Министерство образования республики Молдова

Технический Университет Молдовы

Департамент Программной Инженерии и Автоматики

О т ч ё т

Лабораторная работа №5

По предмету: Metode criptografice de protective a informatiei

Тема: Criptografia cu chei publice

Выполнил студ. гр. SI-202 Абабий Эдуард

Проверил Aureliu Zgureanu

Кишинёв – 2023

Задание 2.1.

Используя платформу wolframalpha.com или приложение Wolfram Mathematica, сгенерируйте ключи, зашифруйте и расшифруйте сообщение m = Имя Фамилия, применяя алгоритм RSA. Значение n должно быть не менее 2048 бит

```
In[15]:= p = NextPrime[2^1024]
Out[15]= 179 769 313 486 231 590 772 930 519 07
         913 110 540 827 237 163 350 510 684

In[14]:= q = NextPrime[2^1048]
Out[14]= 3 016 028 602 530 220 424 421 062 271
         238 620 775 335 376 572 758 801 465
```

Сгенерировали 2 простых числа p и q

Количество бит числа p – 1025

Количество бит числа q – 1049

```
In[16]:= n = p * q
Out[16]= 542 189 391 331 696 172 661 6
         488 665 744 991 822 837 769
         681 562 164 559 112 938 862
```

Вычислили n, количество бит которого 2073

```
In[17]:= fn = (p - 1) * (q - 1)
Out[17]= 542 189 391 331 696 172 661 6
         488 665 744 991 822 837 769
         879 638 542 717 670 648 035
```

Затем вычислили функцию Эйлера, то есть количество простых чисел от 1 до n-1 , которые взаимно просты с n

```
In[19]:= e = RandomInteger[{2, fn - 1}]
Out[19]= 119 145 185 239 943 555 893 484 823 506 483 292 923 159 097 073 82
         000 538 418 599 435 984 555 460 408 555 116 699 621 385 072 367 4
         544 459 884 190 198 141 806 106 315 938 010 959 119 048 566 510 3

In[24]:= While[GCD[e, fn] ≠ 1, e = RandomInteger[{2, fn - 1}]]
```

Необходимо выбрать открытую экспоненту e, которая является целым числом, взаимно простым с функцией Эйлера fn(n). Для этого выбирается случайное число e в диапазоне от 2 до fn(n) - 1. Затем происходит проверка на взаимную простоту чисел e и fn(n), используя функцию GCD, которая вычисляет наибольший общий делитель двух чисел.

Если GCD(e, phi(n)) не равно 1, то это означает, что e не является взаимно простым с fn(n), и выбирается новое случайное значение e и происходит повторная проверка.

Таким образом, цикл While генерирует случайное число e до тех пор, пока оно не будет взаимно простым с fn(n).

```
In[26]:= d = PowerMod[e, -1, fn]
Out[26]= 329 836 976 619 918 999 118 447 243
         784 776 800 778 373 680 725 254 9
         629 465 001 847 916 553 567 905 1
```

Секретная экспонента d вычисляется с использованием открытой экспоненты e и функции Эйлера fn(n). Она используется для расшифровки зашифрованного сообщения

```
In[30]:= m = "Ababii Eduard"
         c = PowerMod[ToCharacterCode[m], e, n]
Out[30]= Ababii Eduard

Out[31]= {534 910 315 359 047 927 329 010 716 293 850 590 516 6
           324 030 402 566 037 300 020 929 326 376 305 081 565
           171 204 457 159 540 057 196 607 849 162 722 231 899
           790 325 419 595 535 742 313 155 493 475 541 273 591
```

В данном случае используется функция ToCharacterCode, которая преобразует строку m в список числовых значений, соответствующих кодам символов в ASCII.

Затем каждое числовое значение символа в сообщении m возводится в степень e по модулю n с помощью функции PowerMod. Результатом является зашифрованный список числовых значений c.

После этого зашифрованный список числовых значений может быть отправлен получателю, который сможет расшифровать его, используя свою секретную экспоненту d и свой секретный ключ (n, d).

```
In[32]:= decrypted = FromCharacterCode[PowerMod[c, d, n]]
Out[32]= Ababii Eduard
```

Для расшифровки каждого элемента списка c, получатель возводит его в степень d по модулю n с помощью функции PowerMod. Результатом будет список числовых значений decrypted, которые соответствуют исходной строке m.

Чтобы получить исходную строку из списка числовых значений, используется функция FromCharacterCode, которая преобразует список числовых значений в соответствующую строку символов с помощью заданной кодировки (например, ASCII или Unicode). Таким образом, переменная decrypted будет содержать расшифрованное сообщение m.

