А-08-19, Балашов, ЛР9-10

1. Разработать программный модуль для формирования системных параметров RSA (модуль, открытый ключ, секретный ключ) на основе заданных номеров простых чисел: Q = Prime[10000-N], P = Prime[10000+N], где N – номер по списку в группе.

In[232]:=
```
N1 = 4
SeedRandom[N1]
Q = Prime[10 000 - N1]
P = Prime[10 000 + N1]
rsaParamsModule[Q_, P_] := Module[{openKey, secretKey, n, phi},
  n = P * Q;
  phi = (Q - 1) * (P - 1);
  openKey = RandomInteger[{1 + 1, phi}];
  While[GCD[openKey, phi] ≠ 1, openKey = RandomInteger[{1 + 1, phi}]];
  secretKey = PowerMod[openKey, -1, phi];
  While[GCD[secretKey, n] ≠ 1,
   While[GCD[openKey, phi] ≠ 1, openKey = RandomInteger[{1 + 1, phi}]];
   secretKey = PowerMod[openKey, -1, phi]];
  Print["N = ", P * Q];
  Print["Open key = ", openKey];
  Print["Secret key = ", secretKey];
 ]
rsaParamsModule[Q, P]
```

Out[232]= 4

Out[233]= RandomGeneratorState[Method: ExtendedCA; State hash: −6 064 038 039 835 711 508]

Out[234]= 104 707

Out[235]= 104 773

```
N = 10 970 466 511
Open key = 10 963 688 705
Secret key = 6 908 245 577
```

In[238]:=
```
n = 10 972 975 879
openKey = 2 396 097 779
secretKey = 4 047 511 979
```

Out[238]= 10 972 975 879

Out[239]= 2 396 097 779

Out[240]= 4 047 511 979

2. Импортировать текстовый файл Text-N с номером по списку в группе из папки Plaintext1RSA. Провести анализ кодов текста и привести к виду : 1XXX или 2XXX - четыре десятичных цифры, представляющие собой блок для шифрования в RSA. Например: код пробела 32 представляем как 2000+32=2032.

In[241]:=
```
openText1 = " информация должна быть неизвестной.
     Субъект – это активный компонент
     системы,который может стать причиной потока информации
     от объекта к субъекту или изменения состояния системы.
     Объект – пассивный компонент системы,
     хранящий, принимающий или передающий информацию. Доступ
     к объекту означает доступ к содержащейся в нем информации.
     Целостность информации обеспечивается
     в том случае, если данные в системе не отличаются в
     семантическом отношении от данных в исходных документах, т.е. если не
     произошло их случайного или преднамеренного искажения или разрушения.
     Целостность компонента или ресурса системы – это свойство
     компонента или ресурса быть неизменными в семантическом
     смысле при функционировании системы в условиях случайных
     или преднамеренных искажений или разрушающих воз-
действий.
     Доступность компонента или ресурса системы – это свойство компонента или
     ресурса быть доступным для авторизованных законных субъектов системы.
     Под угрозой безопасности АСОИ понимаются возможные воздействия на АСОИ,
     которые прямо или косвенно могут нанести ущерб ее безопасности.
     Ущерб безопасности подразумевает нарушение состояния защищенности
     информации, содержащейся и обрабатывающейся в АСОИ. С
     понятием угрозы безопасности тесно связано понятие уязв";
openTextCode1 = ToCharacterCode[openText1];
Do[If[openTextCode1[[i]] < 1000, openTextCode1[[i]] += 2000],
 {i, 1, Length[openTextCode1]}]
Tally[openTextCode1][[All, 1]]
```

In[245]:=
```
N[Entropy[2, openTextCode1]]
```

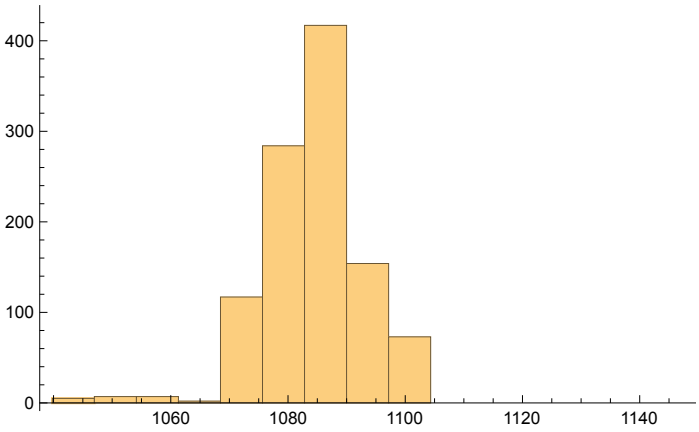
Out[245]=
```
4.61083
```

(энтропия текста)

3. Построить гистограмму распределения кодов символов открытого текста.

