Elliptic SSS scheme

Zuev Egor

Higher School of Economics

zyev.egor@gmail.com

Abstract

This document proposes a standard for running SSS scheme over the elliptic curve secp256k1.

Keywords: SSS, secp256k1, Elliptic curve.

Motivation

SSS scheme is used worldwide to share the secret between several parties. SSS scheme can be adopted and used by bitcoin applications for delegating the control of the certain private key. The adopted SSS (let's call it ESSS – elliptic SSS) gives extra level of security by delegating the private keys only to known public keys (using secp256k1 standard). The key features of this approach are:

1) The $privKey1 = split(m, n)$, where m – amount of peers, n – required shares for restoration of $privKey1$. To restore the $privKey1$, n of peers should sign their share and provide back the signature. As a result, if the share has been signed by wrong private key - the secret can't be restored.
2) This approach can be used as M-of-N multisig on application level
3) Compact: only needed n shares to restore the secret, without any intermediate data.
4) No extra rounds: the secret split happens in one round (if the peer already knows all public keys), the secret restoration happens in one round as well

Description

The original SSS scheme is based on polynomial interpolation, which is an algebraic method of estimating unknown values in a gap between two known data points — without needing to know anything about what is on either side of those points.

SSS encodes the secret into a polynomial, then split it and distribute across peers. The key property of SSS, that in order to restore the secret, you have to provide the N of M shares (also known as threshold). N shares gives enough information to guess the point on the curve.

Preparation round

During the preparation round shares should be build from provided secret. First coefficients have to be generated (there are known as $a_i, where\ i - is\ index$). The amount of a_i is equal to n (threshold) provided. Important note, that $a_0 = secret$. For instance, let's assume our $secret = 1234$, m = 5, n = 3 (3 shares required to restore the secret of 5 generated):

1) The primes should be defined. All coefficients should be in the range: 0 < a_i < prime (including a_0 – which is secret). Let's assume, our prime = 2034
2) Each a_i = random(0;prime). Let's assume, we've generated the following values: $a_1 = 345, a_2 = 543$
3) Our function will have the form: $f(x) = a_0 + a_1 * x + a_2 * x^2\ mod\ prime$ or $f(x) = 1234 + 345 * x + 543 * x^2$ mod 2034
4) In original SSS, each share has $x = index$, for instance for the first share $x = 1$. In our case: $f(1) = 88, f(2) = 28, f(3) = 1054$ and so on. These points are coordinates on defined curve.
5) Each peer should get the share and its index

Restoration

The restoration requires n-of-m shares to be provided. The key point for us is where $x = 0$, because $f(0) = a_0$, where $a_0 = secret$. To find out a_0 the Lagrange basis polynomials can be used:

$$\sum_{j=0}^{n} y_j * l_j(x)\ , where\ l_j(x) = \prod_{\substack{0 \le v \le n \\ j \ne v}} \frac{x - x_v}{x_j - x_v}$$

$$f(0) = 88 * \left(\frac{0-2}{1-2} * \frac{0-3}{1-3}\right) + 28 * \left(\frac{0-1}{2-1} * \frac{0-3}{2-3}\right) + 1054 * \left(\frac{0-1}{3-1} * \frac{0-2}{3-2}\right)$$

$$f(0) = 88 * 3 + 28 * (-3) + 1054 * 1 = 264 - 84 + 1054 = 1234$$

ECC extended version of SSS

Preparation round

Let's assume that $privateKey_0$should be shared between 3 peers. Each of peers has its own keypair (private and public keys generated by secp256k1 standard): $(publicKey_1; privateKey_1)$, $(publicKey_2; privateKey_2)$, $(publicKey_3; privateKey_3)$. In the following example, m = 3 and n = 2 (so only 2 any peers required to restore the $privateKey_0$). The rest of parameters and properties are:

1) The prime number is static and is equal to the order of the group of the curve (n parameter of the curve): $fffffffffffffffffffffffffffffebaaedce6af48a03bbfd25e8cd0364141$
2) $publicKey_i = privateKey_i * G$, where G – is generator (point) and equal to: (79be667ef9dcbbac55a06295ce870b07029bfcdb2dce28d959f2815b16f81798, 483ada7726a3c4655da4fbfc0e1108a8fd17b448a68554199c47d08ffb10d4b8)
3) Coefficients (a_i) and $privateKey_0$ should be less than prime (the same property as in original SSS)
4) Each share represents the coordinate (x, y), where y – is a share and x – is the order of the share.

The flow looks like so:

1) The peer, who shares $privateKey_0$ receives $publicKey_1$, $publicKey_2$, $publicKey_3$
2) For each $publicKey_i$ peer generates a special coefficient $xCoef_i = random(0; prime)$
3) Peer generates $a_i = random(0; prime)$, just like in original flow
4) The formula looks like in original flow: $f(x) = a_0 + a_1 * x + a_2 * x^2$ mod prime, but $x_i = xCoef_i * publicKey_i$
5) The peer shares with $xCoef_i$ and $share_i$. Each peer should receive his $xCoef_i$ and $share_i$ only

Restoration

The restoration process looks similar, however, have some additional steps:

1) Each owner of $publicKey_i$ should create a signature: $sig_i = xCoef_i * privateKey_i$
2) The peers should exchange the signatures and shares (2 of 3 in the following example)
3) To use original SSS restoration procedure, the calculation of x_i has to be done for each provided share: $x_i = sig_i * G$. The proof: $x_i = sig_i * G = xCoef_1 * privateKey_i * G = xCoef_i * publicKey_i$
4) The last step is to perform calculation, with computed x_i and provided $share_i$

Properties

Based on this solution, several additional properties are present:

1) Only the owner of $privateKey_i$ can sign $xCoef_i$
2) Each signature can be validated before private key restoration: $sig_i * G = xCoef_i * publicKey_i$