1. Министерство образования и науки Российской Федерации
2. Санкт-Петербургский Политехнический Университет Петра Великого
3. —
4. Институт кибербезопасности и защиты информации
5. **Кафедра «Информационная безопасность компьютерных систем»**

**ЛАБОРАТОРНАЯ РАБОТА № 1**

1. «**Математические примитивы криптографии**»
2. по дисциплине «Основы информационной безопасности»
3. Выполнили
4. студенты гр. 5131001/40003 Веденеев А. С.
5. <*подпись*>  
     
    Поляков Д. Т.
   1. <*подпись*>
6. Преподаватель
7. ст. преподаватель Вагисаров В. Б.

<*подпись*>

1. Санкт-Петербург
2. 2025

# Цель и задачи

**Цель:** Приобретение расчетных навыков в модульной арифметике, используемой в криптографических алгоритмах и протоколах, ознакомление с математическим вычислениями, используемыми для сокрытия сообщений на примерах алгоритма шифрования RSA и ранцевой криптосистемы Меркля-Хеллмана.

**Задачи:**

1. Вычислить число по формуле .
2. Зашифровать строку, составленную из фамилии, имени и отчества, записанных кириллицей.
3. Вычислить число . Рассчитать число B = ЧЧММГГГГ дня рождения. Найти НОД(A, B(mod 95) + 900), НОД(A, (B + 50)(mod 97) + 700), НОД(A, (B + 20)(mod101) + 1500, (B - 40)(mod 103) + 2500).
4. Методом Миллера доказать, что выбранное число N составное. Выбрать простое число N1, 101<N1<1000 и методом Миллера показать, что оно простое.
5. Для изучения генерации ключей в алгоритме RSA выбрать два любых больших (102<N<106) простых числа p и q: НОД(p, q) = 1. Вычислить число n = p\*q. Вычислить порядок группы φ(n). Выбрать показатель шифрования e такой, что НОД(e, p - 1) = НОД(e, q - 1) = 1. Вычислить показатель d по формуле ed = 1(mod φ(n)).
6. Выбрать и зашифровать произвольный текст x. Затем расшифровать текст с помощью ключа d. Сравнить расшифрованный текст с исходным.
7. Подписать текст x цифровой подписью s. Проверить подпись.
8. Выбрать любое число a. Смоделировать действия пользователя A. Смоделировать действия пользователя B. Установить сеансовый ключ. Проверить равенство полученных ключей пользователя.
9. Разработать утилиту шифрования и дешифрования с помощью алгоритма Меркля-Хеллмана.

# ход работы

## Определение параметра K.

Был совершен расчет по формуле ,

где – номер учебной группы, – порядковый номер в списке группы, – Порядковый номер в алфавите третьей буквы фамилии (буква “Д”).

= 7.

## Шифрование строки с использованием шифра Цезаря.

В качестве значения k было выбрано 7. Шифрование осуществлялось в программе на языке C с использованием кодировки ASCII. Каждая буква заменялась символом, расположенным на (x+k)-й позиции в русском алфавите, где x — порядковый номер исходного символа. Если полученное значение превышало количество букв в алфавите, отсчёт продолжался с начала. В процессе разработки возникла сложность, связанная с буквой "ё", так как в таблице она занимает отдельное место. Этот нюанс был учтён, и программа корректно обрабатывает весь русский алфавит.

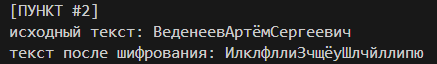


Рисунок 1 – Шифрование строки: “ФамилияИмяОтчество” с помощью алгоритма Цезаря

## Вычисление A и НОД (A, B).

Исходные данные:

* **Номер группы** (Nгр​): 40003
* **Порядковый номер в списке** (Nсп): 6
* **B**: 3042006 (дата рождения в формате ДДММГГГГ — 03.04.2006).

