

Лабораторная работа № 6
«Покер по телефону (2 игрока)»
Отчет по лабораторной работе:

4.8. Реализуйте криптографический протокол «игры в покер по телефону» для двух игроков на основе коммутирующих криптосистем.

Теория. Коммутирующие криптосистемы можно построить по схеме RSA над общим Z_n , так как $(e^{t_1})^{t_2} = (e^{t_2})^{t_1} \pmod{n}$. Описание криптопротокола:

1. Алиса и Боб создают пары «открытый ключ /закрытый ключ»:
Алиса - K_A^{open} / K_A^{close} , Боб - K_B^{open} / K_B^{close} .
2. Алиса зашифровывает своим открытым ключом все 52 карты в колоде, перемешивает и результат отправляет Бобу.
3. Боб случайным образом выбирает из них 5 шифрограмм для Алисы и отправляет их ей.
4. Алиса с помощью своего закрытого ключа расшифровывает свои карты.
5. Боб случайным образом из оставшихся шифрограмм выбирает 5 для себя, зашифровывает их своим открытым ключом и отправляет Алисе.
6. Алиса расшифровывает их своим закрытым ключом и отправляет Бобу.
7. Боб окончательно расшифровывает свои карты своим закрытым ключом.
8. Если требуется добирать карты из колоды, то поступают аналогичным образом.
9. В конце игры Алиса и Боб раскрывают свои карты и пары ключей для того, чтобы каждый мог убедиться в отсутствии мошенничества.

Rsa:

Шаг первый. Подготовка ключей

Я должен проделать предварительные действия: сгенерировать публичный и приватный ключ.

- Выбираю два простых числа. Пусть это будет $p=3$ и $q=7$.
- Вычисляем *модуль* — произведение наших p и q : $n=p \times q=3 \times 7=21$.
- Вычисляем *функцию Эйлера*: $\phi=(p-1) \times (q-1)=2 \times 6=12$.
- Выбираем число e , отвечающее следующим критериям: (i) оно должно быть простым, (ii) оно должно быть меньше ϕ — остаются варианты: 3, 5, 7, 11, (iii) оно должно быть взаимно простым с ϕ ; остаются варианты 5, 7, 11. Выберем $e=5$. Это, так называемая, *открытая экспонента*.

Теперь пара чисел $\{e, n\}$ — это мой открытый ключ. Я отправляю его вам, чтобы вы зашифровали своё сообщение. Но для меня это ещё не всё. Я должен получить закрытый ключ.

Мне нужно вычислить число d , обратное e по модулю ϕ . То есть остаток от деления по модулю ϕ произведения $d \times e$ должен быть равен 1. Запишем это в обозначениях, принятых во многих языках программирования: $(d \times e) \% \phi = 1$. Или $(d \times 5) \% 12 = 1$. d может быть равно 5 ($(5 \times 5) \% 12 = 25 \% 12 = 1$), но чтобы оно не путалось с e в дальнейшем повествовании, давайте возьмём его равным 17. Можете проверить сами, что $(17 \times 5) \% 12$ действительно равно 1 ($17 \times 5 - 12 \times 7 = 1$). Итак $d=17$. Пара $\{d, n\}$ — это секретный ключ, его я оставляю у себя. Его нельзя сообщать никому. Только обладатель секретного ключа может расшифровать то, что было зашифровано открытым ключом.

Шаг второй. Шифрование

Теперь пришла ваша очередь шифровать ваше сообщение. Допустим, ваше сообщение это число 19. Обозначим его $P=19$. Кроме него у вас уже есть мой открытый ключ: $\{e, n\} = \{5, 21\}$. Шифрование выполняется по следующему алгоритму:

- Возводите ваше сообщение в степень e по модулю n . То есть, вычисляете 19 в степени 5 (2476099) и берёте остаток от деления на 21. Получается 10 — это ваши закодированные данные.

Строго говоря, вам вовсе незначает вычислять огромное число «19 в степени 5». При каждом умножении достаточно вычислять не полное произведение, а только остаток от деления на 21. Но это уже детали реализации вычислений, давайте не будем в них углубляться.

Полученные данные $E=10$, вы отправляете мне.

Здесь надо заметить, что сообщение $P=19$ не должно быть больше $n=21$. Иначе ничего не получится.

