

```
In[*]:= (*Определение варианта*)
nNomBrs = 10;
nNom = (nNomBrs - 1) * 2 + 1
nNomImage = Mod[nNom, 4] + 1
           |__остаток от деления
```

Out[\*]:= 19

Out[\*]:= 4

```
In[*]:= (*Постановка задания*)
(*Размер фотографии: 240x320*)
(*Размер блока, точек: 4x4*)
(*Метод встраивания ЦВЗ: метод блочного сокрытия*)
(*Сего путь: последовательно, непрерывно, по столбцам.
*)
blockSize = {4, 4};
```

```
In[*]:= (*Подготовка изображения*)
image = Import["D:\\GitHub Repos\\stud\\mag\\Sem9\\КР ЦТЗИ\\МУ\\240x320.bmp"]
        |__импорт |__дифференцировать
ImageDimensions[image]
        |__размеры изображения
```

Out[\*]:=



Out[\*]:= {240, 320}

In[ ]:= (\*Информация для встраивания\*)

```
rawInfo = "Наименование документа: Зачетная книжка;
Номер документа: 0020201253;
Фамилия: Кутузов;
Имя: Илья;
Отчество: Геннадьевич;
Группа: А-12М-20;
Номер по списку в группе: 10;
Дата выдачи 20200901;
Курс: первый.";
```

In[ ]:= (\*Подготовка информации для встраивания\*)

```
binaryInfo = ToCharacterCode[rawInfo];
               |код символа
commonLengthInfo = IntegerDigits[binaryInfo, 2, 11];
               |цифры целого числа
stringInfoList = Flatten[commonLengthInfo];
               |уплостить
stringInfoString = StringJoin[IntegerString[stringInfoList]]
               |соединить с... |строковая запись целого числа
```


Out[ ]:= 1000001110110000110000100001110001000011110010000110101100001111011000011111010000110 \
010100001100001000011110110000111000100001101010000010000010000110100100001111101 \
00001110101000100001110000111100100001101011000011110110001000010100001100000000 \
111010000001000001000001011110000110000100010001111000011010110001000010100001111 \
011000011000010001001111000001000001000011101010000111101100001110001000011011010 \
000111010100001100000000011101100000001010100000111011000011111010000111100100001 \
101011000100000000000100000100001101001000011111010000111010100010000111000011110 \
01000011010110000111101100010000101000011000000000111010000001000000000110000000 \
00110000000001100100000011000000000110010000001100000000110001000001100100000011 \
010100000110011000001110110000000101010000100100100001100001000011110010000111000 \
100001110111000011100010001001111000001110100000010000010000011010100010000111000 \
100001010001000011100001101111000011111010000110010000001110110000000101010000011 \
000100001111001000100111100000111010000001000001000001100010000111011100010011001 \
000100111100000111011000000010101000001111010001000010100010001111000011010110001 \
000001100010000101000011001010000111110000001110100000010000010000010011100001101 \
011000011110110000111101100001100001000011010010001001100100001101011000011001010 \
000111000100010001110000011101100000001010100000100111000100000010001000011100001 \
11111100001111110000110000000001110100000010000010000010000000001011010000011000 \
100000110010100001111000000010110100000110010000001100000000011101100000001010100 \
00011101100001111101000011110010000110101100010000000000100000100001111111000011 \
11100000010000010001000001100001111110000111000100010000011000011101010001000011 \
000001000001000011001000000100000100001100111000100000010001000011100001111111000 \
011111110000110101000001110100000010000000000110001000001100000000011101100000001 \
010100000101001000011000010001000010100001100000000010000010000110010100010010111 \
0000110100100001100001000100011110000111000000001000000000110010000001100000000 \
110010000001100000000011000000000111001000001100000000011000100000111011000000010 \
101000001101010001000011100010000001000100000100000111010000001000001000011111110 \
0001101011000100000010000110010100010010111000011100100000101110

```

In[ ]:= (*Определение цветового канала для встраивания информации*)
(*Цветовой канал - зеленый*)
colorChannel = Mod[nNom, 3]
           |остаток от деления
pixel = {0, 0, 0}; pixel[[colorChannel + 1]] = 255;
Image[{{pixel}}, "Byte"]
           |байт

```

Out[ ]:= 1

Out[ ]:= 

```

In[ ]:= str = {};
Do[
  |оператор цикла
  AppendTo[str, FromCharacterCode[FromDigits[Partition[stringInfoList, 11][[i]], 2]]],
  |добавить в конце... |символ по его коду |число по ря... |разбиение на блоки
  {i, Length[stringInfoList] / 11}]; StringJoin[str]
           |длина |соединить строки

```

Out[ ]:= Наименование документа: Зачетная книжка;

Номер документа: 0020201253;

Фамилия: Кутузов;

Имя: Илья;

Отчество: Геннадьевич;

Группа: А-12м-20;

Номер по списку в группе: 10;

Дата выдачи 20200901;

Курс: первый.

```

In[ ]:= ProtectWatermark[img0_, strInfo0_] := Module[
  |программный модуль
  {img = img0, strInfo = strInfo0, randomSeed = 10, colorChannelCd = 2, blockSize = {4, 4}
  , waterMark
  , imgParts
  , imgPartsColorSeparated
  , storedDimensions
  , imgFlow
  , imgPartsData
  , imgRow
  , separatedColorRow
  , targetChannel
  , workRow
  , parityList
  , parityListToDo
  , randPixelId
  , blocks
  , combinedColorRow
  , newImgParts},
  (*Получение битового списка из входной строки*)

  waterMark = Flatten[IntegerDigits[ToCharacterCode[strInfo], 2, 11]];
           |упростить |шифры цифрового шифра |код символа