1. Вычисление A: A = (Nгр \* (8 + Nсп (mod 7))) ^ 2

Подставляем значения: A = (40003 \* (8 + (6 mod 7))) ^ 2 = 51581124

Результат: A = 313647041764

1. Поиск НОД (A, B)

Последовательность вычислений для НОД1:

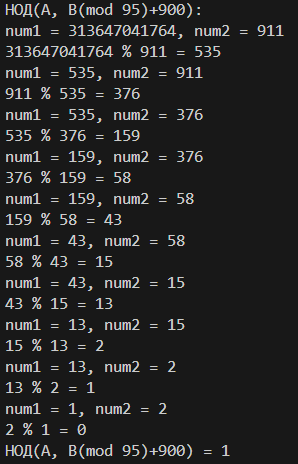


Рисунок 2 – Расчет НОД1 с помощью метода Евклида

Последовательность вычислений для НОД2:

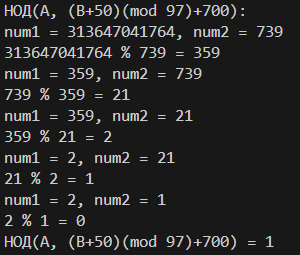


Рисунок 3 – Расчет НОД2 с помощью метода Евклида

Последовательность вычислений для НОД3:

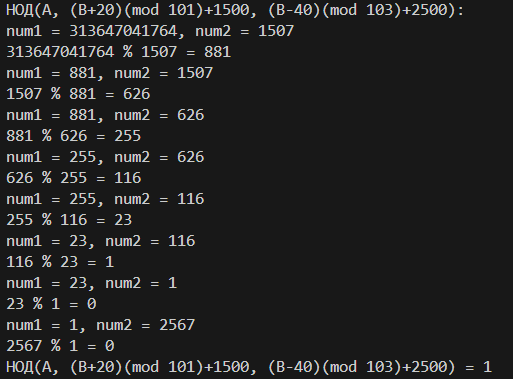


Рисунок 4 – Расчет НОД3 с помощью метода Евклида

С помощью программы на языке C методом Евклида были найдены НОД, приведённые на рисунках: 2, 3, 4. Там же можно найти и результаты вычислений. НОД(A, (B+20)(mod 101)+1500, (B-40)(mod 103)+2500) было вычислено следующим способом: сначала было вычислено НОД(A, (B+20)(mod 101)+1500), а затем НОД(НОД(A, (B+20)(mod 101)+1500), (B-40)(mod 103)+2500). Метод Евклида заключается в следующем: даны два числа, большее из них берётся по модулю меньшего, большее число заменяется полученным в результате данной операции. Это повторяется до тех пор, пока одно из чисел не будет равно 0. Как только одно из чисел станет равным 0, другое число и будет НОД.

## Проверка составного и простого числа

Было принято, что число N = 16784, а число N1 = 601. Методом Миллера было доказано, что N – составное число, а N1 – простое число с вероятностью, практически равной 100%. Вероятность того, что N1 будет определено как составное число равна (¼)k = (¼)50 = 7, 889 \* 10-31. Метод Миллера был реализован функцией в программе на языке C. На её вход поступает число, которое и требуется проверить на простоту. Код данной функции приведен в приложении.

Сначала происходит проверка на то, является ли число N=2 или N=3, так как для таких маленьких простых чисел алгоритм не работает. Если число не равно 2, 3, не меньше 2 и не делится нацело на 2, то происходит проверка на простоту методом Миллера. Для заданного N находятся такие целое число s и целое нечётное число t, что . Выбирается случайное число a, 1 < a < N-1. Если a не является свидетелем простоты числа N(то есть не выполняются следующие условия: 1. N не делится на a; 2. или существует целое r, такое что , то число N является составным, и происходит выход из функции. Иначе выбирается новое случайное число a и процедура проверки повторяется. После нахождения k свидетелей простоты можно прийти к выводу, что вероятнее всего N – простое число. В случае, если число является простым, функция возвращает 1, если составным – 0. Для простого числа при k = 50 количество шагов примерно равно 2 + 2\*s + 50\*(4 + 2\*(t-1) + 5\*(s-1)). Если искомое число является составным, то оно определится за гораздо меньшее число шагов.