Шаг третий. Расшифровка

Я получил ваши данные ($E=10$), и у меня имеется закрытый ключ $\{d, n\} = \{17, 21\}$.

Обратите внимание на то, что открытый ключ не может расшифровать сообщение. А закрытый ключ я никому не говорил. В этом вся прелесть асимметричного шифрования.

Начинаем декодировать:

- Я делаю операцию, очень похожую на вашу, но вместо e использую d . Возвожу E в степень d : получаю 10 в степени 17 (позвольте, я не буду писать единичку с семнадцатью нулями). Вычисляю остаток от деления на 21 и получаю 19 — ваше сообщение.

Заметьте, никто, кроме меня (даже вы!) не может расшифровать ваше сообщение ($E=10$), так как ни у кого нет закрытого ключа.

```

public class Main {
    private static ServerSocket ss = null;
    private static Socket socket = null;
    private static int Port = (int)(Math.random()*((65535-1025)+1))+1025; // Порт Стола (1025..65535)
    public static String internal_username = "unknown";
    public static String external_username = "unknown";
    public static Boolean hasConnection = false;
    public static int cards[] = new int[52];
    public static int en_cards[] = new int[52];
    public static int my_cards[] = new int[5];
    public static int my_enc_cards[] = new int[5];
    public static int ex_cards[] = new int[5];
    public static int test_cards[] = new int[5];
    public static int ex_double_enc_cards[] = new int[5];
    public static int score[][] = new int[5][2]; // Счет стола
    public static int count = 0; // Кол-во честных карт оппонента

    public static int P = 0, Q = 0, N = 0, F = 0;
    public static int E = 0, D = 0; // Открытый и Открытый ключ
    public static int exp[] = new int[100]; // Будет массив из открытых экспонент
    public static int rnd = 3; // Место начала поисков открытых экспонент
    public static int rand = 0; // Позиция выбранной экспоненты в массиве открытых экспонент
    public static int ex_E = 0, ex_D = 0; // Переменные для проверки честности опонента в конце игры

```

Чтобы создать две коммутирующих криптосистемы RSA над общим полем Z , необходим общий модуль и разные открытые экспоненты и закрытые ключи. При запуске программы игроки представляются, после чего кто-то из них создает стол, а второй к нему подключается, как только установилась связь они обмениваются псевдонимами, и

подключившийся игрок ждет начала сессии. После начала сессии происходит всё согласно описанному протоколу, 2 игрок выбирает для 1 игрока его карты, потом для себя в конце они оба имеют по 5 карт, которые они еще дополнительно перемешивают (для излишней уверенности в том, что никто не знает порядок представления карт) и предъявляют их по очереди. Побеждает тот игрок, чья мощность карты будет больше, если же мощность карт одинакова (например Король и Король), то победитель устанавливается тот, чья масть самая высокая. Мощность карт и масть карт узнаётся с помощью функций:

```
public static String value_of(int x) { // Узнаем название карты
    String v = "";
    if (x % 13 == 12) {
        v = "Ace";
        return v;
    }
    if (x % 13 == 0) {
        v = "2";
        return v;
    } else if (x % 13 == 1) {
        v = "3";
        return v;
    } else if (x % 13 == 2) {
        v = "4";
        return v;
    } else if (x % 13 == 3) {
        v = "5";
        return v;
    } else if (x % 13 == 4) {
        v = "6";
        return v;
    } else if (x % 13 == 5) {
        v = "7";
        return v;
    } else if (x % 13 == 6) {
        v = "8";
        return v;
    } else if (x % 13 == 7) {
        v = "9";
        return v;
    } else if (x % 13 == 8) {
        v = "10";
```

```
public static String suit_of(int card){ // Узнаём масть карты
    String s="";
    if(card/13==0)
    {
        s="Spade";
        return s;
    }
    else if(card/13==1)
    {
        s="Heart";
        return s;
    }
    else if(card/13==2)
    {
        s="Club";
        return s;
    }
    else if(card/13==3)
    {
        s="Diamond";
        return s;
    }
    else{
        s="Spade";
        return s;
    }
}
```