Задание 2.2.

Используя платформу wolframalpha.com или приложение Wolfram Mathematica, сгенерируйте ключи, зашифруйте и расшифруйте сообщение m = Имя Фамилия, применяя алгоритм Эль-Гамаля. Значения p и генератора заданы ниже.

p=323170060713110073001535134778251633624880571334890751745884341392698068341362100027920563626401646854585563579353308169288290230805734726252735547424612457410262025279165729728627063003252634282131457669314142236542209411113486299916574782680342305530863490506355577122191878903327295696961297438562417412362372251973464026918557976797682301462539793305801522685873076119753243646747585546071504389684494036613049769781285429595865959756705128385213278446852292550456827287911372009893187395914337417583782600027803497319855206060753323412260325468408812003110590748428100399496695611969695624862903233807283912703, care are 2048 biți și generatorul g=2.

```
In[35]:= p = 32317006071311007300153513477825163362488
          55577122191878903327295696961297438562417411
          06060753323412260325468408812003110590748424

Out[35]= 32 317 006 071 311 007 300 153 513 477 825 163 362 48
          478 268 034 230 553 086 349 050 635 557 712 219 187
          272 879 113 720 098 931 873 959 143 374 175 837 826

In[33]:= g = 2

Out[33]= 2

In[36]:= a = RandomInteger[{1, p - 2}]
          A = PowerMod[g, a, p]
```

Генерация закрытого ключа a и соответствующего ему открытого ключа A. Закрытый ключ обычно представляет собой случайно выбираемое число в определенном диапазоне (в данном случае в диапазоне от 1 до p-2), который затем используется для вычисления открытого ключа.

В формуле A = PowerMod[g, a, p] мы берем число g (в данном случае 2) и возводим его в степень a по модулю p, чтобы получить значение открытого ключа A. Затем A будет использован в дальнейшем для зашифрования сообщения.

```
In[38]:= m = "Ababii Eduard"
Out[38]= Ababii Eduard

In[39]:= k = RandomInteger[{1, p - 2}]
         K = PowerMod[g, k, p]
Out[39]= 28 481 287 170 197 020 404 149 803 16
          500 555 984 990 445 349 897 593 029
          920 572 133 916 667 012 558 657 498

Out[40]= 12 087 660 503 828 824 414 399 416 03
          329 807 049 143 995 780 362 443 302
          213 259 169 145 338 004 922 056 958
```

Выбор случайного числа k из диапазона [1, p-2], которое используется для вычисления открытого ключа K. Открытый ключ K является результатом возведения числа g (генератора) в степень k по модулю p.

```
In[43]:= c1 = PowerMod[g, k, p]
         c2 = Mod[ToCharacterCode[m] * PowerMod[A, k, p], p]
Out[43]= 12 087 660 503 828 824 414 399 416 031 959 588 734 023 266 388 76
          329 807 049 143 995 780 362 443 302 250 641 523 538 612 727 422
          213 259 169 145 338 004 922 056 958 907 270 141 823 742 189 301

Out[44]= {12 791 529 238 180 875 282 352 801 632 587 696 526 606 789 243 5
          447 189 435 440 599 010 870 455 349 052 526 009 680 913 720 73
          429 845 704 243 309 855 617 599 633 171 891 135 762 125 222 03
```

Здесь мы шифруем сообщение m, используя публичный ключ A и сессионный ключ k.

Переменная c1 вычисляется как g в степени k по модулю p, т.е. это первая часть зашифрованного сообщения, которую отправитель передает получателю.

Переменная c2 вычисляется как массив кодов символов сообщения m, умноженный на A в степени k по модулю p. Далее происходит взятие остатка от деления этого произведения на p. Таким образом, c2 представляет собой вторую часть зашифрованного сообщения.

В итоге, пара (c1, c2) будет отправлена получателю в качестве зашифрованного сообщения.

```
In[45]:= s = PowerMod[c1, p - 1 - a, p]
Out[45]= 19 097 166 599 456 986 901 991 275 917 419 821 964 319 67
         248 215 493 045 138 775 838 585 457 131 188 645 559 172
         250 098 253 779 245 975 127 104 035 353 369 568 510 275

In[46]:= decrypted = FromCharacterCode[Mod[c2 * s, p]]
Out[46]= Ababii Eduard
```

Сначала мы вычисляем секретный ключ с помощью операции возведения в степень по модулю p:

s = PowerMod[c1, p - 1 - a, p]

Здесь мы используем значение c1, которое было вычислено при зашифровывании сообщения и передано получателю. Мы также используем значение a, которое было вычислено отправителем и является частью его секретного ключа.