In[246]:=
```
Histogram[openTextCode1]
```

Out[246]=



4. Зашифровать текст на открытом ключе и определить энтропию шифртекста.

In[247]:=
```
cipherCodeText1 = {};
Do[
 AppendTo[
  cipherCodeText1,
  PowerMod[openTextCode1[[i]], openKey, n]], {i, 1, Length[openTextCode1]}]
cipherCodeText1
```

In[250]:=
```
N[Entropy[2, cipherCodeText1]]
```

Out[250]=
```
4.61083
```

(энтропии открытого текста и шифртекста равны 4.64524)

5. Провести расшифрование на секретном ключе.

In[251]:=
```
openTextCode1 = Table[0, {i, Length[cipherCodeText1]}];
Do[openTextCode1[[i]] = PowerMod[cipherCodeText1[[i]], secretKey, n],
  {i, 1, Length[cipherCodeText1]}];
Do[If[openTextCode1[[i]] ≥ 2000 && openTextCode1[[i]] < 3000,
   openTextCode1[[i]] -= 2000], {i, 1, Length[openTextCode1]}];
FromCharacterCode[openTextCode1]
```

(полученный при расшифровке текст совпал с исходным)

6. Сформировать из модифицированных блоков открытого текста (см. п.2) десятичные эквиваленты биграмм: {1079,2032}->{10792032}.

In[255]:=
```
openText1 = " информация должна быть неизвестной.
     Субъект - это активный компонент
     системы,который может стать причиной потока информации
     от объекта к субъекту или изменения состояния системы.
     Объект - пассивный компонент системы,
     хранящий, принимающий или передающий информацию. Доступ
     к объекту означает доступ к содержащейся в нем информации.
     Целостность информации обеспечивается
     в том случае, если данные в системе не отличаются в
     семантическом отношении от данных в исходных документах, т.е. если не
     произошло их случайного или преднамеренного искажения или разрушения.
     Целостность компонента или ресурса системы - это свойство
     компонента или ресурса быть неизменными в семантическом
     смысле при функционировании системы в условиях случайных
     или преднамеренных искажений или разрушающих воз-
действий.
     Доступность компонента или ресурса системы - это свойство компонента или
     ресурса быть доступным для авторизованных законных субъектов системы.
     Под угрозой безопасности АСОИ понимаются возможные воздействия на АСОИ,
     которые прямо или косвенно могут нанести ущерб ее безопасности.
     Ущерб безопасности подразумевает нарушение состояния защищенности
     информации, содержащейся и обрабатывающейся в АСОИ. С
     понятием угрозы безопасности тесно связано понятие уязв";
openTextCode1 = ToCharacterCode[openText1];
Do[If[openTextCode1[[i]] < 1000, openTextCode1[[i]] += 2000],
 {i, 1, Length[openTextCode1]}]
openCodeText1 = {};
openCodeText2 = {};
Do[
 AppendTo[openCodeText1, openTextCode1[[2 * i - 1]]];
 AppendTo[openCodeText2, openTextCode1[[2 * i]]],
 {i, 1, IntegerPart[Length[openTextCode1] / 2]}]
binaryCodeText1 = openCodeText1 * 10 000 + openCodeText2
```

7. Провести шифрование блоков биграмм на открытом ключе. Определить энтропию шифр текста.

In[262]:=
```
cipherCodeBinaryText1 = {};
Do[AppendTo[cipherCodeBinaryText1, PowerMod[binaryCodeText1[[i]], openKey, n]],
 {i, 1, Length[binaryCodeText1]}]
cipherCodeBinaryText1
```

In[265]:=
```
N[Entropy[2, cipherCodeBinaryText1]]
```

Out[265]=
```
7.50035
```

8. Расшифровать полученный шифртекст и вывести его в виде строки.