Сама игра:

```
public static int[] game(int a, int b){
    String suit_a = suit_of(a);
    String suit_b = suit_of(b); // Если карты по мощности равны определяем победителя по старшинству мастей в покере:
    int pow_a = 0;
    int pow_b = 0;
    if(suit_a.equals("Spade")){pow_a = 4;} // Если игрок А прислал самую старшую масть = выиграл А
    if(suit_a.equals("Heart")){pow_a = 3;}
    if(suit_a.equals("Diamond")){pow_a = 2;}
    if(suit_a.equals("Club")){pow_a = 1;} // Если игрок А прислал самую младшую масть = выиграл Б

    if(suit_b.equals("Spade")){pow_b = 4;}
    if(suit_b.equals("Heart")){pow_b = 3;}
    if(suit_b.equals("Diamond")){pow_b = 2;}
    if(suit_b.equals("Club")){pow_b = 1;}

    int temp[] = new int[2];

    if(a>b){temp[0]=1; return temp;}
    if(a<b){temp[1]=1; return temp;}
    else{
        if(pow_a > pow_b){temp[0]=1; return temp;}
        if(pow_a < pow_b){temp[1]=1; return temp;} // Определяем победителя по мастям в случае если мощность карт одинакова
    }
    return temp; // Если возвращает это 0 0, то ошибка в подборе карт
}
```

от большего к меньшему
Spade, Heart, Diamond, Club

После определения победителя они узнают закрытые и открытые ключи друг друга и проверяют действительно ли не было подстановки карт у оппонента.

```
public static void main(String[] args) throws IOException {
    for(int n = 1; n < 53 ; n++){ // Наполняем массив картами
        cards[n-1] = n;
    }
    //for(int n = 0; n < 52 ; n++){System.out.println(cards[n]);} // Посмотреть массив карт
    InputStream sin = null;
    OutputStream sout = null;
    DataInputStream in = null;
    DataOutputStream out = null;
    System.out.println("\n\u001b[37m" + "Вариант 8. «Покер по телефону (2 игрока)»");
    System.out.println("\u001b[0m" + "Порт текущего стола: " + "\u001b[33m" + Port + "\u001b[0m");
    System.out.println("Введите ваш псевдоним:");
    Scanner sc = new Scanner(System.in);
    internal_username = sc.nextLine();
}
```



```

case 1:
    if(hasConnection == false){System.out.println("За столом нет 2 игрока"); System.exit( status: 4);}
    P = generationLargeNumber();
    Q = generationLargeNumber();
    N = P*Q;
    F = (P-1)*(Q-1);
    System.out.println("P = " + P + "\nQ = " + Q + "\nN = " + N + "\nφ = " + F);
    out.writeInt(P);
    out.writeInt(Q);

    for(int i = 0; i < 100;) // Находим 100 различных открытых экспонент
    {
        if((gcd(rnd, F) == 1) && (rnd < F)) // Проверка на Взаимно простое с φ, открытая экспонента < φ
        {
            exp[i] = rnd;
            i++;
        }
        rnd++;
    }
    //for(int i = 0; i < 100; i++){System.out.println(exp[i]);} // Вывести список экспонент
    // Закрытый ключ: modInverse(rnd, FuncEuler) % FuncEuler
    rand = (int)(Math.random()*(99+1))+0; // Случайным образом выбираем экспоненту из массива открытых экспонент [100]
    E = exp[rand];
    D = (int)modInverse(E, F) % F;
    System.out.println("Открытый ключ [E,N]: " + E + ", " + N + "\nЗакрытый ключ [D,N]: " + D + ", " + N);
    en_cards = enc_deck(cards, E, N); // Шифруем карты открытым ключом Алисы
    //for(int n = 0; n < en_cards.length; n++){System.out.println("Шифрованная карта [\"+n+\"]: " + en_cards[n]);}
    en_cards = shuffle(en_cards); // Перемешиваем карты

