FROST

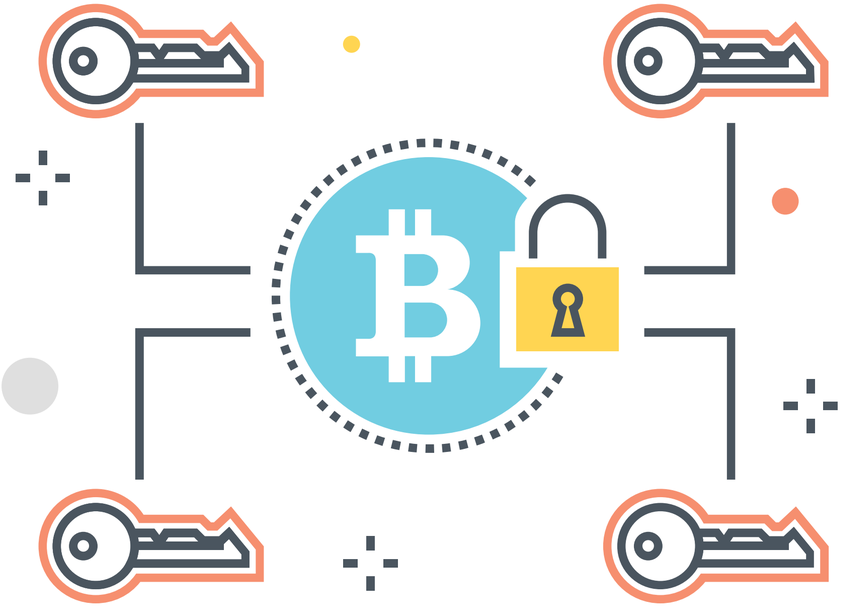
Flexible Round-Optimized Schnorr Threshold Signatures

Jesse Posner

Multi-Party Signatures

Overview

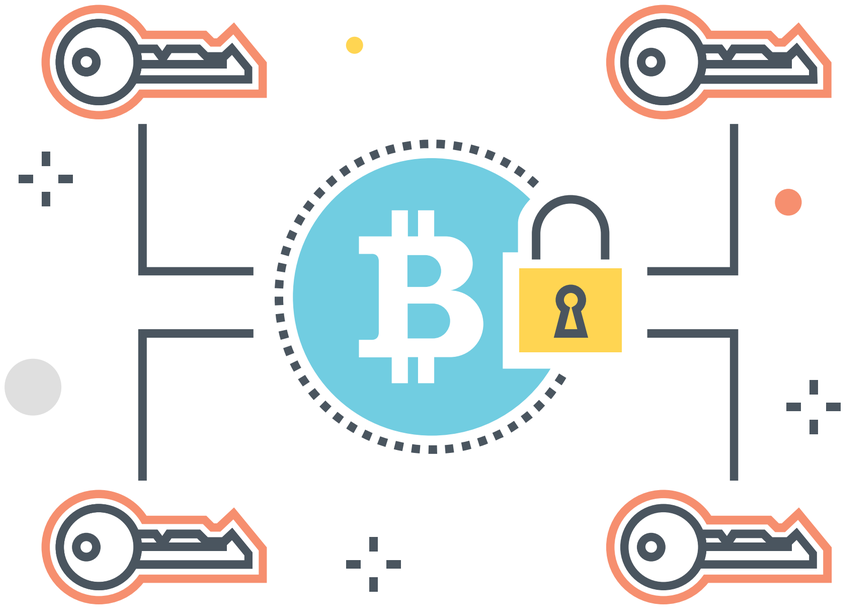
- Benefits
  - segregating access
    - joint accounts
    - escrow
    - role seperation
  - security
    - no single point of failure



Multi-Party Signatures

Scripts and Signatures

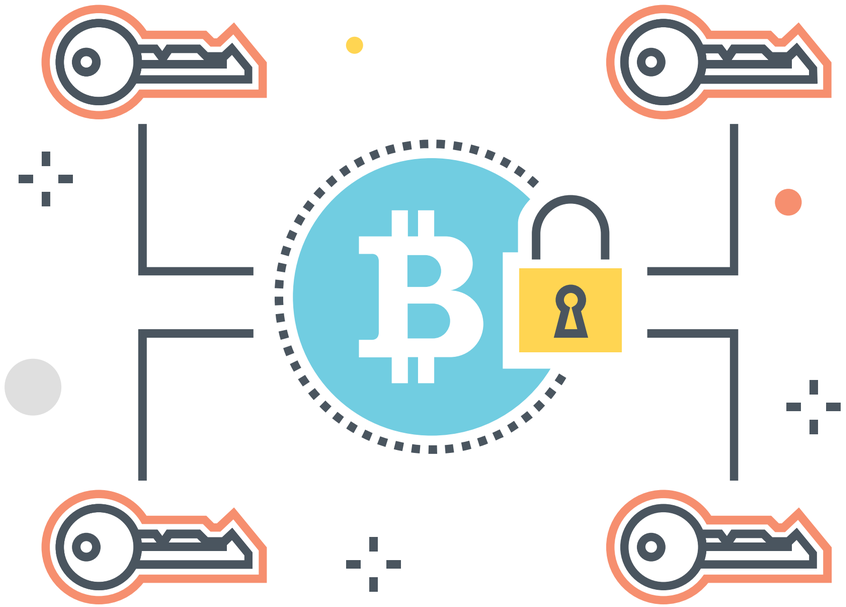
- On-Chain Script Verification
  - information about scheme leaks when output is spent
  - explicit verification data must be stored
- Off-Chain Signature Construction
  - indistinguishable from a single-party signature
  - only a single signature must be stored for verification



