Elliptic Curve Fast Fourier Transform

- FFT: efficient algorithms on polynomials. $O(n \log n)$ for:
  - **EXTEND** (interpolation)
  - multiplication
  - Convert: evaluation table $\leftrightarrow$ coefficient representation

- FFT: efficient algorithms on polynomials. $O(n \log n)$ for:
    - **EXTEND** (interpolation)
    - multiplication
    - Convert: evaluation table $\leftrightarrow$ coefficient representation

- **Problem: FFT requires FFT-friendly field** ($\mathbb{F}^\times$ containing size 2^k subgroup)
    - Not FFT-friendly:
        - secp256k1 (for ECDSA)
        - BN254 (pairing)
    - Use non-native field arithmetic? ~100x overhead.

- FFT: efficient algorithms on polynomials. $O(n \log n)$ for:
  - **EXTEND** (interpolation)
  - multiplication
  - Convert: evaluation table $\leftrightarrow$ coefficient representation

- **Problem: FFT requires FFT-friendly field** ($\mathbb{F}^\times$ containing size 2^k subgroup)
  - Not FFT-friendly:
    - secp256k1 (for ECDSA)
    - BN254 (pairing)
  - Use non-native field arithmetic? ~100x overhead.

- **Solution:** Replace $\mathbb{F}_q^\times$ with an **elliptic curve group** over $\mathbb{F}_q$
  - Theorem: Can always find size $2^k \simeq 2\sqrt{q}$ subgroup!

How fast is ECFFT?

Algorithm	Description	Runtime
ENTER	Coefficients to evaluations (fft analogue)	$\mathcal{O}(n \log^2 n)$
EXIT	Evaluations to coefficients (ifft analogue)	$\mathcal{O}(n \log^2 n)$
DEGREE	Computes a polynomial's degree	$\mathcal{O}(n \log n)$
EXTEND	Extends evaluations from one set to another	$\mathcal{O}(n \log n)$
MEXTEND	EXTEND for special monic polynomials	$\mathcal{O}(n \log n)$
MOD	Calculates the remainder of polynomial division	$\mathcal{O}(n \log n)$
REDC	Computes polynomial analogue of Montgomery's REDC	$\mathcal{O}(n \log n)$
VANISH	Generates a vanishing polynomial (from section 7.1)	$\mathcal{O}(n \log^2 n)$

Table from https://github.com/wborgeaud/ecfft-bn254

How fast is ECFFT?

Evaluate degree n-1 polynomial on n points:

log n	Naive (ms)	Classic (ms)	ECFFT (ms)
1	0.000165	0.000126	0.000384
2	0.00046	0.000256	0.002144
3	0.00203	0.000639	0.008599
4	0.00688	0.001781	0.030458
5	0.032354	0.003268	0.085556
6	0.119391	0.007594	0.239939
7	0.479542	0.018378	0.613242
8	1.873195	0.043694	1.425794
9	7.619662	0.093	3.964933
10	30.034845	0.20955	9.308925
11	121.564343	0.453727	22.186604
12	482.728362	0.976134	51.505625
13	1930.495799	2.166843	119.317395
14	7745.103265	4.57555	275.499648

Table from
https://github.com/wborgeaud/ecfft-bn254

FFT Recap:

- G: group of order $n = 2^k$
- Degree n polynomial $P(X): G \to \mathbb{F}$ in coefficient representation
- Want to compute evaluations at all points of G

FFT Recap:

- G: group of order $n = 2^k$
- Degree n polynomial $P(X) : G \to \mathbb{F}$ in coefficient representation
- Want to compute evaluations at all points of G

Key ingredients:

- 2-to-1 map $\phi : X \mapsto X^2$
- Decomposition: $P(X) = P_0(X^2) + X \cdot P_1(X^2)$

FFT Recap:

- G: group of order $n = 2^k$
- Degree n polynomial $P(X) : G \to \mathbb{F}$ in coefficient representation
- Want to compute evaluations at all points of G