```

```

for(int n = 0; n < en_cards.length; n++){
    out.writeInt(en_cards[n]); // Отправляем перемешанные карты Бобу
}

for(int i = 0; i < my_enc_cards.length; i++){
    my_enc_cards[i] = in.readInt(); // Приняли 5 шифрограмм от Боба
}

for(int i = 0; i < 5; i++){System.out.println("Пришло: " + my_enc_cards[i]);}
System.out.println("Расшифровываем...\u001b[34m");
for(int i = 0; i < 5; i++){
    my_cards[i] = (int)power2((long)my_enc_cards[i], (long)D, (long)N);
}

for(int i = 0; i < 5; i++){
    System.out.println("Alice's card [" + (i+1) + "]: " + my_cards[i] + " (" + suit_of(my_cards[i]) + " " + value_of(my_cards[i]) + ")");
}

System.out.print("\u001b[0m");
// Шаг 4 закончен.

ex_double_enc_cards[0] = in.readInt();
ex_double_enc_cards[1] = in.readInt();
ex_double_enc_cards[2] = in.readInt();
ex_double_enc_cards[3] = in.readInt();
ex_double_enc_cards[4] = in.readInt(); // Приняли карты Боба в двойном шифровании

// Снимаем свой шифр и отправляем назад
out.writeInt((int)power2((long)ex_double_enc_cards[0], (long)D, (long)N));
out.writeInt((int)power2((long)ex_double_enc_cards[1], (long)D, (long)N));
out.writeInt((int)power2((long)ex_double_enc_cards[2], (long)D, (long)N));
out.writeInt((int)power2((long)ex_double_enc_cards[3], (long)D, (long)N));
out.writeInt((int)power2((long)ex_double_enc_cards[4], (long)D, (long)N));

```

```

shuffle(my_cards); // Дополнительно перемешиваем свои карты
for(int i = 0; i<my_cards.length; i++){ // Играем
    int temp[] = new int[2];
    System.out.println("\u001b[33mAlice plays: " + my_cards[i] + " (" + suit_of(my_cards[i]) + " " + value_of(my_cards[i]) + ")");
    out.writeInt(my_cards[i]);
    ex_cards[i] = in.readInt();
    System.out.println("Bob plays: " + ex_cards[i] + " (" + suit_of(ex_cards[i]) + " " + value_of(ex_cards[i]) + ")\n\u001b[0m");
    temp = game(my_cards[i], ex_cards[i]);
    score[i][0] = temp[0];
    score[i][1] = temp[1];
}
System.out.println("Счёт текущей партии: \nAlice: \u001b[33m" + (score[0][0]+score[1][0]+score[2][0]+score[3][0]+score[4][0]));
System.out.println("\u001b[0mBob: \u001b[33m" + (score[0][1]+score[1][1]+score[2][1]+score[3][1]+score[4][1]) + "\u001b[0m");

out.writeInt(E); // Обмен открытым и закрытым ключом в конце игры
out.writeInt(D);
ex_E = in.readInt();
ex_D = in.readInt();

test_cards[0] = (int)power2(((int)power2((long)ex_double_enc_cards[0], (long)D, (long)N)), ex_D, N);
test_cards[1] = (int)power2(((int)power2((long)ex_double_enc_cards[1], (long)D, (long)N)), ex_D, N);
test_cards[2] = (int)power2(((int)power2((long)ex_double_enc_cards[2], (long)D, (long)N)), ex_D, N);
test_cards[3] = (int)power2(((int)power2((long)ex_double_enc_cards[3], (long)D, (long)N)), ex_D, N);
test_cards[4] = (int)power2(((int)power2((long)ex_double_enc_cards[4], (long)D, (long)N)), ex_D, N);

count = 0;
for(int i = 0; i<test_cards.length; i++){
    for(int j = 0; j<ex_cards.length; j++){
        if(test_cards[i]==ex_cards[j]){count++;}
    }
}
}

