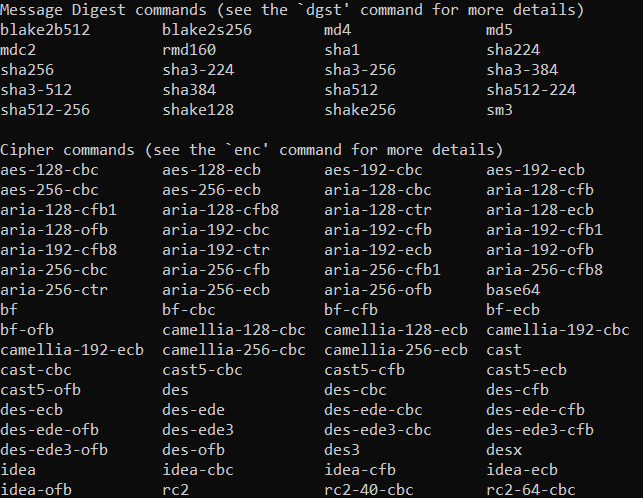


Рисунок 12 – Установка OpenSSL

**Вывод:** в рамках лабораторной работы были изучены два асимметричных шифра - RSA и Эль-Гамаля. RSA может использоваться для шифрования и подписи данных, в то время как Эль-Гамаля предназначен только для шифрования.

Было выявлено, что RSA обладает высокой производительностью и высокой стойкостью к взлому при правильной настройке параметров, однако его безопасность требует использования больших ключей.

С другой стороны, Эль-Гамаля требует больше вычислительных ресурсов для шифрования и расшифрования, но при этом обеспечивает более высокий уровень безопасности по сравнению с RSA при одинаковой длине ключа.

Таким образом, выбор конкретного шифра зависит от поставленных задач и требований к безопасности данных.

**Контрольные вопросы**

**1. Охарактеризовать алгоритмы RSA и Эль-Гамаля. Для каких целей они могут применяться?**

RSA и Эль-Гамаля являются асимметричными криптографическими алгоритмами, которые используются для защиты конфиденциальности данных, аутентификации и цифровой подписи в различных сферах, включая финансы, банковское дело и электронную коммерцию.

RSA основывается на сложности факторизации больших целых чисел и позволяет генерировать открытый и закрытый ключи для шифрования и дешифрования данных.

Алгоритм Эль-Гамаля основывается на сложности вычисления дискретного логарифма в конечных полях и также позволяет генерировать открытый и закрытый ключи для шифрования и дешифрования данных.

Оба алгоритма имеют преимущества и недостатки и могут использоваться в зависимости от конкретной ситуации. Они являются важными инструментами для защиты информации в цифровой эпохе.

**2. На чем основана криптостойкость алгоритмов RSA и Эль-Гамаля?**

Все системы асимметричного зашифрования основаны:

– на проблеме факторизации (RSA)

– на проблеме дискретного логарифма (Эль-Гамаля)

**3. Что такое первообразный корень?**

Первообразный корень по модулю *р* является таким числом, что его степени (, 1 ≤ *i* ≤ *p* – 1) дают все возможные по модулю р вычеты (остатки), которые взаимно просты с *p*.

**4. Найти первообразные корни (если они существуют) чисел (*р*): 13, 19, 23, 27, 31, 37, 39, 43.**

Для того чтобы найти первообразный корень по модулю *p*, где *p* - простое число, нужно найти наименьшее целое число *g*, такое что все числа от 1 до *p*-1 можно представить в виде , где *k* - натуральное число.

1. Для числа 13 первообразных корней нет.
2. Для числа 19 первообразный корень равен 2.
3. Для числа 23 первообразный корень равен 5.
4. Для числа 27 первообразных корней нет.
5. Для числа 31 первообразный корень равен 3.
6. Для числа 37 первообразный корень равен 5.
7. Для числа 39 первообразных корней нет.
8. Для числа 43 первообразный корень равен 3.

**5. Пусть пользователь А хочет передать пользователю В сообщение *М*, которое в некоторой кодировке соответствует числу 17 и зашифровано с помощью алгоритма RSA. Пользователь В имеет следующие ключевые параметры: *p* = 7, *q* = 11, *d* = 47. Описать процесс зашифрования сообщения пользователем А.**

Для зашифрования сообщения М пользователем А в кодировке RSA нужно использовать открытый ключ пользователя В, который состоит из модуля n и показателя шифрования e.

Модуль n для пользователя В вычисляется как произведение двух простых чисел p и q: n = p \* q = 7 \* 11 = 77.