Multi-Party Signatures

ECDSA and Schnorr

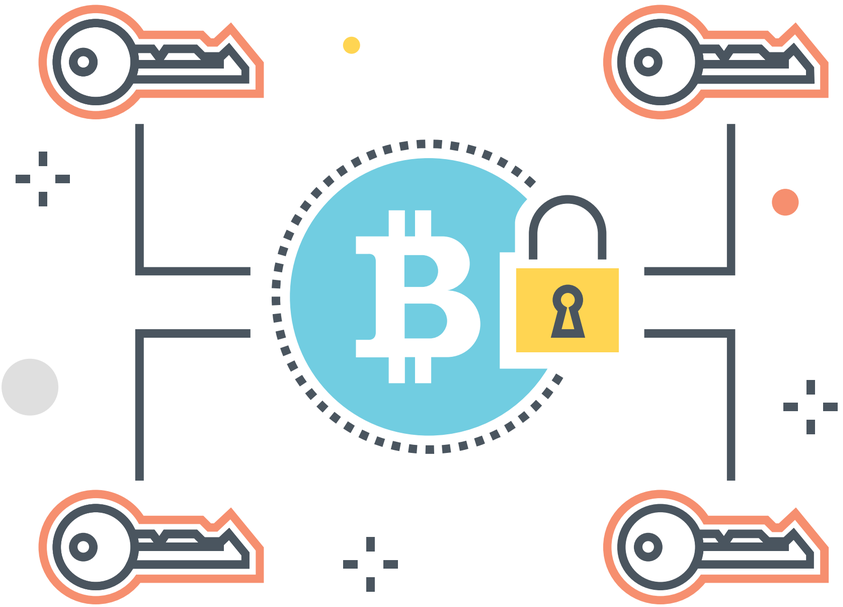
- ECDSA
  - non-linear signature
  - complex protocols with new cryptographic assumptions and large number of rounds
- Schnorr
  - linear signature
  - simpler protocols with no new assumptions
  - requires Taproot



Multi-Party Signatures

Multi-Signatures and Threshold Signatures

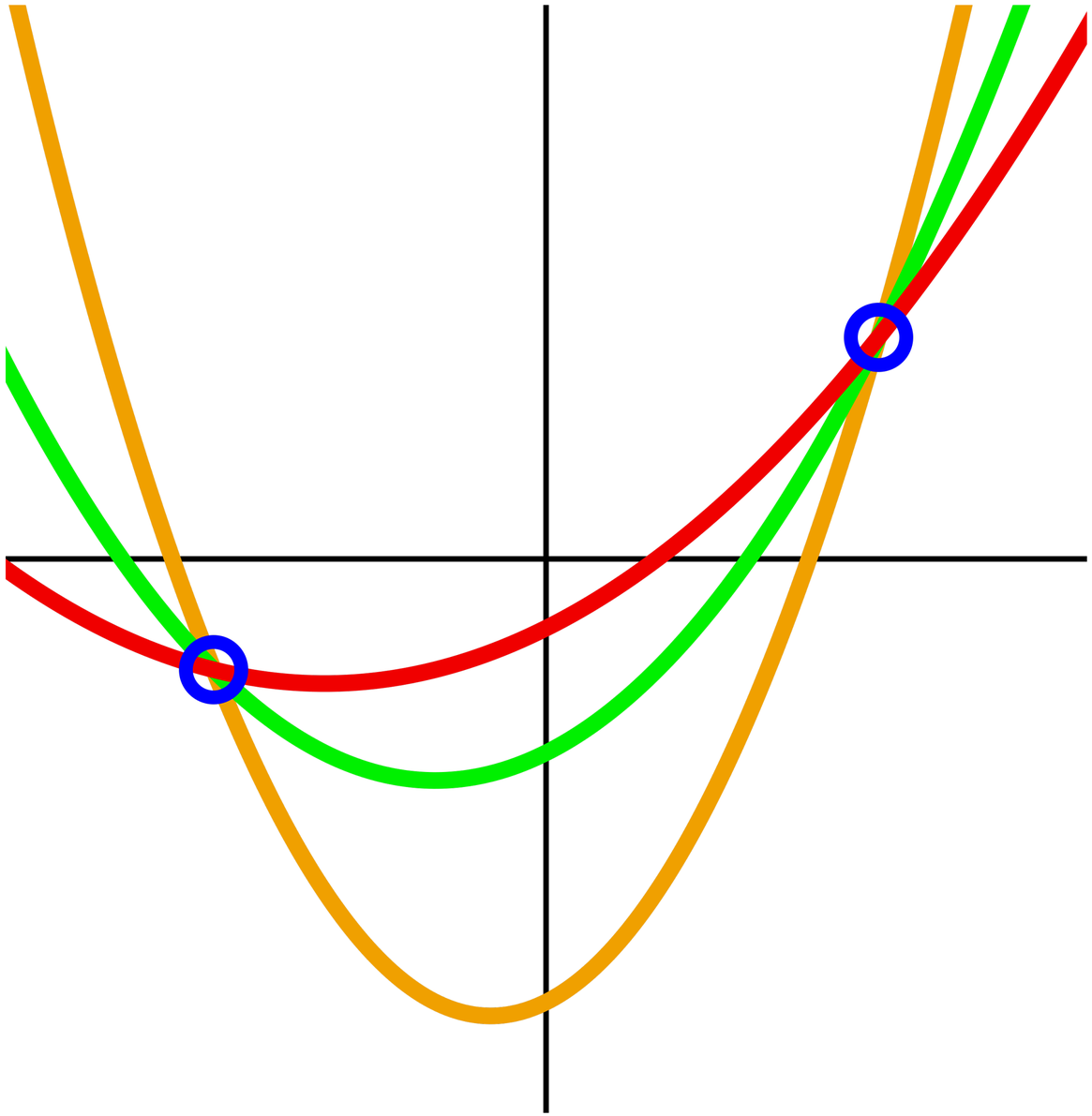
- Multi-Signatures
  - n-of-n
  - non-interactive key generation
  - MuSig/MuSig2
- Threshold Signatures
    - t-of-n
    - interactive key generation
    - FROST