```

Второй игрок:

```
case 3:
    try {
        while(true){
            System.out.println("Введите порт стола (1025..65535): ");
            Scanner sp = new Scanner(System.in);
            if(sp.hasNextInt()){Port = sp.nextInt();}else{System.out.println("Попробуйте ещё раз!");}
            break;
        }
        if(Port < 1025 || Port > 65535){System.exit( status: 4);}
        System.out.println("Подключаемся к столу: " + Port);
        socket = new Socket( host: "127.0.0.1", Port); // Подключаемся к столу
        sin = socket.getInputStream(); // Конвертируем потоки в другой тип, чтоб легче обрабатывать текстовые сообщения.
        sout = socket.getOutputStream();
        in = new DataInputStream(sin);
        out = new DataOutputStream(sout);
        System.out.println("Подключение успешно!");
        out.writeUTF(internal_username);
        external_username = in.readUTF();
        System.out.println("Оппонент: " + "\u001b[33m" + external_username + "\u001b[0m");
        System.out.println("Ожидание начала игры...");
        P = in.readInt();
        Q = in.readInt();
        N = P*Q;
        F = (P-1)*(Q-1);
        System.out.println("P = " + P + "\nQ = " + Q + "\nN = " + N + "\nФ = " + F);
        hasConnection = true;
```



```

for(int i = 0; i < 100;) // Находим 100 различных открытых экспонент
{
    if((gcd(rnd, F) == 1) && (rnd < F)) // Проверка на Взаимно простое с  $\varphi$ , открытая экспонента <  $\varphi$ 
    {
        exp[i] = rnd;
        i++;
    }
    rnd++;
}
//for(int i = 0; i < 100; i++){System.out.println(exp[i]);}
// Закрытый ключ: modInverse(rnd, FuncEuler) % FuncEuler
rand = (int)(Math.random()*(99+1))+0; // Случайным образом выбираем экспоненту из массива открытых экспонент [100]
E = exp[rand];
D = (int)modInverse(E,F) % F;
System.out.println("Открытый ключ [E,N]: " + E + ", " + N + "\nЗакрытый ключ [D,N]: " + D + ", " + N);

for(int n = 0; n < en_cards.length; n++){
    en_cards[n] = in.readInt(); // Принимаем зашифрованные + перетасованные карты от Алисы
}
int temp1 = 0;
for(int i = 0; i < ex_cards.length;){
    do {temp1 = (int)(Math.random()*(51+1))+0;
        //System.out.println("Где-то совпало, temp1: " + temp1);
    }
    while(((ex_cards[0] == temp1)||ex_cards[1] == temp1)||ex_cards[2] == temp1)||ex_cards[3] == temp1)||ex_cards[4] == temp1));
    ex_cards[i]=temp1;
    i++;
}

for (int i = 0; i < ex_cards.length; i++){System.out.println("EX_CARDS[" + i + "]: " + ex_cards[i]);} // Проверяем что напикали

```

```

test_cards[0] = en_cards[ex_cards[0]];
test_cards[1] = en_cards[ex_cards[1]];
test_cards[2] = en_cards[ex_cards[2]];
test_cards[3] = en_cards[ex_cards[3]];
test_cards[4] = en_cards[ex_cards[4]];
out.writeInt(test_cards[0]);
out.writeInt(test_cards[1]);
out.writeInt(test_cards[2]);
out.writeInt(test_cards[3]);
out.writeInt(test_cards[4]); // Отправили 5 шифрограмм Алисы -> Алисе

for(int i = 0; i < 5; i++){System.out.println("Отправили: " + en_cards[ex_cards[i]]);}

// Шаг 5
for(int i = 0; i < my_enc_cards.length;){// проверка на то чтобы не было карт которые отправил Алисе + на повторение
    do {temp1 = (int)(Math.random()*(51+1))+0;
        //System.out.println("Где-то совпало, temp: " + temp);
    }
    while(((ex_cards[0] == temp1)|| (ex_cards[1] == temp1)|| (ex_cards[2] == temp1)|| (ex_cards[3] == temp1)|| (ex_cards[4] == temp1)|| (my_enc_cards[0] == temp1)
    my_enc_cards[i]=temp1;
    i++;
}

for (int i = 0; i < my_enc_cards.length; i++){System.out.println("My_Enc_CARDS[" + i + "]: " + my_enc_cards[i]);} // Проверяем что написали

out.writeInt((int)power2((long)(en_cards[my_enc_cards[0]]),(long)E,(long)N));
out.writeInt((int)power2((long)(en_cards[my_enc_cards[1]]),(long)E,(long)N));
out.writeInt((int)power2((long)(en_cards[my_enc_cards[2]]),(long)E,(long)N));
out.writeInt((int)power2((long)(en_cards[my_enc_cards[3]]),(long)E,(long)N));
out.writeInt((int)power2((long)(en_cards[my_enc_cards[4]]),(long)E,(long)N)); // Отправили 5 шифрограмм Боба -> Алисе

// Шаг 5 закончен.
// Шаг 7