## Генерация открытого и закрытого ключей RSA с использованием алгоритма Евклида

1. Выбор простых чисел p и q.

Пусть p=587 и q=743. Эти числа простые, и их НОД равен 1.

1. Вычисление n.

n = p \*q = 587 \* 743 = 436141

1. Вычисление функции Эйлера φ(n).

= 586 \* 742 = 434812

1. Выбор числа e.

Пусть e = 29, так как: gcd(29, 586) = 1, gcd(29, 742) = 1

(Это означает, что 29 не делится на 586 и 742)

1. Вычисление d.

Нам нужно найти такое d, что: 29 \* d ≡ 1 (mod 434812)

Сначала находим gcd(29, 434812) с вычислением коэффициентов:

Расширенный алгоритм Евклида (нахождение коэффициентов x и y в уравнении ax + by = gcd(a, b))

1. Разложение:

434812 : 29 = 14993, 434812 – 14993 \* 29 = 15

29 : 15 = 1, 29 – 1 \* 15 = 14

15 : 14 = 1, 15 – 1 \* 14 = 1

14 : 1 = 14, 14 – 14 \* 1 = 0

Следовательно: gcd(434812, 29) = 1

1. Обратный ход (выражаем НОД в виде линейной комбинации)

* 1 = 15 – 1 \* 14
* 14 = 29 – 1 \* 15, подставляем:
  + 1 = 15 – 1 \* (29 – 1 \* 15)
  + 1 = 2 \* 15 – 1 \* 29
* 15 = 434812 – 14993 \* 29, подставляем:
  + 1 = (434812 - 14993 \* 29) – 1 \* 29
  + 1 = 434812 – 14994 \* 29

— Значит: d = −14994 ≡ 404825 (mod 434812)

\*альтернатива данному решению является перебор, который реализован в коде на языке Си

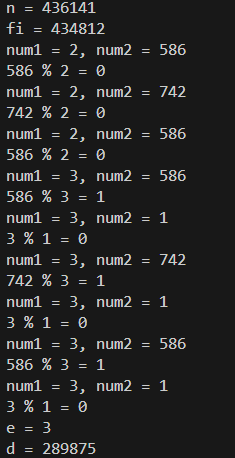


Рисунок 5 – Программное вычисление с другими данными.

Так как d должно быть положительным, прибавляем модуль:

d = 434812 – 14994 = 404825

Итого получаем:

Открытый ключ (e, n) = (29, 436141)

Закрытый ключ (404825, 436141)

## Реализация RSA-шифрования в ASCII

Исходные данные:

p = 587, q = 743, e = 29, text = “RSA test”, n = 436141, d = 404825

ASCII Таблица:

|  |  |
| --- | --- |
| R | 82 |
| S | 83 |
| A | 65 |
| (space) | 32 |
| t | 116 |
| e | 101 |
| s | 115 |
| t | 116 |

Шифрование: (x^e)(mod n)

R = (82^29)(mod 27685) = 333773

S = (83^29)(mod 27685) = 295059

A = (65^29)(mod 27685) = 22248

(space) = (32^29)(mod 27685) = 163678

t = (116^29)(mod 27685) = 205879

e = (101^29)(mod 27685) = 310128

s = (115^29)(mod 27685) = 120674

Расшифрование: (y^d)(mod n)

R = (333773^404825)(mod 436141) = 82

S = (295059^404825)(mod 436141) = 83

A = (22248^404825)(mod 436141) = 65

(space) = (163678^404825)(mod 436141) = 32

t = (205879^404825)(mod 436141) = 116

e = (310128^404825)(mod 436141) = 101

s = (120674^404825)(mod 436141) = 115

Используя данные из кода:

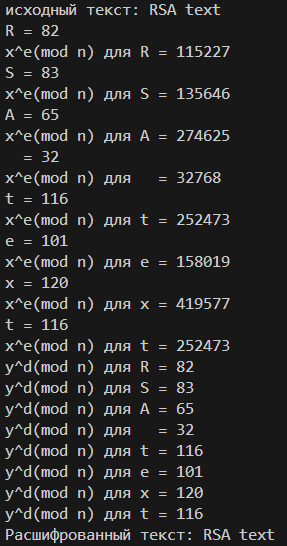


Рисунок 6 – Шифрование и дешифрование текста программно.