Schnorr Threshold Signatures

Shamir Secret Sharing

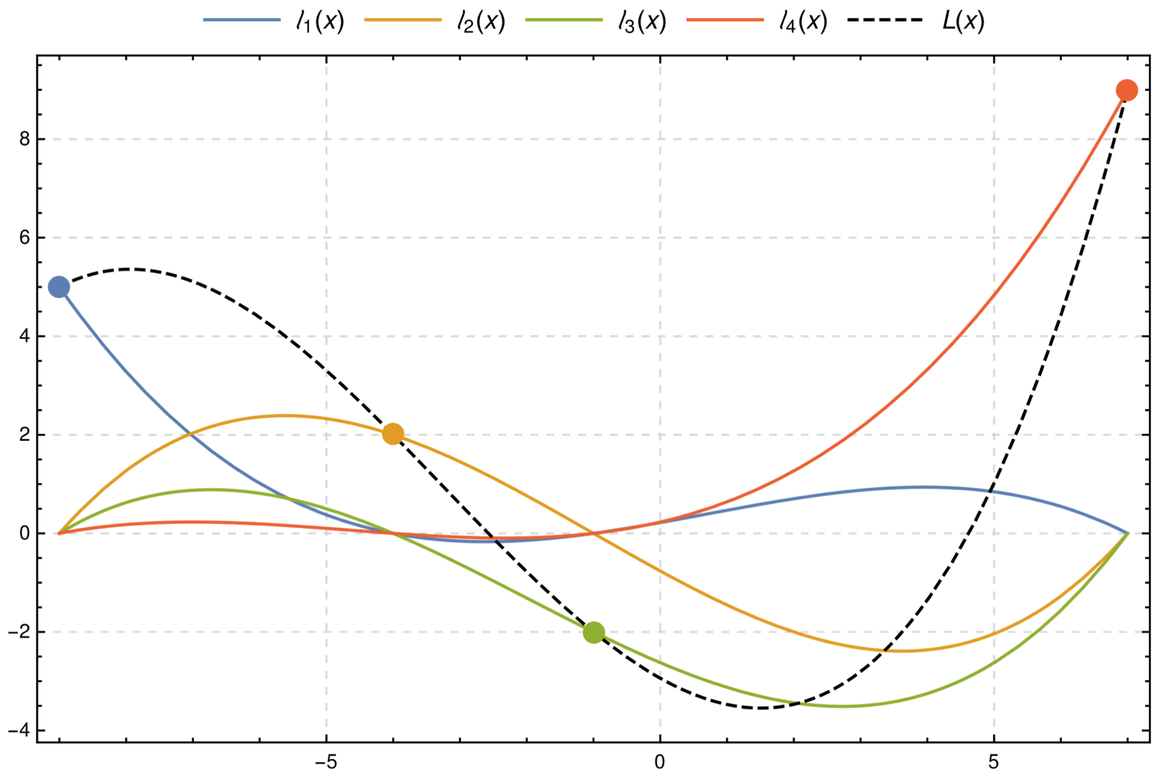
- 2 points determine a unique line (1 degree polynomial: y = Ax^0 + Bx^1)
- 3 point determine a unique parabola (2 degree polynomial: y = Ax^0 + Bx^1 + Cx^2)
- every set of n points uniquely determine a polynomial with degree n - 1



Shamir Secret Sharing

Lagrange Polynomial Interpolation

- for each point, derive a linear equation in which the x value of that point returns 1 and the x value of any other point returns 0 (a Lagrange polynomial)



$$\ell_j(x) := \prod_{\substack{0 \le m \le k \\ m \neq j}} \frac{x - x_m}{x_j - x_m}$$

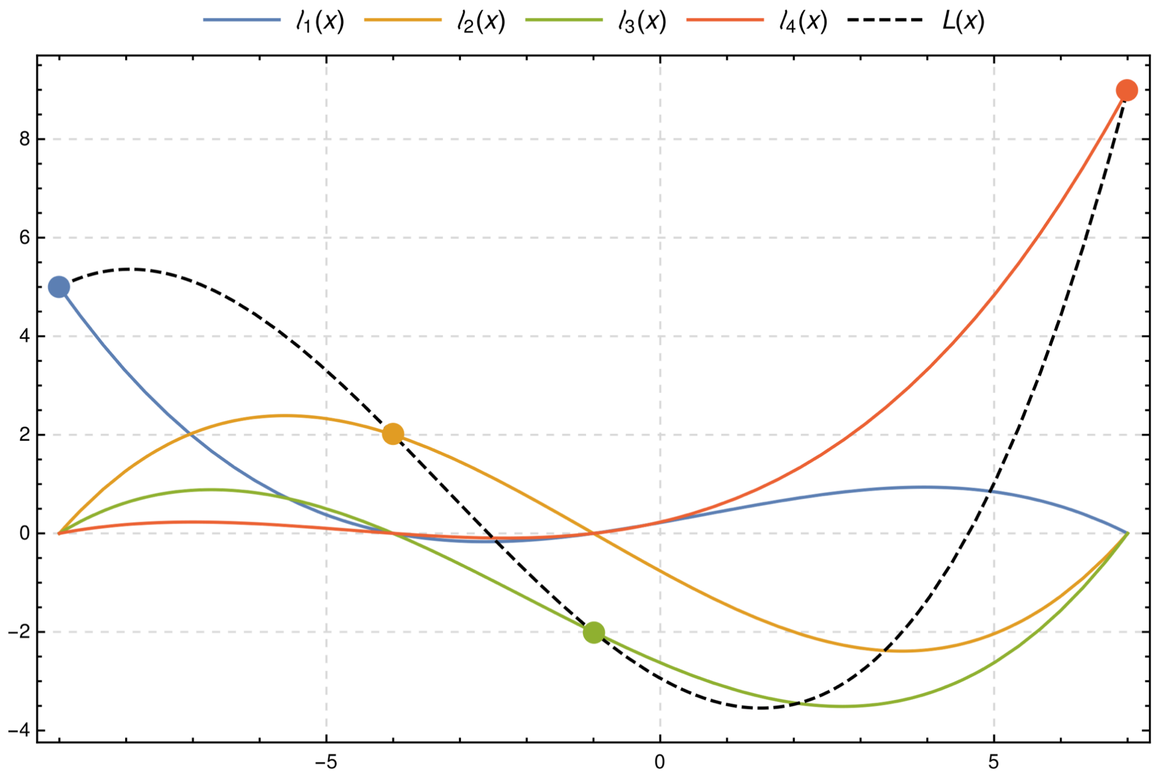
$$\ell_j(x_i) = \prod_{m \neq j} \frac{x_i - x_m}{x_j - x_m} = \frac{(x_i - x_0)}{(x_j - x_0)} \cdots \frac{(x_i - x_i)}{(x_j - x_i)} \cdots \frac{(x_i - x_k)}{(x_j - x_k)} = 0$$