In[266]:=
```
binaryCodeText1 = {};
Do[
 AppendTo[binaryCodeText1, PowerMod[cipherCodeBinaryText1[[i]], secretKey, n]],
 {i, 1, Length[cipherCodeBinaryText1]}]
openCodeText1 = IntegerPart[binaryCodeText1 / 10 000];
openCodeText2 = Mod[binaryCodeText1, 10 000];
openTextCode1 = {};
Do[
 AppendTo[openTextCode1, openCodeText1[[i]]];
 AppendTo[openTextCode1, openCodeText2[[i]],
 {i, 1, Length[binaryCodeText1]}]
Do[If[openTextCode1[[i]] ≥ 2000 && openTextCode1[[i]] < 3000,
   openTextCode1[[i]] -= 2000], {i, 1, Length[openTextCode1]}];
FromCharacterCode[openTextCode1]
```

Out[273]=
```
информация должна быть неизвестной.
   Субъект – это активный компонент системы,который может стать причиной
  потока информации от объекта к субъекту или изменения состояния системы.
   Объект – пассивный компонент системы,
  хранящий, принимающий или передающий информацию. Доступ к
  объекту означает доступ к содержащейся в нем информации.
   Целостность информации обеспечивается в том случае,
  если данные в системе не отличаются в семантическом отношении
  от данных в исходных документах, т.е. если не произошло их
  случайного или преднамеренного искажения или разрушения.
   Целостность компонента или ресурса системы – это
  свойство компонента или ресурса быть неизменными в семантическом
  смысле при функционировании системы в условиях случайных
  или преднамеренных искажений или разрушающих воз-
действий.
   Доступность компонента или ресурса системы – это свойство компонента или
  ресурса быть доступным для авторизованных законных субъектов системы.
   Под угрозой безопасности АСОИ понимаются возможные
  воздействия на АСОИ, которые прямо или косвенно могут нанести
  ущерб ее безопасности.  Ущерб безопасности подразумевает нарушение
  состояния защищенности информации, содержащейся и обрабатывающейся
  в АСОИ. С понятием угрозы безопасности тесно связано понятие уяз
```

текст снова совпал с исходным

9. Ввести следующие изменения в Text-N и создать модифицированные строки:

text1– убрать точку в Text-N;

text2 – добавить пробел в Text-N;

text3 – поменять местами две расположенные рядом (разные) буквы в Text-N.

In[274]:=
```
text1 = " информация должна быть неизвестной.
    Субъект – это активный компонент системы,который может стать причиной
   потока информации от объекта к субъекту или изменения состояния системы.
```

Объект – пассивный компонент системы,
хранящий, принимающий или передающий информацию. Доступ к
объекту означает доступ к содержащейся в нем информации.
Целостность информации обеспечивается
в том случае, если данные в системе не отличаются в
семантическом отношении от данных в исходных документах, т.е. если не
произошло их случайного или преднамеренного искажения или разрушения.
Целостность компонента или ресурса системы – это свойство
компонента или ресурса быть неизменными в семантическом
смысле при функционировании системы в условиях случайных
или преднамеренных искажений или разрушающих воз-
действий.
Доступность компонента или ресурса системы – это свойство компонента или
ресурса быть доступным для авторизованных законных субъектов системы.
Под угрозой безопасности АСОИ понимаются возможные воздействия на АСОИ,
которые прямо или косвенно могут нанести ущерб ее безопасности.
Ущерб безопасности подразумевает нарушение состояния защищенности
информации, содержащейся и обрабатывающейся в АСОИ С
понятием угрозы безопасности тесно связано понятие уязв"

text2 = " информация должна быть неизвестной.

Субъект – это активный компонент системы,который может стать причиной
потока информации от объекта к субъекту или изменения состояния системы.
Объект – пассивный компонент системы,
хранящий, принимающий или передающий информацию. Доступ
к объекту означает доступ к содержащейся в нем информации.
Целостность информации обеспечивается
в том случае, если данные в системе не отличаются в
семантическом отношении от данных в исходных документах, т.е. если не
произошло их случайного или преднамеренного искажения или разрушения.
Целостность компонента или ресурса системы – это свойство
компонента или ресурса быть неизменными в семантическом
смысле при функционировании системы в условиях случайных
или преднамеренных искажений или разрушающих воз-
действий.
Доступность компонента или ресурса системы – это свойство компонента или
ресурса быть доступным для авторизованных законных субъектов системы.
Под угрозой безопасности АСОИ понимаются возможные воздействия на АСОИ,
которые прямо или косвенно могут нанести ущерб ее безопасности.
Ущерб безопасности подразумевает нарушение состояния защищенности
информации, содержащейся и обрабатывающейся в АСОИ. С
понятием угрозы безопасности тесно связано понятие уязв "

text3 = " нифомация должна быть неизвестной.

