A Mathematical Theory of Danksharding

Yuval Domb
yuval@ingonyama.com

Second Revision

Abstract

Danksharding is the new sharding design proposed for the Ethereum 2.0 blockchain, which introduces significant simplifications compared to previous designs. In Danksharding, the Beacon block is a periodic data structure, constructed by a Builder, whose primary concern is to enable Validators to verify the correctness and availability of its data, using constant-time sampling and verification. This report provides a mathematical analysis of the Beacon block construction process, including new structural and processing improvements and alternatives.

1 Introduction

Danksharding is central to Ethereum's *rollup-centric* roadmap [1]. The idea is to provide space for large blobs of data, which are verifiable and available, without attempting to interpret them. The blobs data space is expected to be used by layer-2 rollup protocols, supporting high throughput transactions. The main innovation introduced by Danksharding is the *merged-fee-market*, which is enabled by enforcing of a single proposer per slot. Danksharding introduces separation of Proposers and Builders, to avoid high system requirements on Validators. The Builders bid for the right to choose the contents of the slot, while the Proposer selects the highest bidder. Only block Builders need to process the entire block, while Validators and users can download and verify parts or all of the data very efficiently through *data-availability-sampling* [2].

This report is focused on the Beacon block building process. Bearing very high computational complexity, optimization of the block building process is invaluable. This, in our opinion, is best done via thorough understanding of the mathematical models supporting the process. The block building process is broken down to five stages: data organization, coefficient extraction, data interpolation, KZG commitments, and KZG proofs. The processing per stage is described in detail, attempting to be as self-contained as possible. The reader is encouraged to obtain some basic understanding of Discrete-Fourier-Transform (DFT) [3] prior to reading the report.

One outcome of the report is reinforcement of the notion that the basic computationally dominant primitives, required by Danksharding, are highly correlated with ones required for Zero-Knowledge-Proofs. Amongst those primitives are Multi-Scalar-Multiplications (MSM), Number-Theoretic-Transforms (NTT), a new class of NTT - NTT over Elliptic-Curve (EC) group-element vectors (ECNTT), and various lower-lever finite-field arithmetic primitives, such as modular-multiplier, EC-adder etc.

The report structure follows the chronological order of the processing stages, presenting required mathematical tools as needed.

2 Preliminaries

All references to EC, in this report, refer to BLS12_381 [4]. We are mainly concerned with the EC Abelian-group $\mathbb{G}_1$ and the EC scalar-field $\mathbb{F}_r$. Both are of size $r < 2^{256}$ with

$$r = 0\text{x}73eda753299d7d483339d80809a1d80553bda402fffe5bfeffffffff00000001$$

Note that 2^{32} divides the size of the multiplicative-group in $\mathbb{F}_r$. Field-elements and group-elements refer hereafter to elements from $\mathbb{F}_r$ and $\mathbb{G}_1$, respectively. A field-element is conveniently described using a 32 bytes unsigned integer smaller than r. Field-addition and field-multiplication refer hereafter to operations in $\mathbb{F}_r$. Group-addition and scalar-multiplication refer hereafter to addition in $\mathbb{G}_1$ and multiplication of a group-element in $\mathbb{G}_1$ by a field-element in $\mathbb{F}_r$, respectively. We use s, $\mathbf{v}$, and $\mathbf{M}$ to denote field-element scalars, vectors, and matrices, respectively. We use $\mathcal{G}$ and $\boldsymbol{\mathcal{G}}$ to denote group-element scalars and vectors, respectively.

3 Data Organization

The input data consists of $n = 256$ shard-blobs. Each shard-blob is a vector of $m = 4096$ field-elements referred-to as symbols. The data symbols are organized in an $n \times m$ input matrix

$$\mathbf{D}^{\text{in}}_{n\times m} = \begin{pmatrix} d(0,0) & d(0,1) & ... & d(0,m-1) \\ d(1,0) & d(1,1) & ... & d(1,m-1) \\ ... & ... & ... & ... \\ d(n-1,0) & d(n-1,1) & ... & d(n-1,m-1) \end{pmatrix} \tag{1}$$

where each row is one of the shard-blob vectors [5].

In order to enable interpolation and polynomial-commitment to the data, we will proceed to treat the data symbols as polynomial evaluations. To that end, the data symbols can be treated as evaluations of a 2D-polynomial, as suggested by the Ethereum foundation. We suggest a simpler, in our opinion, way to satisfy the necessary requirements by treating each data symbol as an evaluation of two 1D-polynomials, corresponding to the row and column of the symbol. This interlaced coding method is typically referred to as a product-code [6]. The proceeding sections analyze both alternatives and their consequences.