$$\ell_j(x_j) := \prod_{m \neq j} \frac{x_j - x_m}{x_j - x_m} = 1$$

Shamir Secret Sharing

Lagrange Polynomial Interpolation

- the interpolated polynomial is the sum of each Lagrange polynomial multiplied by the respective y values for each point
- for each point, this polynomial will return the y value when provided the x value (because each Lagrange polynomial will return 0 when another point is provided, and 1 when the same point is provided)



$$\ell_j(x_i) = \prod_{m \neq j} \frac{x_i - x_m}{x_j - x_m} = \frac{(x_i - x_0)}{(x_j - x_0)} \cdots \frac{(x_i - x_i)}{(x_j - x_i)} \cdots \frac{(x_i - x_k)}{(x_j - x_k)} = 0$$

$$\ell_j(x_j) := \prod_{m \neq j} \frac{x_j - x_m}{x_j - x_m} = 1$$

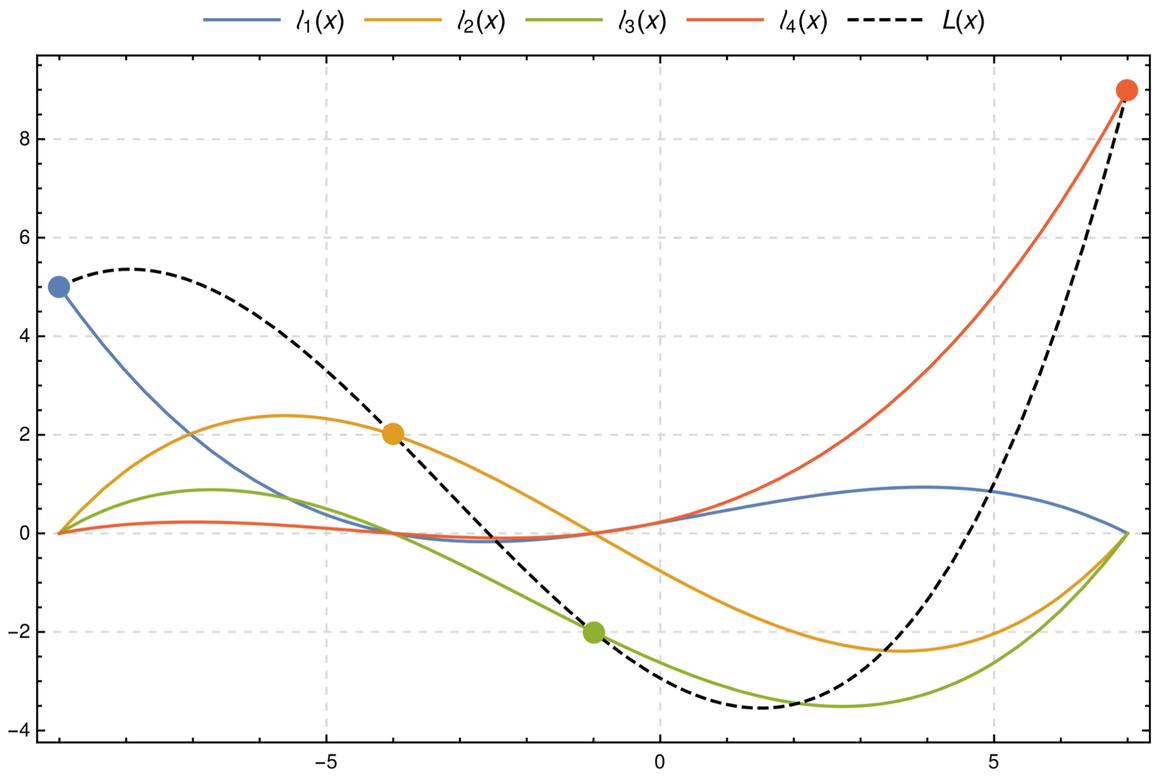
$$L(x) := \sum_{j=0}^{k} y_j \ell_j(x)$$

Shamir Secret Sharing

Lagrange Polynomial Interpolation

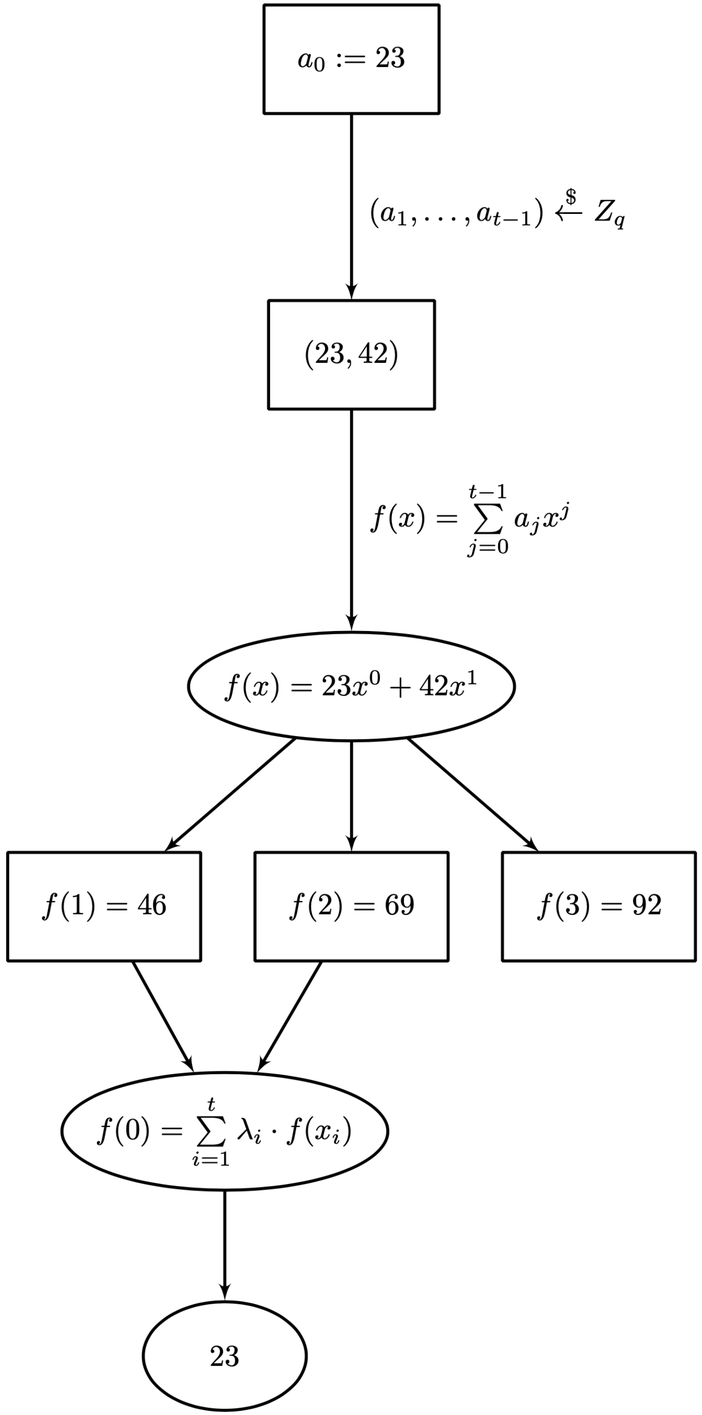
- if we are only interested in interpolating the first coefficient, rather than the entire polynomial, we can compute a Lagrange coefficient for each point, and the sum of a threshold of y values multiplied by their respective Lagrange coefficients will derive the first coefficient

