In[ ]:=
```
(*Сложение двух точек эллиптической кривой*)
EllipticAdd[p_, a_, b_, c_, P_List, Q_List] := Module[{lam, x3, y3, P3},
                                                 программный модуль
  Which[
  условный оператор с множественными ветвями
   P == {O}, Q,
        О большое
   Q == {O}, P,
        О большое
   P[[1]] ≠ Q[[1]],
       lam = Mod[(Q[[2]] - P[[2]]) PowerMod[Q[[1]] - P[[1]], p - 2, p], p];
             остаток от деления          степень по модулю
       x3 = Mod[lam^2 - a - P[[1]] - Q[[1]], p];
            остаток от деления
       y3 = Mod[-(lam (x3 - P[[1]]) + P[[2]]), p];
            остаток от деления
       {x3, y3},
   (P == Q) ∧ (P[[2]] == 0), {O},
                             О большое
   (P == Q) ∧ (P ≠ {O}),
                  О большое
       lam = Mod[(3 * P[[1]]^2 + 2 a * P[[1]] + b) PowerMod[2 P[[2]], p - 2, p], p];
             остаток от деления                    степень по модулю
       x3 = Mod[lam^2 - a - P[[1]] - Q[[1]], p];
            остаток от деления
       y3 = Mod[-(lam (x3 - P[[1]]) + P[[2]]), p];
            остаток от деления
       {x3, y3},
   (P[[1]] == Q[[1]]) ∧ (P[[2]] ≠ Q[[2]]), {O}
                                           О большое
  ]
 ]
```

In[ ]:=
```
(*Умножение точки эллиптической кривой на константу*)
EllipticMult[p0_, a0_, b0_, c0_, pointP0_, n0_] := Module[
                                                   программный модуль
  {pointP = pointP0, n = n0, p = p0, a = a0, b = b0, c = c0, pointQ = {0, 0}, binN},
  binN = IntegerDigits[n, 2];
         цифры целого числа
  Do[
  оператор цикла
   If[binN[[i]] == 0,
   условный оператор
    pointQ = EllipticAdd[p, a, b, c, pointQ, pointQ]
    , pointQ = EllipticAdd[p, a, b, c, EllipticAdd[
        p, a, b, c, pointQ, pointQ
       ], pointP]
   ]
   , {i, 1, Length[binN]}];
           длина
  pointQ]
```

In[ ]:=
```
(*Генерация подписи ECDSA*)
ECDSAGeneration[p0_, a0_, b0_, c0_,
  pointP0_, rank0_, textFile0_, secretKey0_] := Module[
                                                 программный модуль
  {p = p0, a = a0, b = b0, ci = c0, pointP = pointP0, rank = rank0, textFile = textFile0,
   secretKey = secretKey0, k, R, r, e, d, c}, k = RandomInteger[{2, rank - 2}];
                                                 случайное целое число
  R = EllipticMult[p, a, b, ci, pointP, k];
  r = Mod[R[[1]], rank];
      остаток от деления
  e = FileHash[textFile, "MD5"];
      хэш-сумма файла
  d = PowerMod[k, -1, rank];
      степень по модулю
  c = Mod[d * (e + secretKey * r), rank];
      остаток от деления
  {r, c}]
```

In[ ]:=
```
(*Проверка подписи*)
ECDSAVerification[p0_, a0_, b0_, c0_, pointP0_,
  rank0_, textFile0_, publicKey0_, signature0_] := Module[
                                                   программный модуль
  {p = p0, a = a0, b = b0, ci = c0, pointP = pointP0, rank = rank0, textFile = textFile0,
   publicKey = publicKey0, signature = signature0, W, hash, U1, U2, U1P, U2rank, R1},
  If[(1 < signature[[1]] < rank - 1) && (1 < signature[[2]] < rank - 1),
  условный оператор
    W = PowerMod[signature[[2]], -1, rank];
        степень по модулю
    hash = FileHash[textFile, "MD5"];
           хэш-сумма файла
    U1 = Mod[W * hash, rank];
         остаток от деления
    U2 = Mod[W * signature[[1]], rank];
         остаток от деления
    U1P = EllipticMult[p, a, b, ci, pointP, U1];
    U2rank = EllipticMult[p, a, b, ci, publicKey, U2];
    R1 = EllipticAdd[p, a, b, ci, U1P, U2rank];
    If[R1[[1]] == signature[[1]], Print[True], Print[False]]
    условный оператор                печа… истина печа… ложь
    , Print[-1]];]
      печатать
```