```



```

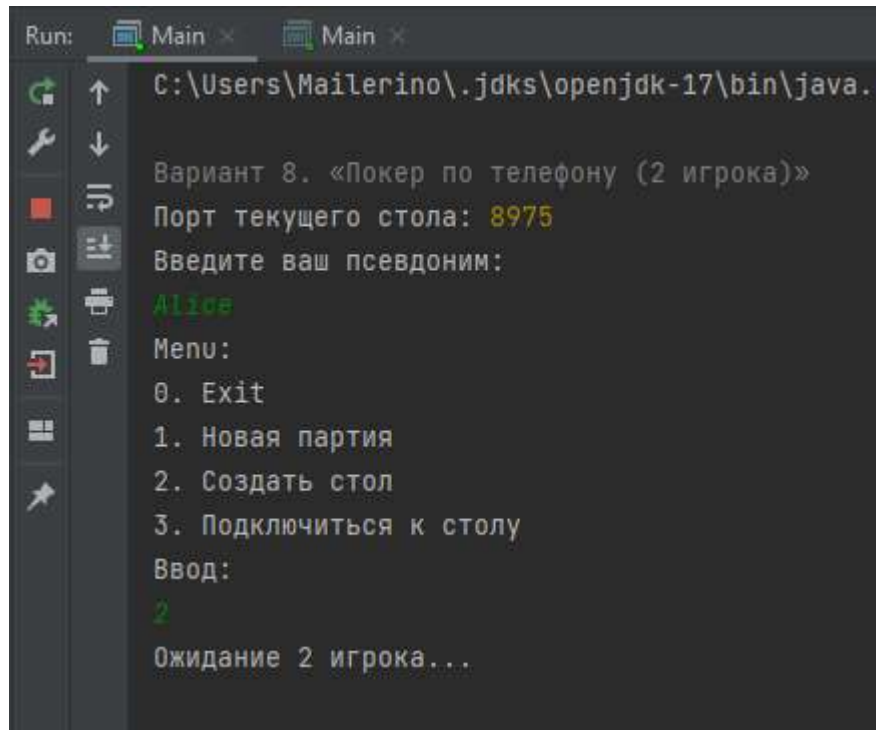
// Шаг 5 закончен.
// Шаг 7
for(int i = 0; i < my_enc_cards.length; i++){
    my_enc_cards[i] = in.readInt(); // Приняли 5 шифрограмм от Алисы
}
for(int i = 0; i < 5; i++){System.out.println("Пришло: " + my_enc_cards[i]);}
System.out.println("Расшифровываем...");
for(int i = 0; i < 5; i++){
    my_cards[i] = (int)power2((long)my_enc_cards[i], (long)D, (long)N);
}
System.out.print("\u001b[34m");
for(int i = 0; i < 5; i++){
    System.out.println("Bob's card [" + (i+1) + "]: " + my_cards[i] + " (" + suit_of(my_cards[i]) + " " + value_of(my_cards[i]) + ")");
}
System.out.print("\u001b[0m");
shuffle(my_cards); // Дополнительно перемешиваем свои карты

for(int i = 0; i < my_cards.length; i++){ // Играем
    int temp[] = new int[2];
    ex_cards[i] = in.readInt();
    System.out.println("\u001b[33mAlice plays: " + ex_cards[i] + " (" + suit_of(ex_cards[i]) + " " + value_of(ex_cards[i]) + ")");
    out.writeInt(my_cards[i]);
    System.out.println("Bob plays: " + my_cards[i] + " (" + suit_of(my_cards[i]) + " " + value_of(my_cards[i]) + ")\n\u001b[0m");
    temp = game(my_cards[i], ex_cards[i]);
    score[i][0] = temp[0];
    score[i][1] = temp[1];
}
System.out.println("Счёт текущей партии: \nAlice: \u001b[33m" + (score[0][1]+score[1][1]+score[2][1]+score[3][1]+score[4][1]));
System.out.println("\u001b[0mBob: \u001b[33m" + (score[0][0]+score[1][0]+score[2][0]+score[3][0]+score[4][0]) + "\u001b[0m");

out.writeInt(E); // Обмен открытым и закрытым ключом в конце игры
out.writeInt(D);