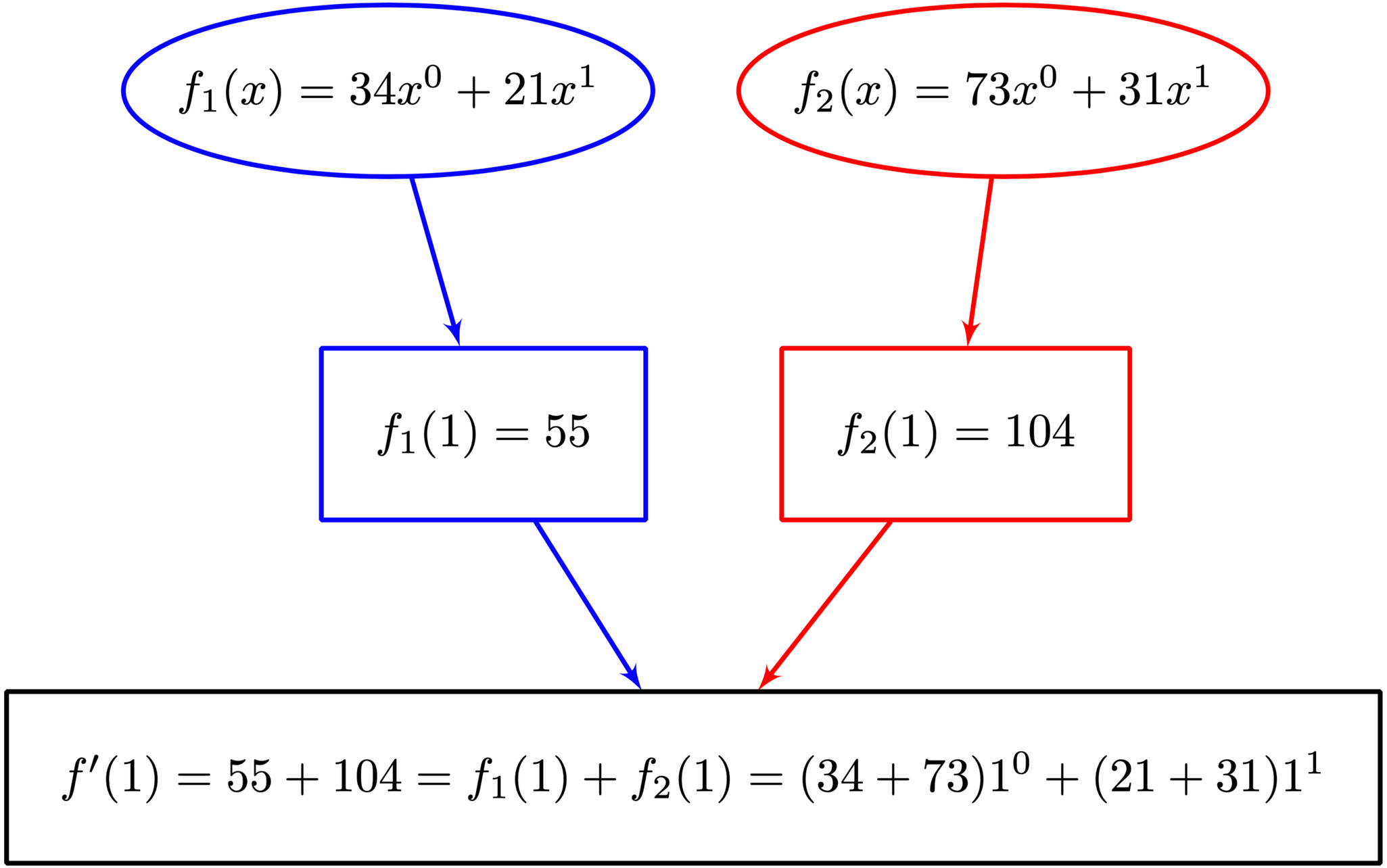
- this gives a coefficient for each share that can be computed by reference to the index (x value) for each share