Зашифровка и расшифровка были реализованы программой на языке C. На рисунке 6 представлены зашифровка и расшифровка сочетания слов “RSA text”. Сначала была произведена зашифровка. Для этого вычислен тем самым был получен зашифрованный тест deciphered. Далее этот текст был расшифрован с помощью ключа d по формуле Расшифрованный текст оказался полностью идентичен исходному.

## Цифровая подпись и проверка с помощью RSA

Исходные данные:

p = 587, q = 743, e = 29, text = “RSA test”, n = 436141, d = 404825

Подпись текста: s = (x^d)(mod n)

R = (82^404825)(mod 436141) = 15815

S = (83^404825)(mod 436141) = 368221

A = (65^404825)(mod 436141) = 309125

(space) = (32^404825)(mod 436141) = 216459

t = (116^404825)(mod 436141) = 143509

e = (101^404825)(mod 436141) = 433244

s = (115^404825)(mod 436141) = 75414

Проверка подписи: (s^e)(mod n)

R = (15815^29)(mod 436141) = 82

S = (368221^29)(mod 436141) = 83

A = (309125^29)(mod 436141) = 65

(space) = (216459^29)(mod 436141) = 32

t = (143509^29)(mod 436141) = 116

e = (433244^29)(mod 436141) = 101

s = (75414^29)(mod 436141) = 115

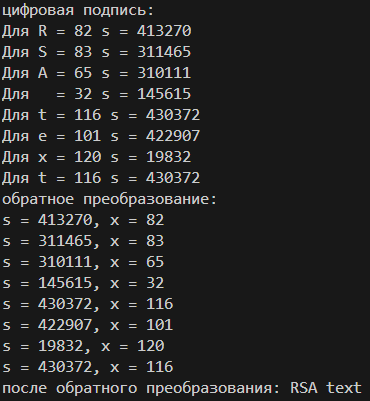
Итог: цифровая подпись смоделирована верно. 

Рисунок 7 – реализация цифровой подписи на C

Для цифровой подписи и обратного преобразования также был написан код на языке C. На рисунке 4 представлен пример цифровой подписи и её обратного преобразования для слова “алгоритм”. Сначала была произведена подпись текста x по формуле , где s – сама цифровая подпись. Затем было совершено обратное преобразование по формуле В итоге получили текст, полностью идентичный исходному.

## Моделирование установления сеансового ключа по схеме Диффи-Хеллмана.

Начальные данные: a = 5, n = 436141

1. Пользователь A:

X = 39 (x < f(n))

A = a^x (mod n) = 5^39 (mod 436141) = 326937

1. Пользователь B:

y = 298 (y < f(n))

B = a^y (mod n) = 5^298 (mod 436141) = 132472

1. Сеансовые ключи:

* A: B^x mod n = 260987
* B: A^y mod n = 260987
* B^x = A^y = a^xy (mod n) = 260987

Сеансовый ключ вычислен правильно.

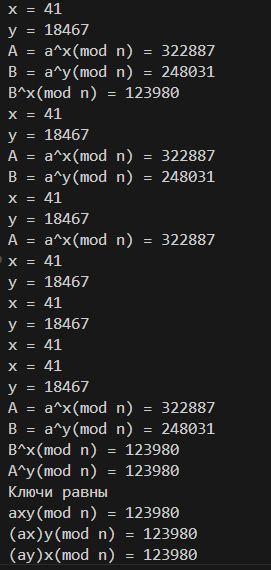


Рисунок 8 – реализация сеансового ключа

Для моделирования процесса установления сеансового ключа в программе было выбрано число a = 5. На языке C было смоделировано действие пользователя A: случайным образом было выбрано число найдено число Далее было смоделировано действие пользователя B: случайным образом было выбрано число , найдено число Затем пользователь A вычислил , а пользователь B вычислил . Полученные значения образовали сеансовый ключ. На рисунке 7 представлены все полученные значения. Ключи пользователей оказались равны, а также

## Реализация утилиты шифрования и дешифрования с помощью алгоритма Меркля-Хеллмана.

\**Приложение*

В данной программе реализован алгоритм шифрования и дешифрования на основе криптосистемы Меркля-Хеллмана. Алгоритм основан на использовании сверхвозрастающей последовательности и модулярной арифметики.