Key ingredients:

- 2-to-1 map $\phi : X \mapsto X^2$
- Decomposition: $P(X) = P_0(X^2) + X \cdot P_1(X^2)$

- Each iteration halves:
  - Number of evaluation points
  - Degree of polynomials

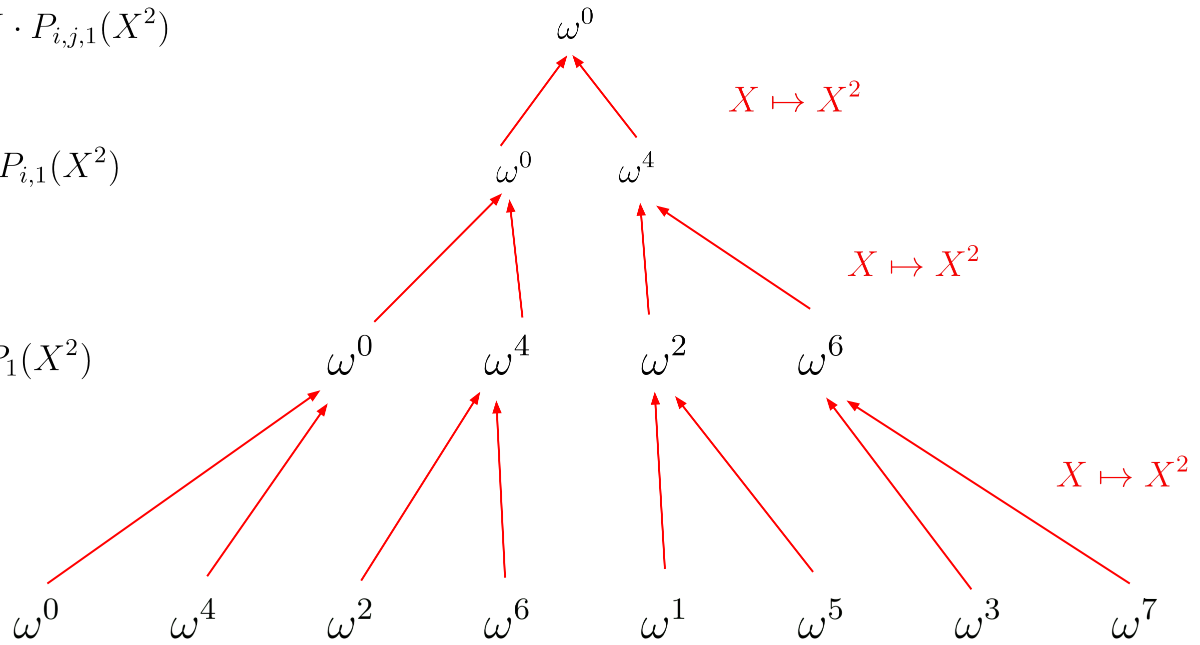
FFT illustrated as an FFTree

$$P_{i,j}(X) = P_{i,j,0}(X^2) + X \cdot P_{i,j,1}(X^2)$$

$$P_i(X) = P_{i,0}(X^2) + X \cdot P_{i,1}(X^2)$$

$$P(X) = P_0(X^2) + X \cdot P_1(X^2)$$

$P(X)$:



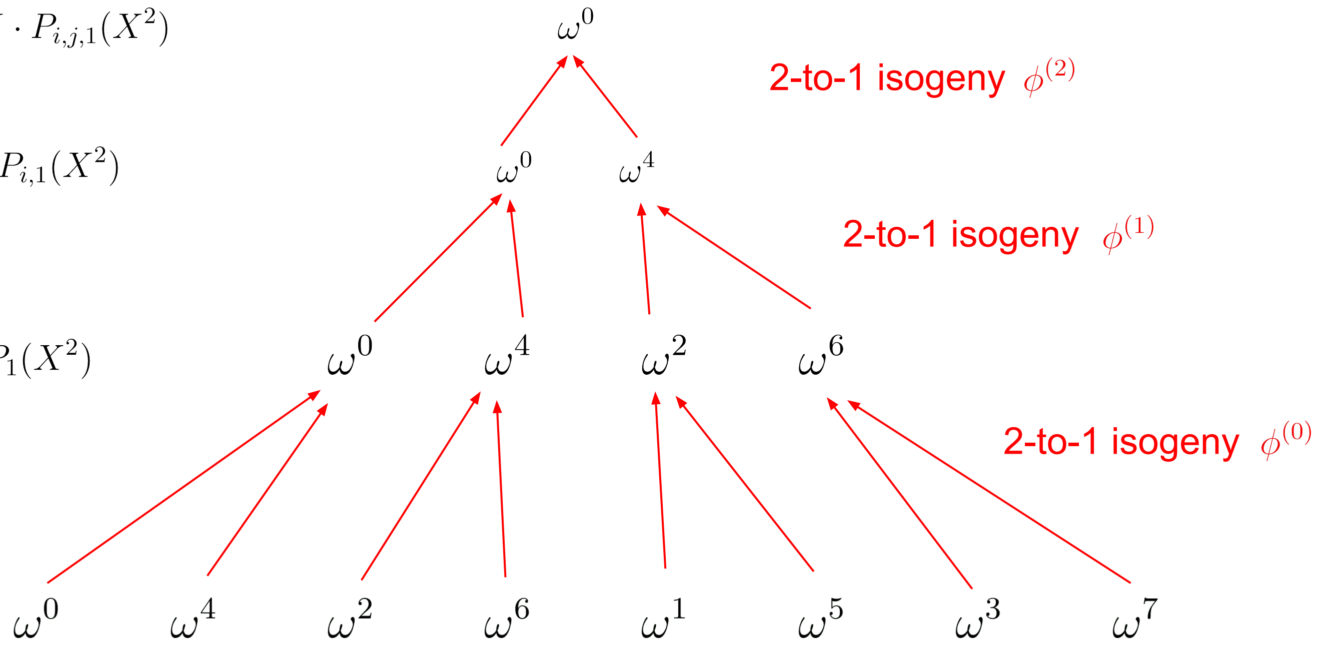
ECFFT illustrated as an FFTree

$$P_{i,j}(X) = P_{i,j,0}(X^2) + X \cdot P_{i,j,1}(X^2)$$

$$P_i(X) = P_{i,0}(X^2) + X \cdot P_{i,1}(X^2)$$

$$P(X) = P_0(X^2) + X \cdot P_1(X^2)$$

$P(X)$:



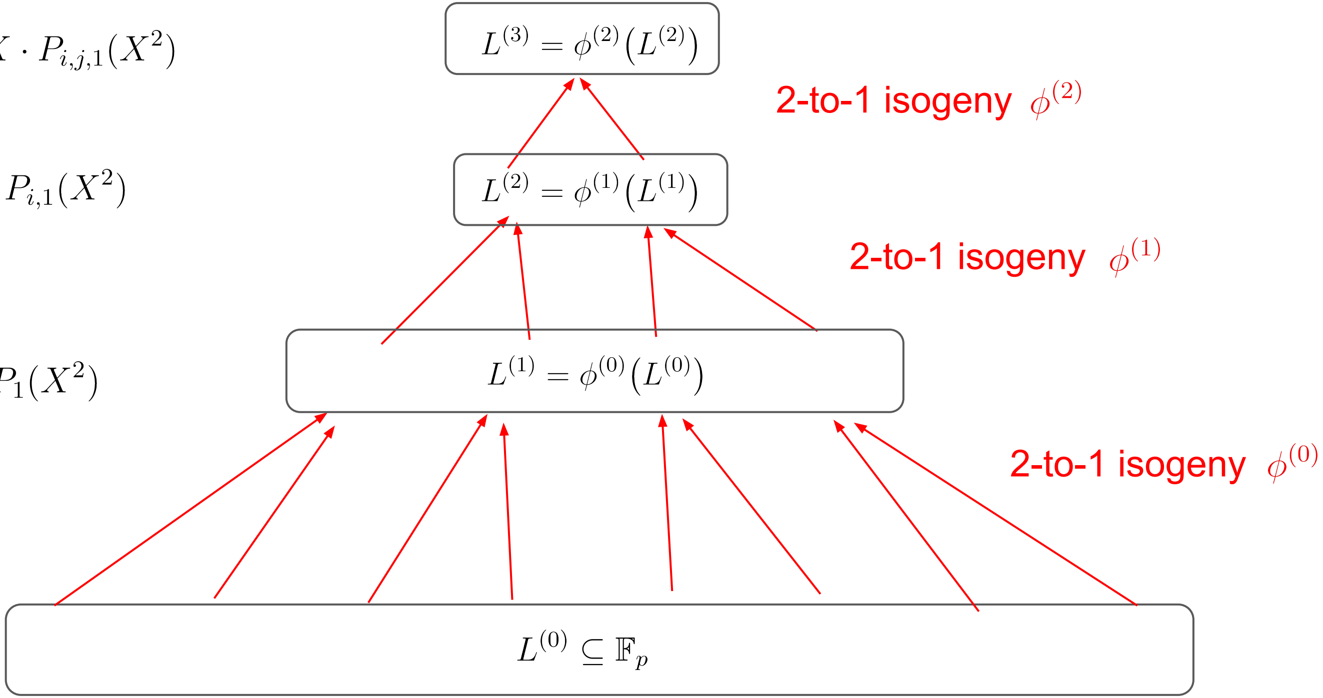
ECFFT illustrated as an FFTree

$$P_{i,j}(X) = P_{i,j,0}(X^2) + X \cdot P_{i,j,1}(X^2)$$

$$P_i(X) = P_{i,0}(X^2) + X \cdot P_{i,1}(X^2)$$

$$P(X) = P_0(X^2) + X \cdot P_1(X^2)$$

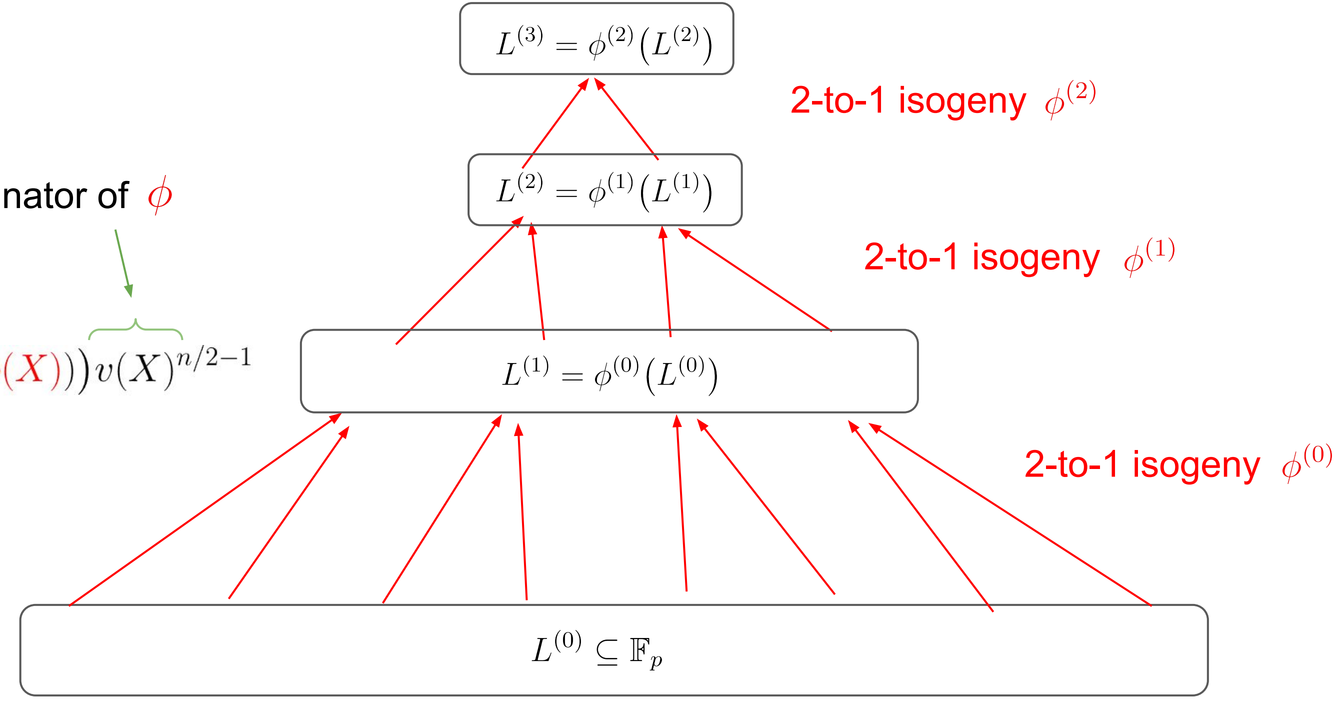
$P(X)$:



ECFFT illustrated as an FFTree

$$P(X) = \big(P_0(\phi(X)) + X P_1(\phi(X))\big)\overbrace{v(X)}^{\text{Denominator of } \phi}{}^{n/2-1}$$

$P(X)$:



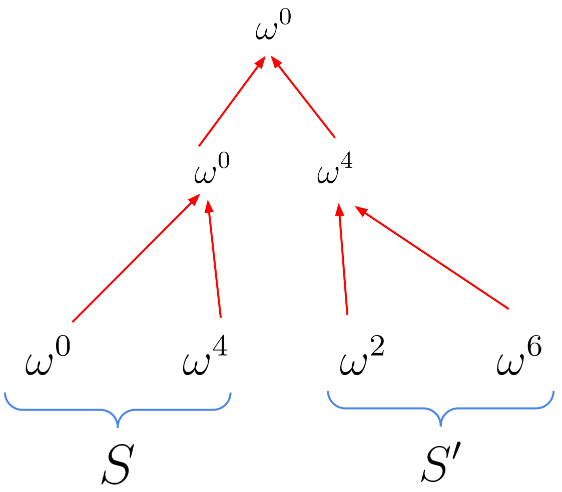
Main algorithm: EXTEND

Given:

- Two sub-FFTrees with n leaves S, S’.
- Evaluations of deg <n polynomial P on S

EXTEND(S,S’) computes evaluations of P on S’ in O(n log n)**:**

- Base case: |S|=|S’|=1 and P is constant.
- Recursive step:
    - Use 2-to-1 isogeny ϕ to decompose evaluations of P on S into evaluations of P_0, P_1 on $\phi(S)$
        - P_0, P_1 have half the degree of P
        - $\phi(S)$ is half the size of S
    - Apply EXTEND to get evaluations of P_0, P_1 on S’
    - Invert decomposition to recover evaluations of P on S’



ENTER: Coefficients -> Evaluations

Given:

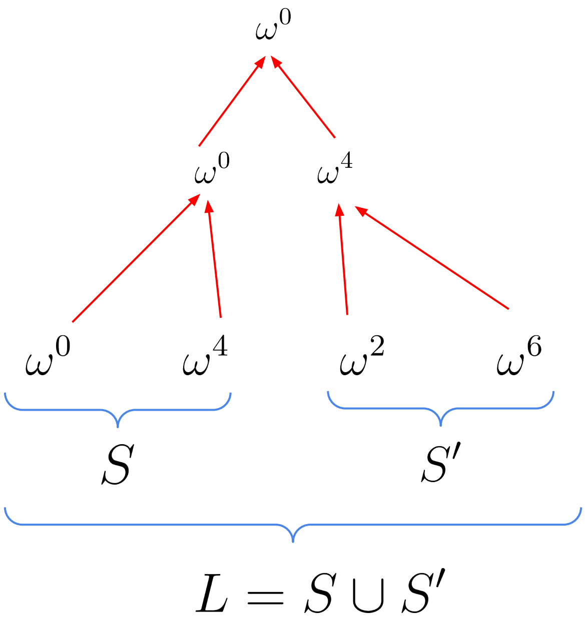
- A sub-FFTree with n leaves L.
- Coefficients of deg <n polynomial

ENTER(L) computes evaluations of P on L in $O(n \log^2 n)$

- Base case: |L|=1 and P is constant.
- Recursive step:
  - Decompose P into low and high coefficients:

$$P(X) = U(X) + X^{n/2} V(X)$$

  - Call ENTER to get evaluations of U, V on S
  - Call EXTEND on U, V to get their evaluations on S'
  - Invert decomposition to get evaluations of P on L



Isogenies: replacing the squaring map

Definition: An isogeny is a morphism between elliptic curves that is also a group homomorphism.

- Morphism: polynomial mapping $\phi : (x : y : z) \mapsto (\phi_0(x, y, z) : \phi_1(x, y, z) : \phi_2(x, y, z))$
- Normalize z=1 $\phi : (x, y) \mapsto \left(\frac{\phi_0(x,y,1)}{\phi_2(x,y,1)}, \frac{\phi_1(x,y,1)}{\phi_2(x,y,1)}\right)$

Equivalent definition: An isogeny $\phi : E \to E'$ is a rational map sending the identity to the identity: $0_E \mapsto 0_{E'}$

Isogenies: replacing the squaring map

Equivalent definition: An isogeny $\phi : E \to E'$ is a rational map sending the identity to the identity: $0_E \mapsto 0_{E'}$

2-to-1 isogeny example:

$$E_0 : y^2 = x^3 + ax^2 + b^2x$$

$$E_1 : y^2 = x^3 + (a + 6b)x^2 + (4ab + 8b^2)x$$

$$\phi : (x, y) \mapsto \left(x - 2b + \tfrac{b^2}{x}, y(1 - \tfrac{b}{x^2})\right)$$

$$\ker(\phi) = \{(0, 0), \infty\}$$

Problem: mismatched types

- Elliptic curve on $\mathbb{F}_p$ contains points $(x, y) \in \mathbb{F}_p^2$
- But for evaluation sets $L^{(i)} \subseteq \mathbb{F}_p$

Problem: mismatched types

- Elliptic curve on $\mathbb{F}_p$ contains points $(x, y) \in \mathbb{F}_p^2$
- But for evaluation sets $L^{(i)} \subseteq \mathbb{F}_p$

Solution:

- Project EC points and isogenies to x-coordinate.
- Theorem: x-coordinate of any isogeny depends only on x-coordinate of input.

$$\phi : (x, y) \mapsto (\phi_x(x), \phi_y(x, y)))$$

Problem: mismatched types

- Elliptic curve on $\mathbb{F}_p$ contains points $(x, y) \in \mathbb{F}_p^2$
- But for evaluation sets $L^{(i)} \subseteq \mathbb{F}_p$

Solution:

- Project EC points and isogenies to x-coordinate.
- Theorem: x-coordinate of any isogeny depends only on x-coordinate of input.