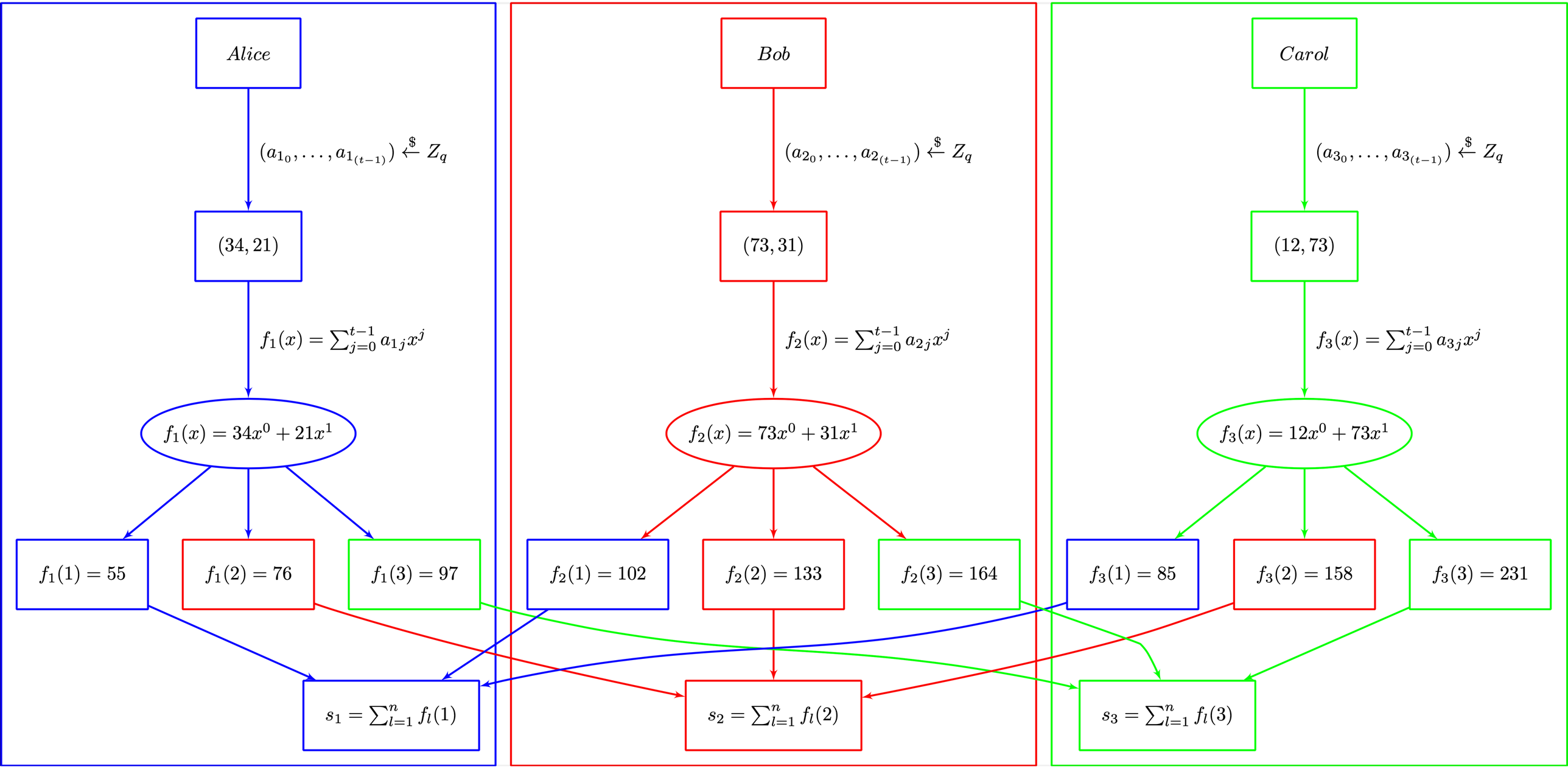


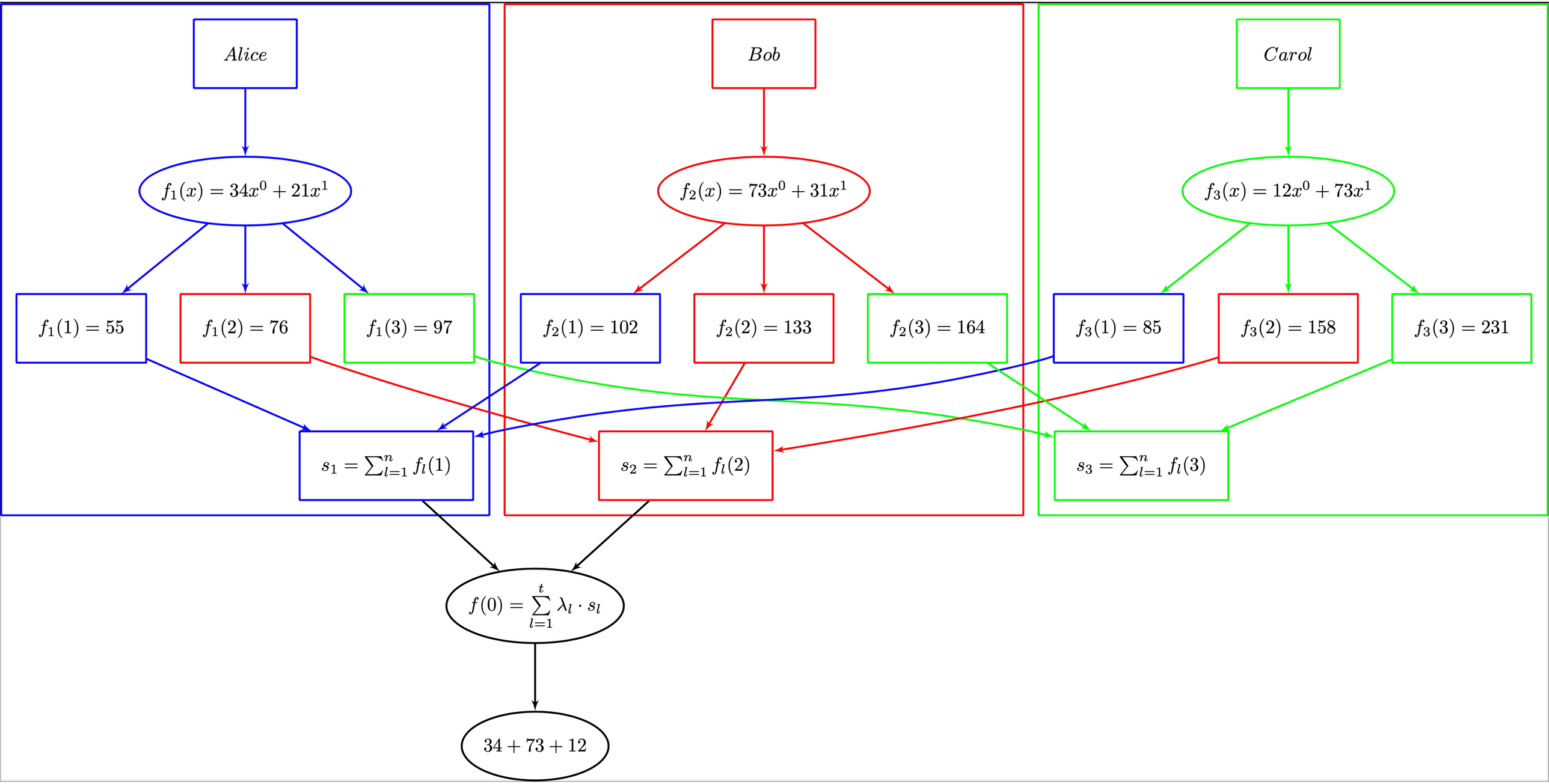
$$\ell_j(x) := \prod_{\substack{0 \le m \le k \\ m \ne j}} \frac{x - x_m}{x_j - x_m}$$