1. Генерация закрытого ключа

Закрытый ключ состоит из сверхвозрастающей последовательности чисел w, модуля n и числа e, взаимно простого с n:

* Сначала формируется массив w, где каждый последующий элемент больше суммы всех предыдущих:
* Затем выбирается число n, большее суммы всех элементов последовательности w.
* Подбирается число e, взаимно простое с n, используя алгоритм Евклида для проверки gcd(e, n) = 1.

1. Генерация открытого ключа

На основе закрытого ключа вычисляется открытый ключ β по формуле:

Где последовательность β и число n составляют открытый ключ.

1. Шифрованние сообщения

* Сообщение представляется в виде двоичной последовательности длины *KEY\_SIZE*
* Вычисляется сумма:
* Полученное число *с* передается в зашифрованном виде.

1. Дешифрование сообщения

* Вычисляется s - восстановленное значение суммы элементов закрытого ключа:

Где *d* – обратный элемент к *e* по модулю *n*, вычисленный через расширенный алгоритм Евклида.

* Определяется исходное сообщение, путем разложения s по элементам сверхвозрастающей последовательности w. Если очередной элемент w[i] вычитается из s, то бит сообщения устанавливается в 1, иначе — в 0.
* В результате получается исходное двоичное сообщение.

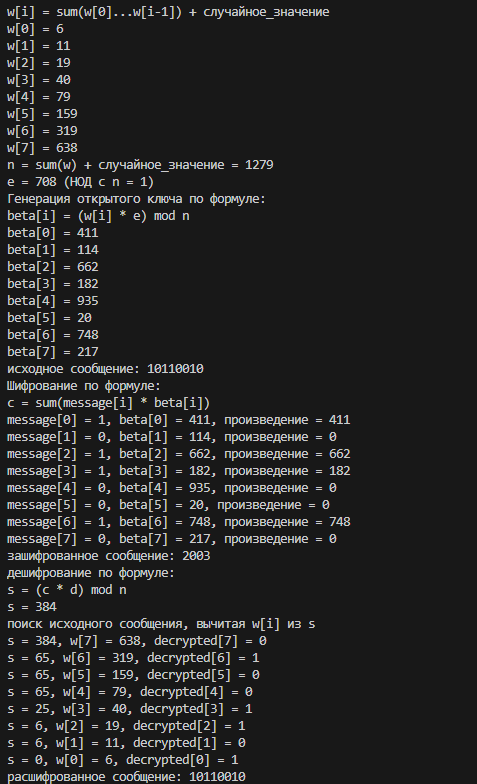


Рисунок 9 – результаты работы утилиты шифрования и дешифрования с помощью алгоритма Меркля-Хеллмана.

## Ответы на контрольные вопросы

1. Что такое вычет? На чем основан алгоритм шифрования Цезаря?

Вычет – это число, которое является остатком от деления одного числа на другое. Записывается как: a ≡ r (mod b) и означает, что число a при делении на b дает остаток r.

Алгоритм шифрования Цезаря - то простой метод подстановки, в котором каждая буква исходного текста заменяется буквой, находящейся на k позиций вперед в алфавите. Например, при k=3:

A -> D, B -> E, C -> F.

1. Каковы особенности чисел Кармайкла?

Число Кармайкла – это составное число N, которое удовлетворяет:

a^(N−1) ≡ 1 (mod N), для всех a, GCD (a, N) = 1.