```

```

(*Добавление стоп символа из 11 нулей*)
waterMark = Flatten[Append[waterMark, IntegerDigits[0, 2, 11]]];
(*Получение Блоков изображения*)
imgParts = ImagePartition[img, blockSize];
(*Обмен строк со столбцами*)
imgParts = Transpose[imgParts];
(*Непрерывность пути*)
Do[imgParts[[i]] = Reverse[imgParts[[i]], 1], {i, 2, Length[imgParts], 2}];
imgRow = Flatten[imgParts];
separatedColorRow = {};
Do[AppendTo[separatedColorRow, ColorSeparate[imgRow[[i]]], {i, Length[imgRow]}];
(*Выделение канала в рабочую строку*)
workRow = {};
Do[AppendTo[workRow, ImageData[separatedColorRow[[i, colorChannelCd]], "Byte"],
{i, Length[separatedColorRow]}];
(*Получение четности блоков*)
parityList = {};
Do[AppendTo[parityList, Mod[Total[Flatten[workRow[[i]]], 2]],
{i, Length[waterMark]}];
(*Наложение сообщения на четность блоков для получения
списка блоков к изменению четности*)
parityListToDo = Mod[parityList + waterMark, 2];
(*Приведение четности блоков к требуемым значениям*)
Do[
randPixelId =
{RandomInteger[{1, blockSize[[1]]}, RandomInteger[{1, blockSize[[2]]}]}];
workRow[[i, randPixelId[[1]], randPixelId[[2]]]] =
BitXor[parityListToDo[[i]], workRow[[i, randPixelId[[1]], randPixelId[[2]]]]],
{i, Length[parityListToDo]}];
(*Сборка изображения*)
blocks = {};
Do[AppendTo[blocks, Image[workRow[[i]], "Byte"], {i, Length[workRow]}];
separatedColorRow[[All, colorChannelCd]] = blocks;

```

```

combinedColorRow = {};
Do[AppendTo[combinedColorRow, ColorCombine[separatedColorRow[[i]], "RGB"]],
  {i, Length[separatedColorRow]}];
newImgParts = Partition[combinedColorRow, Dimensions[imgParts][[2]]];
Do[newImgParts[[i]] = Reverse[newImgParts[[i]], 1],
  {i, 2, Length[newImgParts], 2}];
newImgParts = Transpose[newImgParts];
ImageAssemble[newImgParts]
]

```

```

In[ ]:= ReadWatermark[img0_] := Module[
    |программный модуль
    {img = img0, colorChannelCd = 2, blockSize = {4, 4}, randomSeed
    , strInfo
    , waterMark
    , imgParts
    , imgPartsData
    , imgRow
    , workRow
    },
    (*Получение битового списка из входной строки*)
    imgParts = ImagePartition[img, blockSize];
    |разбиение изображения
    imgParts = Transpose[imgParts];
    |транспозиция
    Do[imgParts[[i]] = Reverse[imgParts[[i]], 1], {i, 2, Length[imgParts], 2}];
    |оператор цикла |расположить в обратном порядке |длина
    imgRow = Flatten[imgParts];
    |уплостить
    workRow = {};

    Do[AppendTo[workRow, ImageData[
    |... |добавить в конец к |данные изображения
        ColorSeparate[imgRow[[i]]][[colorChannelCd]], "Byte"]], {i, Length[imgRow]}];
    |разделить цветовые ка... |байт |длина
    waterMark = {};

    Do[AppendTo[waterMark, Mod[Total[Flatten[workRow[[i]]]], 2]],
    |... |добавить в конец к |ос... |сумм... |уплостить
        {i, Length[workRow]}];
    |длина
    waterMark = Partition[waterMark, 11];
    |разбиение на блоки
    waterMark =
        Take[waterMark, Flatten[Position[waterMark, IntegerDigits[0, 2, 11]]][[1]] - 1];
    |изв... |уплостить |позиция по образцу |цифры целого числа

    strInfo = {};
    Do[AppendTo[strInfo, FromDigits[waterMark[[i]], 2]], {i, Length[waterMark]}];
    |добавить в конец к |число по ряду цифр |длина
    FromCharCode[strInfo]
    |символ по его коду
]

```

```

In[ ]:= protectedImage = ProtectWatermark[image, rawInfo]
filePath =
  Export["D:\\GitHub Repos\\stud\\mag\\Sem9\\KP ЦТЗИ\\МУ\\protected_image.bmp",
    Экспорт... Дифференцировать
    protectedImage];
ReadWatermark[Import[filePath]]
Импорт

```

Out[ ]:=



```

Out[ ]:= Наименование документа: Зачетная книжка;
Номер документа: 0020201253;
Фамилия: Кутузов;
Имя: Илья;
Отчество: Геннадьевич;
Группа: А-12м-20;
Номер по списку в группе: 10;
Дата выдачи 20200901;
Курс: первый.

```

```

In[ ]:= p = 13 642 163;
a = 0;
b = 23;
c = 66;
P = {6 821 082, 5 569 902};
rank = 13 645 001;
secretKey = RandomInteger[{1, rank - 1}]
      |случайное целое число
publicKey = EllipticMult[p, a, b, c, P, secretKey]
signature = ECDSAGeneration[p, a, b, c + 59, P, rank, filePath, secretKey]
ECDSAVerification[p, a, b, c + 59, P, rank, filePath, publicKey, signature]
ECDSAVerification[p, a, b, c + 59, P, rank,
  "D:\\GitHub Repos\\stud\\mag\\Sem9\\KP ЦТЗИ\\MY\\240x320.bmp", publicKey, signature]
      |дифференцировать

```

```
Out[ ]:= 3 181 056
```

```
Out[ ]:= {6 030 880, 868 506}
```

```
Out[ ]:= {13 177 322, 6 231 937}
```

```
True
```

```
False
```