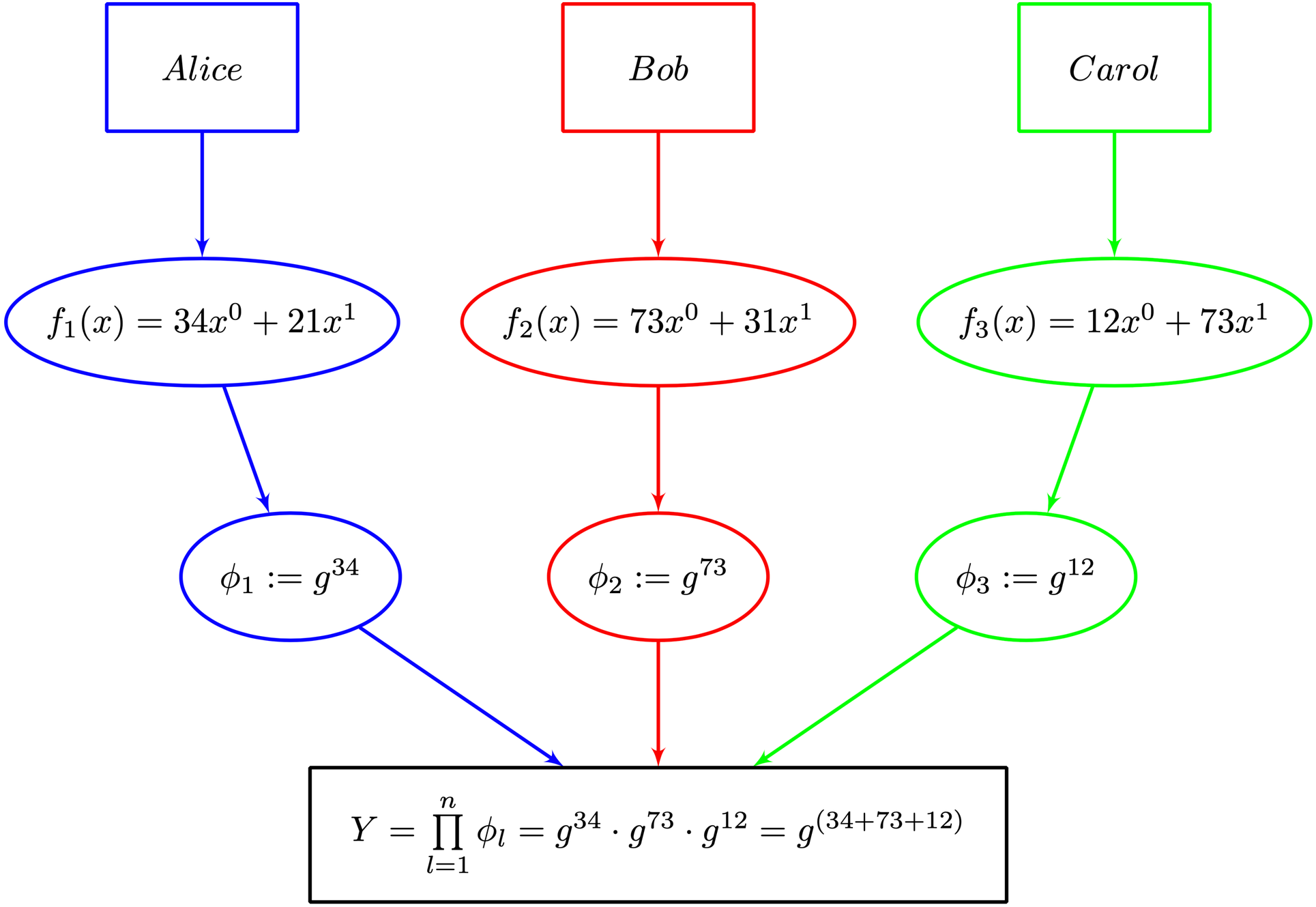
$$f(0) = \sum_{j=0}^{k-1} y_j \prod_{\substack{m=0 \\ m \ne j}}^{k-1} \frac{x_m}{x_m - x_j}$$


$a_0 := 23$
$(a_1, \ldots, a_{t-1}) \xleftarrow{\$} Z_q$
$(23, 42)$
$f(x) = \sum_{j=0}^{t-1} a_j x^j$
$f(x) = 23x^0 + 42x^1$
$f(1) = 46$
$f(2) = 69$
$f(3) = 92$
$f(0) = \sum_{i=1}^{t} \lambda_i \cdot f(x_i)$
23


$f_1(x) = 34x^0 + 21x^1$
$f_2(x) = 73x^0 + 31x^1$
$f_1(1) = 55$
$f_2(1) = 104$
$f'(1) = 55 + 104 = f_1(1) + f_2(1) = (34 + 73)1^0 + (21 + 31)1^1$