$$\phi : (x, y) \mapsto (\phi_x(x), \phi_y(x, y))$$

This diagram commutes

Hence isogeny x-projections inherit 2-to-1 property

$$\begin{tikzcd}
E_0 \arrow[r, "\psi_0"] \arrow[d, "x"'] & E_1 \arrow[r, "\psi_1"] \arrow[d, "x"'] & E_2 \arrow[r, "\psi_2"] \arrow[d, "x"'] & \dots \arrow[r, "\psi_{n-2}"] & E_{n-1} \arrow[d, "x"'] \\
\mathbb{P}^1(K) \arrow[r, "\psi_{0,x}"] & \mathbb{P}^1(K) \arrow[r, "\psi_{1,x}"] & \mathbb{P}^1(K) \arrow[r, "\psi_{2,x}"] & \dots \arrow[r, "\psi_{n-2,x}"] & \mathbb{P}^1(K)
\end{tikzcd}$$

Polynomial Decomposition

Classical FFT: $P(X) = P_0(X^2) + X \cdot P_1(X^2)$

Polynomial Decomposition

Classical FFT: $P(X) = P_0(X^2) + X \cdot P_1(X^2)$

Generalize from squaring to any degree 2 rational map $\phi(X) = \frac{u(X)}{v(X)}$

Lemma 3.1: For any degree $\leq n$ polynomial $P(X)$, there is a unique pair of degree $\leq n/2$ polynomials $P_0(X), P_1(X)$ such that

$$P(X) = \big(P_0(\phi(X)) + X P_1(\phi(X))\big) v(X)^{n/2-1}$$

Polynomial Decomposition

Classical FFT: $P(X) = P_0(X^2) + X \cdot P_1(X^2)$

Take two points s, s' with same square: $s^2 = s'^2 = t$

Can express decomposition with invertible matrix:

$$\begin{pmatrix} P(s) \\ P(s') \end{pmatrix} = \begin{pmatrix} 1 & s \\ 1 & s' \end{pmatrix} \begin{pmatrix} P_0(t) \\ P_1(t) \end{pmatrix}$$

Polynomial Decomposition

Classical FFT: $P(X) = P_0(X^2) + X \cdot P_1(X^2)$

Take two points s, s' with same square: $s^2 = s'^2 = t$

Can express decomposition with invertible matrix:

$$\begin{pmatrix} P(s) \\ P(s') \end{pmatrix} = \begin{pmatrix} 1 & s \\ 1 & s' \end{pmatrix} \begin{pmatrix} P_0(t) \\ P_1(t) \end{pmatrix}$$

Lemma 3.2: degree 2 rational map $\phi(X) = \frac{u(X)}{v(X)}$ sending both s, s' to t.

Can express decomposition with invertible matrix:

$$\begin{pmatrix} P(s) \\ P(s') \end{pmatrix} = \begin{pmatrix} v(s)^q & sv(s)^q \\ v(s')^q & s'v(s')^q \end{pmatrix} \begin{pmatrix} P_0(t) \\ P_1(t) \end{pmatrix}$$

Steps to implement in practice

- Fix a field $\mathbb{F}_q$

Steps to implement in practice

- Fix a field $\mathbb{F}_q$
- Precompute an FFTree:
  - Find an elliptic curve E on $\mathbb{F}_q$ containing a size $n = 2^k$ subgroup G.
  - Choose a coset $H = r \cdot G$. Its x-projection H_x will be the FFTree leaves.
  - Recursively add an FFTree layer :
    - Find a 2-to-1 isogeny $\phi : E \to E'$
    - Compute 2x2 decomposition matrix for ϕ

Steps to implement in practice

- Fix a field $\mathbb{F}_q$
- Precompute an FFTree:
  - Find an elliptic curve E on $\mathbb{F}_q$ containing a size $n = 2^k$ subgroup G.
  - Choose a coset $H = r \cdot G$. Its x-projection H_x will be the FFTree leaves.
  - Recursively add an FFTree layer :
    - Find a 2-to-1 isogeny $\phi : E \to E'$
    - Compute 2x2 decomposition matrix for ϕ

- Represent polynomials as evaluations on FFTree leaves H_x
- Now you can do ECFFT operations with just $+, \times$ in $\mathbb{F}_q$!