Субъект – это активный компонент системы,который может стать причиной
потока информации от объекта к субъекту или изменения состояния системы.
Объект – пассивный компонент системы,
хранящий, принимающий или передающий информацию. Доступ
к объекту означает доступ к содержащейся в нем информации.

Целостность информации обеспечивается в том случае, если данные в системе не отличаются в семантическом отношении от данных в исходных документах, т.е. если не произошло их случайного или преднамеренного искажения или разрушения.

Целостность компонента или ресурса системы – это свойство компонента или ресурса быть неизменными в семантическом смысле при функционировании системы в условиях случайных или преднамеренных искажений или разрушающих воздействий.

Доступность компонента или ресурса системы – это свойство компонента или ресурса быть доступным для авторизованных законных субъектов системы.

Под угрозой безопасности АСОИ понимаются возможные воздействия на АСОИ, которые прямо или косвенно могут нанести ущерб ее безопасности. Ущерб безопасности подразумевает нарушение состояния защищенности информации, содержащейся и обрабатывающейся в АСОИ. С понятием угрозы безопасности тесно связано понятие уязв"

10. Найти расстояние Дамерау-Левенштейна (DLD) - минимальное количество операций вставки одного символа, удаления одного символа и замены одного символа на другой, необходимых для превращения одной строки в другую- (DamerauLevenshteinDistance[,]) между строкой Text-N и строками text1, text2, text3.

In[277]:=
```
DamerauLevenshteinDistance[openText1, text1]
DamerauLevenshteinDistance[openText1, text2]
DamerauLevenshteinDistance[openText1, text3]
```

Out[277]= 1

Out[278]= 1

Out[279]= 1

	text1	text2	text3
DLD	1	1	1

11. Найти расстояние Дамерау-Левенштейна (DamerauLevenshteinDistance[,]) между значениями хэш-функций Hash[] строки Text-N и значениями хэш-функций строк text1, text2, text3.

In[280]:=
```
hashMsg = Hash[openText1]
hash1 = Hash[text1]
hash2 = Hash[text2]
hash3 = Hash[text3]
```

Out[280]= 2 432 905 821 434 766 105

Out[281]= 8 305 437 116 682 763 522

Out[282]= 4 378 173 096 024 090 805

Out[283]= 4 419 797 063 243 909 028

In[284]:=
```
DamerauLevenshteinDistance[ToString[hashMsg], ToString[hash1]]
DamerauLevenshteinDistance[ToString[hashMsg], ToString[hash2]]
DamerauLevenshteinDistance[ToString[hashMsg], ToString[hash3]]
```

Out[284]= 16

Out[285]= 15

Out[286]= 15

	Hash [text1]	Hash[text2]	Hash[text3]
DLD	16	16	16

12. Определить расстояние ДЛ между значениями хэш-функций строк Text-N и text1 для алгоритмов хэширования, приведенных в таблице.

In[287]:=
```
DamerauLevenshteinDistance[openText1, text1]
```

Out[287]= 1

In[288]:=
```
DamerauLevenshteinDistance[
 ToString[Hash[openText1, "CRC32"]], ToString[Hash[text1, "CRC32"]]]
```

Out[288]= 8

In[289]:=
```
DamerauLevenshteinDistance[
 ToString[Hash[openText1, "MD5"]], ToString[Hash[text1, "MD5"]]]
```

Out[289]= 32

In[290]:=
```
DamerauLevenshteinDistance[
 ToString[Hash[openText1, "SHA"]], ToString[Hash[text1, "SHA"]]]
```

Out[290]= 38

In[291]:=
```
DamerauLevenshteinDistance[
 ToString[Hash[openText1, "SHA256"]], ToString[Hash[text1, "SHA256"]]]
```

Out[291]= 58

In[292]:=
```
DamerauLevenshteinDistance[
 ToString[Hash[openText1, "SHA384"]], ToString[Hash[text1, "SHA384"]]]
```

Out[292]= 86

In[293]:=
```
DamerauLevenshteinDistance[
 ToString[Hash[openText1, "SHA512"]], ToString[Hash[text1, "SHA512"]]]
```

Out[293]= 117

13. Преобразовать свою фамилию и имя в числовой код m (а->1, ..я->32), получить криптограмму с зашифровав m на секретном ключе ks. Рассмотреть два варианта: разбиение m на максимальное число элементов и разбиение (или его отсутствие) m на минимально возможное число элементов, при этом допускается изменение параметров RSA.