```

Тестирование:



```
Run: Main x Main x
C:\Users\Mailerino\.jdk\openjdk-17\bin\java.
Вариант 8. «Покер по телефону (2 игрока)»
Порт текущего стола: 8975
Введите ваш псевдоним:
Alice
Menu:
0. Exit
1. Новая партия
2. Создать стол
3. Подключиться к столу
Ввод:
2
Ожидание 2 игрока...
```

```
Run: Main x Main x
C:\Users\Mailerino\.jdk\openjdk-17\bin\java.exe "-ja
Вариант 8. «Покер по телефону (2 игрока)»
Порт текущего стола: 53639
Введите ваш псевдоним:
Bob
Menu:
0. Exit
1. Новая партия
2. Создать стол
3. Подключиться к столу
Ввод:
3
Введите порт стола (1025..65535):
8975
Подключаемся к столу: 8975
Подключение успешно!
Оппонент: Alice
Ожидание начала игры...
```

```
Run: Main x Main x
Ожидание 2 игрока...
К столу подключился 2 игрок, его псевдоним: Bob
Menu:
0. Exit
1. Новая партия
2. Создать стол
3. Подключиться к столу
Ввод:
1
P = 5431
Q = 10477
N = 56900587
φ = 56884680
Открытый ключ [E,N]: 329, 56900587
Закрытый ключ [D,N]: 40113209, 56900587
Пришло: 3962412
Пришло: 52281802
Пришло: 31161756
Пришло: 29457801
Пришло: 33391803
Расшифровываем...
Alice's card [1]: 21 (Heart 10)
Alice's card [2]: 6 (Spade 8)
Alice's card [3]: 35 (Club Jack)
Alice's card [4]: 10 (Spade Queen)
Alice's card [5]: 14 (Heart 3)
```

```
← Alice's card [1]: 21 (Heart 10)
Alice's card [2]: 6 (Spade 8)
Alice's card [3]: 35 (Club Jack)
Alice's card [4]: 10 (Spade Queen)
Alice's card [5]: 14 (Heart 3)
Alice plays: 35 (Club Jack)
Bob plays: 23 (Heart Queen)

Alice plays: 21 (Heart 10)
Bob plays: 47 (Diamond 10)

Alice plays: 10 (Spade Queen)
Bob plays: 7 (Spade 9)

Alice plays: 14 (Heart 3)
Bob plays: 34 (Club 10)

Alice plays: 6 (Spade 8)
Bob plays: 44 (Diamond 7)

Счёт текущей партии:
Alice: 2
Bob: 3
Кол-во честных карт Bob: 5
Открытый ключ оппонента: 113
Закрытый ключ оппонента: 2013617
```

2 Игрок:

```
Run: Main x Main x
8975
Подключаемся к столу: 8975
Подключение успешно!
Оппонент: Alice
Ожидание начала игры...
P = 5431
Q = 10477
N = 56900587
φ = 56884680
Открытый ключ [E,N]: 113, 56900587
Закрытый ключ [D,N]: 2013617, 56900587
EX_CARDS[0]: 38
EX_CARDS[1]: 28
EX_CARDS[2]: 49
EX_CARDS[3]: 2
EX_CARDS[4]: 34
Отправили: 3962412
Отправили: 52281802
Отправили: 31161756
Отправили: 29457801
Отправили: 33391803
My_Enc_CARDS[0]: 6
My_Enc_CARDS[1]: 36
My_Enc_CARDS[2]: 1
My_Enc_CARDS[3]: 50
My_Enc_CARDS[4]: 10
Пришло: 38016664
Пришло: 2812837
Пришло: 27592650
Пришло: 9494140
Пришло: 14497218
Расшифровываем...
Bob's card [1]: 47 (Diamond 10)
Bob's card [2]: 44 (Diamond 7)
```



```
Bob's card [1]: 47 (Diamond 10)
Bob's card [2]: 44 (Diamond 7)
Bob's card [3]: 7 (Spade 9)
Bob's card [4]: 34 (Club 10)
Bob's card [5]: 23 (Heart Queen)
Alice plays: 35 (Club Jack)
Bob plays: 23 (Heart Queen)

Alice plays: 21 (Heart 10)
Bob plays: 47 (Diamond 10)

Alice plays: 10 (Spade Queen)
Bob plays: 7 (Spade 9)

Alice plays: 14 (Heart 3)
Bob plays: 34 (Club 10)

Alice plays: 6 (Spade 8)
Bob plays: 44 (Diamond 7)

Счёт текущей партии:
Alice: 2
Bob: 3
Кол-во честных карт Alice: 5
Открытый ключ оппонента: 329
Закрытый ключ оппонента: 40113209
```

Открытые ключи и закрытые ключи совпали, 5/5 честных карт оппонента после проверки.