Показатель шифрования e для пользователя В можно найти, зная закрытый ключ d. По определению, показатель шифрования e должен удовлетворять следующему условию:

d \* e ≡ 1 (mod (p-1)\*(q-1))

Подставляя значения для p и q, получаем:

d \* e ≡ 1 (mod 60)

Чтобы найти e, можно воспользоваться алгоритмом расширенного алгоритма Евклида, который позволяет найти наибольший общий делитель двух чисел и их линейную комбинацию. Применяя его к числам d и 60, получим:

60 = 1 \* 47 + 13

47 = 3 \* 13 + 8

13 = 1 \* 8 + 5

8 = 1 \* 5 + 3

5 = 1 \* 3 + 2

3 = 1 \* 2 + 1

Из последнего уравнения следует, что НОД(d, 60) = 1, и мы можем выразить обратный элемент d^-1 по модулю 60. Это можно сделать, решив линейное уравнение:

d \* x ≡ 1 (mod 60)

Используя обратный элемент d^-1, мы можем найти показатель шифрования e:

e ≡ d^-1 (mod 60)

Для этого нужно решить уравнение:

47 \* x ≡ 1 (mod 60)

Решая это уравнение, получим x = 23, и значит e = 23.

Теперь пользователь А может зашифровать сообщение М, используя открытый ключ пользователя В. Для этого он преобразует сообщение М в число m, соответствующее выбранной кодировке. Затем он вычисляет шифротекст c по формуле:

c ≡ m^e (mod n)

Подставляя значения для e и n, получаем:

c ≡ 17^23 (mod 77)

Для вычисления c можно воспользоваться методом быстрого возведения в степень. Этот метод позволяет вычислить m^e по модулю n за O(log e) операций умножения. Применяя его к числу 17 и показателю шифрования e = 23, получаем:

17^23 = ((17^2)^2)^2 \* 17^2 \* 17^2 \* 17

17^2 = 289, а по модулю 77 это равно 20.

Тогда мы можем продолжить последовательность:

17^4 = (17^2)^2 = 400, а по модулю 77 это равно 38.

17^8 = (17^4)^2 = 1444, а по модулю 77 это равно 29.

Таким образом:

17^23 = (17^8)^2 \* 17^7 = 29^2 \* 17^7 = 841 \* 5764801 = 1399475241

И по модулю 77:

1399475241 ≡ 24 (mod 77)

Таким образом, шифрованный текст c, который пользователь А отправит пользователю В, будет равен 24.

**6. Пользователю системы RSA с ключевыми параметрами *n* = 33, *d* = 3 передано зашифрованное сообщение С, состоящее из блока цифр: 13. Расшифровать это сообщение (взломав систему RSA пользователя).**

*d* \* *e* ≡ 1 (mod (*n*)).

1) Находим закрытый ключ *e*: ищем значение функции Эйлера от *n*.

n = 33 => функция Эйлера от *n* будет равна φ(33) = (11 - 1) \* (3 - 1) = 20.

2) Находим такое значение *e*, что *d* \* *e* ≡ 1 (mod (20)).

Используя расширенный алгоритм Евклида, можно найти решение этого уравнения: d \* e + k \* 20 = 1, где *k* - целое число.

3 \* 7 + (-4) \* 20 = 1 => e = 7

Теперь для расшифровки сообщения С необходимо возвести его в степень *d*, по модулю *n*:

13^3 mod 33 = 13

Ответ: Расшифрованный текст равен 13.

**7. В системе связи, применяющей шифр Эль-Гамаля, пользователь А желает передать сообщение *М* пользователю В. Найти недостающие параметры системы при следующих заданных параметрах: *p* = 19, *g* = 2, *х* = 3, *k* = 5, *М* = 10. Описать по шагам зашифрование сообщения и расшифрование шифртекста.**

1. Вычисляем открытый ключ *y* пользователя А:
2. Генерируем случайное число *k*, взаимно простое с *p*-1: *k* = 5
3. Вычисляем первую составляющую шифртекста:
4. Вычисляем вторую составляющую шифртекста:
5. Пользователь А передает пользователю В пару (*a*, *b*).

Пользователь В выполняет следующие шаги для расшифрования полученного шифртекста:

1. Вычисляем значение общего секретного ключа *s*:
2. Вычисляем мультипликативно обратное значение *y*:
3. Вычисляем исходное сообщение *М*:

Ответ: Полученное исходное сообщение *М* равно 10.