Generates pseudorandom espresso

This step consists in the generation through a pre-defined algorithm of mathematical expressions based on a given public key according to the desired private key position.

For example:

Public key: **(65 bytes)**

04CDDCE816EF153B8E8EADECE2A6489481B7332FD99A4718066C40B1B688F6A08828
241A5CC0A97E2C916C2EC610838325FB49403BB3ED352BB4574776FEC5E3B3, **ou seja:**

X = CDDCE816EF153B8E8EADECE2A6489481B7332FD99A4718066C40B1B688F6A088

Y = 28241A5CC0A97E2C916C2EC610838325FB49403BB3ED352BB4574776FEC5E3B3

Private key: **(32 bytes)**

375D75D0A1188016E9DE9395BFF6334BD3FDCEB5884766CE87454DB30612D936

Get the value of the private key at position 0 (0x37):

After **513** attempts, the following expression is generated:

(Y[12] ^ Y[12]) * (Y[18] ^ Y[26]) + (Y[6] | Y[14]) % (X[13] | X[31]), substituting os values and calculating the result of the expression, we get **55** or **0x37 (correct value of the private key at position 0) as a result.**

Now, let's try to generate the expression that leads to the value of the private key at position **1:**

After **270** attempts, the following expression is generated:

(X[18] % Y[23]) | (~Y[10] * X[0]), substituting the values and calculating the result of the expression, result **93** or **0x5D (correct value of private key at position 1).**

The algorithm has **3 input values:** Public key **(X and Y),** Position of the private key to be discovered **(0-31)** and attempt **(0-).**

Using the public key above as an example (knowing the correct value of the private key), we will have as a result: [513, 270, 151, 500, 546, 176, 305, 823, 369, 1310,

751, 792, 30, 61, 464, 3419, 1091, 84, 780, 52, 313, 112, 447, 52, 265, 10, 333, 775, 622, 2, 92, 2], each value refers to the attempt in which the expression that resulted in the correct value of the private key in its respective position was generated.