2 игра:

Вариант 8. «Покер по телефону (2 игрока)»

Порт текущего стола: 44477

Введите ваш псевдоним:

Vector

Menu:

0. Exit

1. Новая партия

2. Создать стол

3. Подключиться к столу

Ввод:

2

Ожидание 2 игрока...

К столу подключился 2 игрок, его псевдоним: Mailer

Menu:

0. Exit

1. Новая партия

2. Создать стол

3. Подключиться к столу

Ввод:

1

P = 11353

Q = 8963

N = 101756939

φ = 101736624

Открытый ключ [E,N]: 23, 101756939

Закрытый ключ [D,N]: 8846663, 101756939

Пришло: 30443880

Пришло: 21047060

Пришло: 54384517

Пришло: 39287215

Пришло: 14569173

Расшифровываем...

Alice's card [1]: 23 (Heart Queen)

Alice's card [2]: 27 (Club 3)

Alice's card [3]: 16 (Heart 5)

Alice's card [4]: 48 (Diamond Jack)

Alice's card [5]: 21 (Heart 10)

Alice plays: 27 (Club 3)

Bob plays: 45 (Diamond 8)

Alice plays: 48 (Diamond Jack)

Bob plays: 13 (Heart 2)

Alice plays: 23 (Heart Queen)

Bob plays: 10 (Spade Queen)

Alice plays: 16 (Heart 5)

Bob plays: 19 (Heart 8)

Alice plays: 21 (Heart 10)

Bob plays: 3 (Spade 5)

Счёт текущей партии:

Alice: 3

Bob: 2

Кол-во честных карт Mailer: 5

Открытый ключ оппонента: 193

Закрытый ключ оппонента: 85922641

Вариант 8. «Покер по телефону (2 игрока)»

Порт текущего стола: 23734

Введите ваш псевдоним:

Mailer

Menu:

0. Exit

1. Новая партия

2. Создать стол

3. Подключиться к столу

Ввод:

3

Введите порт стола (1025..65535):

44477

Подключаемся к столу: 44477

Подключение успешно!

Оппонент: Vector

Ожидание начала игры...

P = 11353

Q = 8963

N = 101756939

φ = 101736624

Открытый ключ [E,N]: 193, 101756939

Закрытый ключ [D,N]: 85922641, 101756939

EX_CARDS[0]: 49

EX_CARDS[1]: 18

EX_CARDS[2]: 42

EX_CARDS[3]: 1

EX_CARDS[4]: 35

Отправили: 30443880

Отправили: 21047060

Отправили: 54384517

Отправили: 39287215

Отправили: 14569173

My_Enc_CARDS[0]: 34

My_Enc_CARDS[1]: 9

My_Enc_CARDS[2]: 4

My_Enc_CARDS[3]: 28

My_Enc_CARDS[4]: 5

Пришло: 19731767

Пришло: 20228210

Пришло: 84927615

Пришло: 33490499

Пришло: 20259089

Расшифровываем...

Bob's card [1]: 45 (Diamond 8)

Bob's card [2]: 13 (Heart 2)

Bob's card [3]: 10 (Spade Queen)

Bob's card [4]: 3 (Spade 5)

Bob's card [5]: 19 (Heart 8)

Alice plays: 27 (Club 3)

Bob plays: 45 (Diamond 8)

Alice plays: 48 (Diamond Jack)

Bob plays: 13 (Heart 2)

Alice plays: 23 (Heart Queen)

Bob plays: 10 (Spade Queen)

Alice plays: 16 (Heart 5)

Bob plays: 19 (Heart 8)

Alice plays: 21 (Heart 10)

Bob plays: 3 (Spade 5)

Счёт текущей партии:

Alice: 3

Bob: 2

Кол-во честных карт Vector: 5

Открытый ключ оппонента: 23

Закрытый ключ оппонента: 8846663