In[294]:=
```
openText1 = "балашовсавва";
charText1 = Characters[openText1];
msgCodeMinList = {};
Do[
 AppendTo[msgCodeMinList, ToCharacterCode[charText1[[i]]] - 1071],
 {i, 1, StringLength[openText1]}]
msgCodeMinList = Flatten[msgCodeMinList]
```

Out[298]= {2, 1, 12, 1, 25, 15, 3, 18, 1, 3, 3, 1}

In[299]:=
```
msgCodeMax =
 FromDigits[Flatten[Table[PadLeft[IntegerDigits[msgCodeMinList[[i]]], 2],
    {i, 1, Length[msgCodeMinList]}]]]
```

Out[299]= 20 112 012 515 031 801 030 301

In[300]:=
```
SeedRandom[N1]
RandomPrime[{Sqrt[100^11], Sqrt[100^11] * 10}, 2]
```

Out[300]= RandomGeneratorState[Method: ExtendedCA
State hash: 9 145 254 414 741 617 443]

Out[301]= {670 851 072 517, 706 680 115 903}

In[302]:=
```
rsaParamsModule[892 665 323 731, 350 261 647 993]
```

N = 312 666 427 396 224 911 421 883

Open key = 310 364 379 519 556 483 574 101

Secret key = 215 990 638 475 088 643 273 981

In[303]:=
```
nLong =312 666 427 396 224 911 421 883
openKeyLong = 95 459 891 171 862 768 459 367
secretKeyLong = 52 592 236 725 673 796 713 063
```

Out[303]= 312 666 427 396 224 911 421 883

Out[304]= 95 459 891 171 862 768 459 367

Out[305]= 52 592 236 725 673 796 713 063

In[306]:=
```
cryptCodeMinList = {};
Do[
 AppendTo[
  cryptCodeMinList,
  PowerMod[msgCodeMinList[[i]], secretKeyLong, nLong]],
 {i, 1, Length[msgCodeMinList]}]
cryptCodeMinList
```

Out[308]= {11 719 095 496 731 650 299 727, 1, 262 878 728 072 002 346 760 832,
1, 298 534 804 050 277 359 404 271, 172 492 641 208 996 860 866 343,
270 660 851 830 313 279 062 996, 140 718 830 989 352 764 622 586, 1,
270 660 851 830 313 279 062 996, 270 660 851 830 313 279 062 996, 1}

In[309]:=
```
cryptCodeMax = PowerMod[msgCodeMax, secretKeyLong, nLong]
```

Out[309]= 267 257 565 138 513 633 561 663

14. Расшифровать два варианта криптограммы с на ключе ko и получить m.

In[310]:=
```
msgCodeMinList = {};
Do[
 AppendTo[
  msgCodeMinList,
  PowerMod[cryptCodeMinList[[i]], openKeyLong, nLong]],
 {i, 1, Length[cryptCodeMinList]}]
msgCodeMinList
```

Out[312]= {2, 1, 12, 1, 25, 15, 3, 18, 1, 3, 3, 1}

In[313]:=
```
FromCharacterCode[msgCodeMinList + 1071]
```

Out[313]= балашовсавва

15. Преобразовать строку хеш-кода сообщения m в последовательность (список) чисел при минимальном возможном числе элементов шифрования, определить длину этого списка и подготовить два новых списка для шифр текста и восстановленного (расшифрованного) хеш-кода. Номер варианта хэш-функции Nmod5+1:

1	2	3	4	5
MD5	SHA	SHA256	SHA384	SHA512

In[314]:=
```
Mod[4, 5] + 1
```

Out[314]= 5

In[315]:=
```
msgHash = Hash[openText1, "SHA512"]
```

Out[315]= 10 036 238 053 069 880 501 118 656 072 266 056 640 138 530 271 337 404 246 637 525 484 880 312 ⁘
896 756 542 747 168 763 366 265 228 062 932 598 908 406 034 071 635 279 010 475 850 073 179 ⁘
985 114 213 065 690