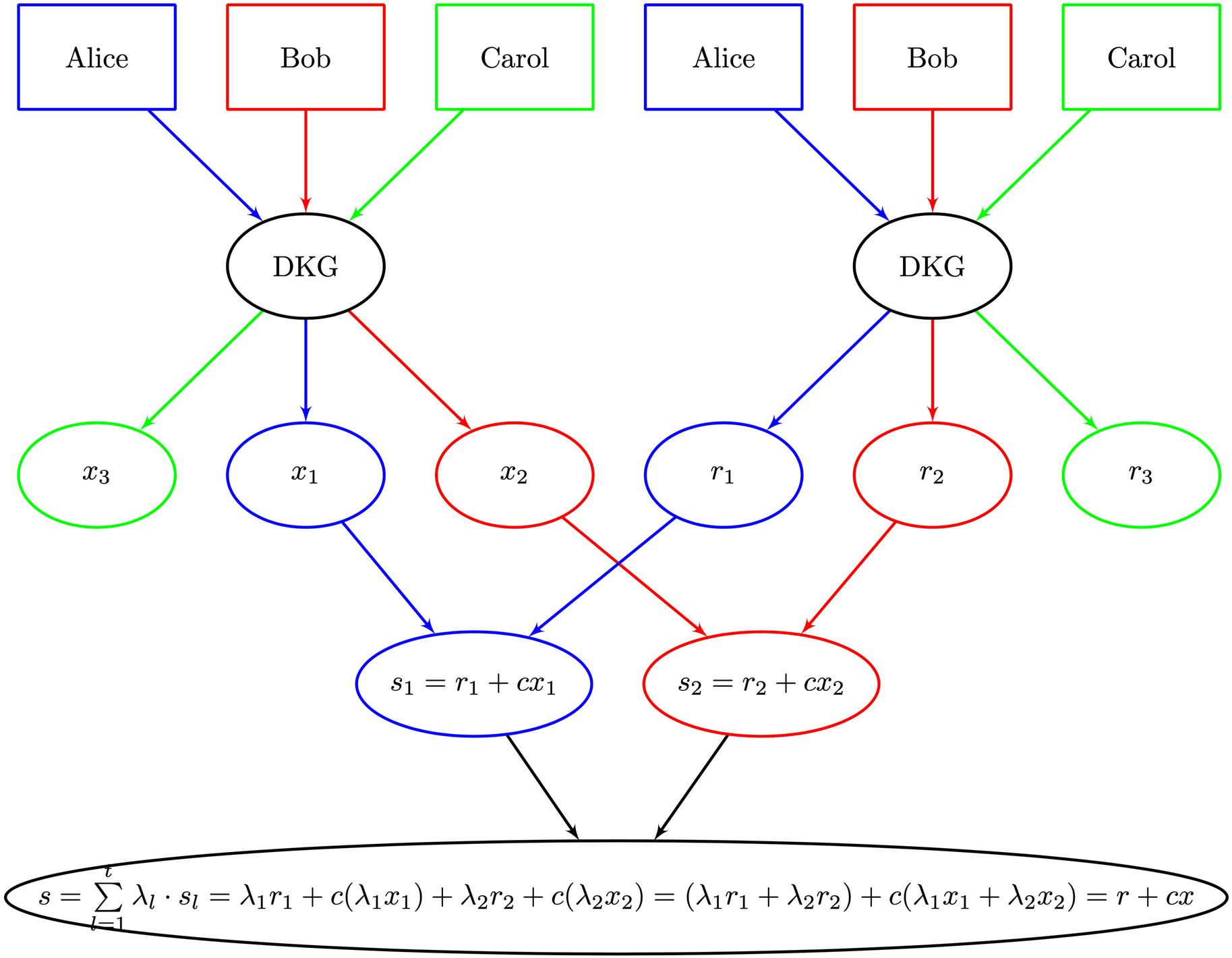

Alice
$(a_{1_0}, \ldots, a_{1_{(t-1)}}) \xleftarrow{\$} Z_q$
$(34, 21)$
$f_1(x) = \sum_{j=0}^{t-1} a_{1j} x^j$
$f_1(x) = 34x^0 + 21x^1$
$f_1(1) = 55$
$f_1(2) = 76$
$f_1(3) = 97$
$s_1 = \sum_{l=1}^{n} f_l(1)$
Bob
$(a_{2_0}, \ldots, a_{2_{(t-1)}}) \xleftarrow{\$} Z_q$
$(73, 31)$
$f_2(x) = \sum_{j=0}^{t-1} a_{2j} x^j$
$f_2(x) = 73x^0 + 31x^1$
$f_2(1) = 102$
$f_2(2) = 133$
$f_2(3) = 164$
$s_2 = \sum_{l=1}^{n} f_l(2)$
Carol
$(a_{3_0}, \ldots, a_{3_{(t-1)}}) \xleftarrow{\$} Z_q$
$(12, 73)$
$f_3(x) = \sum_{j=0}^{t-1} a_{3j} x^j$
$f_3(x) = 12x^0 + 73x^1$
$f_3(1) = 85$
$f_3(2) = 158$
$f_3(3) = 231$
$s_3 = \sum_{l=1}^{n} f_l(3)$


Alice
Bob
Carol
$f_1(x) = 34x^0 + 21x^1$
$f_2(x) = 73x^0 + 31x^1$
$f_3(x) = 12x^0 + 73x^1$
$f_1(1) = 55$
$f_1(2) = 76$
$f_1(3) = 97$
$f_2(1) = 102$
$f_2(2) = 133$
$f_2(3) = 164$
$f_3(1) = 85$
$f_3(2) = 158$
$f_3(3) = 231$
$s_1 = \sum_{l=1}^{n} f_l(1)$
$s_2 = \sum_{l=1}^{n} f_l(2)$
$s_3 = \sum_{l=1}^{n} f_l(3)$
$f(0) = \sum_{l=1}^{t} \lambda_l \cdot s_l$
$34 + 73 + 12$


Alice
Bob
Carol
$f_1(x) = 34x^0 + 21x^1$
$f_2(x) = 73x^0 + 31x^1$
$f_3(x) = 12x^0 + 73x^1$
$\phi_1 := g^{34}$
$\phi_2 := g^{73}$
$\phi_3 := g^{12}$
$Y = \prod_{l=1}^{n} \phi_l = g^{34} \cdot g^{73} \cdot g^{12} = g^{(34+73+12)}$

Schnorr Signature

$$s = r + cx$$


Alice
Bob
Carol
Alice
Bob
Carol
DKG
DKG
x_3
x_1
x_2
r_1
r_2
r_3
$s_1 = r_1 + cx_1$
$s_2 = r_2 + cx_2$
$s = \sum_{l=1}^{t} \lambda_l \cdot s_l = \lambda_1 r_1 + c(\lambda_1 x_1) + \lambda_2 r_2 + c(\lambda_2 x_2) = (\lambda_1 r_1 + \lambda_2 r_2) + c(\lambda_1 x_1 + \lambda_2 x_2) = r + cx$