Затем мы используем полученный секретный ключ s, чтобы расшифровать значение c2, которое также было передано получателю:

decrypted = FromCharacterCode[Mod[c2 * s, p]]

Значение c2 было вычислено отправителем, а затем зашифровано с помощью открытого ключа получателя. Здесь мы используем операцию умножения и взятия остатка от деления на модуль p, чтобы дешифровать сообщение. Затем мы используем функцию FromCharacterCode, чтобы получить исходную строку из расшифрованных значений.

Задание 3.

Используя платформу wolframalpha.com или приложение Wolfram Mathematica, выполнить обмен ключами Диффи-Хелмана между Алисой и Бобом, которые используют алгоритм AES с 256-битным ключом. Секретные числа a и b должны быть выбраны случайным образом в соответствии с требованиями алгоритма. Значения p и генератора заданы ниже

p=323170060713110073001535134778251633624880571334890751745884341392698068341362100027920563626401646854585563579353308169288290230805734726252735547424612457410262025279165729728627063003252634282131457669314142236542209411113486299916574782680342305530863490506355577122191878903327295696961297438562417412362372251973464026918557977679768230146253979330580152268587307611975324364674758554607150438968449403661304976978128542959586595975670512838521327844685229255045682728791137200989318739591433741758378260002780349731985520606075332341226032546840881200311059074842810039949669561196969562486290323380728391270391, care are 2048 biți și generatorul g=2.

```
In[47]:= p = 3231700607131100730015351347782551
          555577122191878903327295696961297 43
          758378260002780349731985520606075 33
         g = 2
Out[47]= 3 231 700 607 131 100 730 015 351 347 782 5
          747 826 803 423 055 308 634 905 063 555 5
          283 852 132 784 468 522 925 504 568 272 8

Out[48]= 2

In[49]:= a = RandomInteger[{1, p - 1}]
         b = RandomInteger[{1, p - 1}]
Out[49]= 2 315 042 944 048 057 723 328 986 949 803 1
          199 182 077 425 192 131 654 659 677 827 5
          401 310 682 434 181 497 174 648 164 226 4

Out[50]= 2 043 258 559 690 831 398 440 123 800 168 8
          006 692 782 365 003 560 556 442 264 673 6
          801 445 293 015 429 849 413 849 747 344 0
```

Переменные a и b - секретные случайные числа, которые выбирают Алиса и Боб соответственно. Они являются приватными ключами, которые будут использоваться для вычисления общего ключа, который будут использовать для шифрования сообщений с помощью алгоритма AES с 256-битным ключом.

```
In[51]:= A = PowerMod[g, a, p]
         B = PowerMod[g, b, p]

Out[51]= 2 448 818 343 786 563 203 23
          706 194 979 454 017 097 35
          023 673 323 086 682 183 69

Out[52]= 1 453 816 896 987 533 309 10
          025 519 229 048 041 044 60
          479 520 368 768 745 792 36
```

Далее, после того, как Алиса и Боб выбрали свои секретные числа, они вычисляют соответствующие открытые числа.

Алиса вычисляет своё открытое число, используя формулу A = g^a mod p, где g - это генератор, a - это её секретное число, а p - это большое простое число, известное обоим участникам.

Боб вычисляет своё открытое число, используя аналогичную формулу B = g^b mod p, где b - это его секретное число.

Таким образом, Алиса и Боб обменялись открытыми числами A и B, не раскрывая при этом свои секретные числа.

```
In[53]:= sA = PowerMod[B, a, p]
         sB = PowerMod[A, b, p]

Out[53]= 715 292 119 311 553 527 215 33
          138 146 331 815 086 343 291 9
          365 867 082 741 312 431 492 3

Out[54]= 715 292 119 311 553 527 215 33
          138 146 331 815 086 343 291 9
          365 867 082 741 312 431 492 3

In[55]:= sA == sB

Out[55]= True
```

После того, как Алиса и Боб выбрали свои секретные числа, они вычисляют открытые числа, используя параметры g и p и свои секретные числа a и b, соответственно.

Далее, они передают эти открытые числа друг другу. Затем каждый из них вычисляет общий секретный ключ, используя открытое число, полученное от другого и своё секретное число.

После этого они должны получить одинаковый общий секретный ключ sA, который означает, что процедура Диффи-Хеллмана прошла успешно и они могут использовать sA для защищенного обмена сообщениями. Сравнение sA == sB проверяет, что общий секретный ключ sA у Алисы и Боба совпадает.

Вывод:

В данной лабораторной работе я научился зашифровывать сообщения с помощью алгоритма RSA, Эль-Гамаля и создавать пару открытых и закрытых ключей для обмена.