In[316]:=
```
openText1
```

Out[316]= балашовсавва

In[317]:=
```
msgHashList = IntegerDigits[msgHash]
Length[msgHashList]
```

Out[317]= {1, 0, 0, 3, 6, 2, 3, 8, 0, 5, 3, 0, 6, 9, 8, 8, 0, 5, 0, 1, 1, 1, 8, 6, 5,
6, 0, 7, 2, 2, 6, 6, 0, 5, 6, 6, 4, 0, 1, 3, 8, 5, 3, 0, 2, 7, 1, 3, 3, 7, 4,
0, 4, 2, 4, 6, 6, 3, 7, 5, 2, 5, 4, 8, 4, 8, 8, 0, 3, 1, 2, 8, 9, 6, 7, 5, 6,
5, 4, 2, 7, 4, 7, 1, 6, 8, 7, 6, 3, 3, 6, 6, 2, 6, 5, 2, 2, 8, 0, 6, 2, 9, 3,
2, 5, 9, 8, 9, 0, 8, 4, 0, 6, 0, 3, 4, 0, 7, 1, 6, 3, 5, 2, 7, 9, 0, 1, 0, 4,
7, 5, 8, 5, 0, 0, 7, 3, 1, 7, 9, 9, 8, 5, 1, 1, 4, 2, 1, 3, 0, 6, 5, 6, 9, 0}

Out[318]= 155

In[319]:=
```
msgHashList = PadLeft[msgHashList, 120];
msgHashList = Partition[msgHashList, 20];
Do[
 msgHashList[[i]] = FromDigits[msgHashList[[i]]],
 {i, 1, Length[msgHashList]}]
msgHashList
```

Out[322]=
```
{64 013 853 027 133 740 424, 66 375 254 848 803 128 967, 56 542 747 168 763 366 265,
 22 806 293 259 890 840 603, 40 716 352 790 104 758 500, 73 179 985 114 213 065 690}
```

In[323]:=
```
Length[msgHashList]
```

Out[323]= 6

In[324]:=
```
cryptHashList = Table[0, {i, 6}]
hashFromCryptList = Table[0, {i, 6}]
```

Out[324]= {0, 0, 0, 0, 0, 0}

Out[325]= {0, 0, 0, 0, 0, 0}

16. Провести операцию шифрования хеш-кода на ключе ks и зафиксировать результат.

In[326]:=
```
Do[
 cryptHashList[[i]] = PowerMod[msgHashList[[i]], secretKeyLong, nLong],
 {i, 1, Length[msgHashList]}]
cryptHashList
```

Out[327]=
```
{286 868 373 309 645 563 287 311, 145 886 601 887 610 628 426 283,
 128 513 941 594 365 089 084 659, 7 733 910 287 123 085 972 931,
 227 141 830 843 045 281 544 087, 223 905 860 332 308 437 091 150}
```

17. Провести операцию расшифрования хеш-кода на ключе ko и зафиксировать результат.

In[328]:=
```
Do[
 hashFromCryptList[[i]] = PowerMod[cryptHashList[[i]], openKeyLong, nLong],
 {i, 1, Length[cryptHashList]}]
hashFromCryptList
```

Out[329]=
```
{64 013 853 027 133 740 424, 66 375 254 848 803 128 967, 56 542 747 168 763 366 265,
 22 806 293 259 890 840 603, 40 716 352 790 104 758 500, 73 179 985 114 213 065 690}
```

18. Сравнить результат, полученный в п. 17 с исходным хэш-кодом п.15.

In[330]:=
```
msgHashList
hashFromCryptList
HammingDistance[msgHashList, hashFromCryptList]
```

Out[330]=
```
{64 013 853 027 133 740 424, 66 375 254 848 803 128 967, 56 542 747 168 763 366 265,
 22 806 293 259 890 840 603, 40 716 352 790 104 758 500, 73 179 985 114 213 065 690}
```

Out[331]=
```
{64 013 853 027 133 740 424, 66 375 254 848 803 128 967, 56 542 747 168 763 366 265,
 22 806 293 259 890 840 603, 40 716 352 790 104 758 500, 73 179 985 114 213 065 690}
```

Out[332]= 0

In[333]:=
```
Do[
 If[msgHashList[[i]] == hashFromCryptList[[i]],
  Print["совпадают"],
  Print["не совпадают"]; Break[]],
 {i, 1, Length[msgHashList]}]
```

```
совпадают
совпадают
совпадают
совпадают
совпадают
совпадают
```