Это делает их "псевдопростыми" по малой теореме Ферма, что затрудняет проверку их составности. Пример самого малого числа Кармайкла: 561.

1. Основные свойства мультипликативной группы кольца вычетов по модулю pqpq:
   1. Коммутативность: Умножение элементов в мультипликативной группе кольца вычетов по модулю pq является коммутативным, то есть для любых элементов a, b из группы выполняется свойство ab = ba.
   2. Ассоциативность: Умножение элементов в группе ассоциативно, то есть для любых элементов a, b, c из группы выполняется свойство a(bc) = (ab)c.
   3. Существование единицы: В мультипликативной группе кольца вычетов по модулю pq существует единичный элемент, обозначаемый как 1, который удовлетворяет условию 1a = a1 = a для любого элемента a.
   4. Существование обратного элемента: Для каждого элемента a из мультипликативной группы существует обратный элемент a^(-1), такой что a a^(-1) = a^(-1) a = 1.
   5. Замкнутость: Результат умножения двух элементов из мультипликативной группы также является элементом этой же группы.
   6. Порядок группы: Порядок мультипликативной группы кольца вычетов по модулю pq равен φ(pq) = (p-1)(q-1), где φ - функция Эйлера. Порядок мультипликативной группы кольца вычетов по модулю n обозначает количество элементов в этой группе
2. Почему порядок группы (Zn​/Z)∗ должен иметь большой простой делитель?
   1. Безопасность RSA: В криптографическом алгоритме RSA безопасность основана на сложности задачи факторизации больших составных чисел n, состоящих из двух больших простых множителей p и q. Если порядок группы мультипликативной группы (Z/nZ) имеет большой простой делитель, то это осложняет атаки, основанные на разложении n на простые множители.Большой простой делитель затрудняет разложение числа n на p и q*.*
   2. Усиление модулярной арифметики: Большой простой делитель порядка группы (Z/nZ) способствует увеличению сложности вычислений в модулярной арифметике, что делает криптографические методы более стойкими к различным типам атак.
   3. Уменьшение вероятности коллизий и атак: Большой простой делитель порядка группы (Z/nZ) повышает вероятность уникальности получаемых ключей шифрования и уменьшает возможность успешных атак, основанных на алгоритмах разделения и анализа группы элементов.
3. Алгоритм расчета кодов символов при декодировании шифрограмм по алгоритму Меркля-Хеллмана:
4. Генерация ключей:

* Закрытый ключ создается как сверхвозрастающая последовательность ww длиной 8.
* Вычисляется модуль nn, который больше суммы всех элементов ww.
* Выбирается число ee, взаимно простое с nn.
* Открытый ключ формируется по формуле:

Βi = (wi \* e) mod  n

1. Шифрование:

* Сообщение представляется в виде двоичной строки длиной 8.
* Для каждого i, если бит сообщения message[i] = 1, то суммируется соответствующее значение из открытого ключа:

c = ∑ message[i] \* β[i]

* Полученное зашифрованное число cc передается для расшифрования.

1. Дешифрование:

* Вычисляется обратный элемент d, такой что:

(e \* d) mod n = 1

* Дешифрованное значение s получается, как:

s = (c \* d) mod n

* Далее, используя жадный алгоритм, из s поэтапно вычитаются элементы ww, начиная с наибольшего.
* Если s больше или равен w[i], то устанавливается decrypted[i] = 1, иначе decrypted[i] = 0
* В результате получается исходное двоичное сообщение.

Этот алгоритм основан на том, что разложение числа ss по сверхвозрастающей последовательности однозначно определяет исходное сообщение.

# Выводы

В ходе лабораторной работы были изучены основы криптографических методов, включая принципы модульной арифметики и ее применение в криптографических алгоритмах. Были рассмотрены простейшие способы шифрования, включая шифр Цезаря, а также реализованы и протестированы более сложные криптосистемы, такие как алгоритм RSA и ранцевая криптосистема Меркля-Хеллмана. Практическая реализация позволила закрепить знания о шифровании и расшифровании сообщений, генерации ключей и свойствах криптографических систем.

# Приложения