As will be presented shortly, DFT enables efficient data processing so long as the data is accessed in cosets. Let us define a data-coset as a batch of data symbols whose domain forms a coset of a multiplicative subgroup. Using appropriately sized roots-of-unity (multiplicative) groups for the data domain allows forming these cosets. Let us thus associate each domain location in the input matrix with a field-element pair (u_μ, w_η), where $\mu \in [0, n-1]$, $\eta \in [0, m-1]$ correspond to the row and column indexes, respectively. The row field-element is defined as $u_\mu \equiv u^{\text{rbo}(\mu)}$, where u is a $2n$'th root-of-unity such that $u^{2n} = 1$. The column field-element is defined as $w_\eta \equiv w^{\text{rbo}(\eta)}$, where w is a $2m$'th root-of-unity such that $w^{2m} = 1$. The function $\text{rbo}(\cdot)$ outputs the reversed-bit-order of its input. Examine, for

instance, the column domain. The complete domain $\{u^0, u^1, u^2, ..., u^{2n-1}\}$ forms a group, and appropriately selected subsets such as $\{u^0, u^2, u^4, ..., u^{2n-2}\}$ form subgroups. A coset is defined as a subgroup translation such as $\{u^1, u^3, u^5, ..., u^{2n-1}\}$. Using reverse-bit-order ordering rather than natural-ordering allows accessing cosets in block (consecutive) rather than interleaved manner. This block-wise access will be used with the reverse-bit-ordering assumption implicitly, henceforth.

4 Coefficients Extraction

The size of the input data matrix $\mathbf{D}^{\text{in}}$ is $n \times m$. The row (column) domain of the input data is the first n (m) locations in the reverse-bit-ordered, root-of-unity group formed by u (w). As such it forms the domain subgroup corresponding to the n'th (m'th) root-of-unity. Taking the data symbols to be evaluations of a 2D-polynomial or 1D-product-polynomials with row degree $n-1$ and column degree $m-1$ uniquely defines the polynomials' coefficients.

4.1 2D Coefficients Extraction

The 2D-polynomial representing the input data can be expressed as

$$d(x, y) = \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} \hat{c}[i, j] x^i y^j \tag{2}$$

where x and y correspond to the previously mentioned rows and columns, respectively. The input data evaluations (1), treated as $d(\mu, \eta) = d(u_\mu, w_\eta)$ form a fully-constrained linear system for which the polynomial coefficients can be extracted.

Plugging an evaluation from (1) into (2) results in the following:

$$d(u_\mu, w_\eta) = \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} \hat{c}[i, j] u_\mu^i w_\eta^j \tag{3}$$

Defining a scaled version of the coefficients as

$$c[i, j] = \sqrt{nm} \cdot \hat{c}[i, j] \tag{4}$$

leads to the following normalized DFT relationship between polynomial coefficients $c[i, j]$ and evaluations $d(u_\mu, w_\eta)$

$$d(u_\mu, w_\eta) = \frac{1}{\sqrt{nm}} \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} c[i, j] u_\mu^i w_\eta^j \tag{5}$$

$$= \frac{1}{\sqrt{m}} \sum_{j=0}^{m-1} \left(\frac{1}{\sqrt{n}} \sum_{i=0}^{n-1} c[i, j] u_\mu^i \right) w_\eta^j \tag{6}$$

$$= \mathcal{F}_m \left\{ \mathcal{F}_n \left\{ c[i, j] \right\}_{i=0}^{n-1} \right\}_{j=0}^{m-1} \Bigg|_{(\mu,\eta)} \tag{7}$$

$$= \mathcal{F}_n \left\{ \mathcal{F}_m \left\{ c[i, j] \right\}_{j=0}^{m-1} \right\}_{i=0}^{n-1} \Bigg|_{(\mu,\eta)} \tag{8}$$

where $\mathcal{F}_m$ can be represented by right-multiplication by

$$\mathbf{W} = \frac{1}{\sqrt{m}} \begin{pmatrix} 1 & 1 & ... & 1 \\ 1 & w_1 & ... & w_1^{m-1} \\ . & . & ... & . \\ 1 & w_{m-1} & ... & w_{m-1}^{m-1} \end{pmatrix} \tag{9}$$

and $\mathcal{F}_n$ can be represented by left-multiplication by

$$\mathbf{U} = \frac{1}{\sqrt{n}} \begin{pmatrix} 1 & 1 & ... & 1 \\ 1 & u_1 & ... & u_{n-1} \\ . & . & ... & . \\ 1 & u_1^{n-1} & ... & u_{n-1}^{n-1} \end{pmatrix} \tag{10}$$

The input data matrix can thus be extracted from the 2D-polynomial coefficients as

$$\mathbf{D}^{\text{in}} = \mathbf{UCW} \tag{11}$$

where

$$\mathbf{C}_{n\times m} = \begin{pmatrix} c[0,0] & c[0,1] & ... & c[0,m-1] \\ c[1,0] & c[1,1] & ... & c[1,m-1] \\ ... & ... & ... & ... \\ c[n-1,0] & c[n-1,1] & ... & c[n-1,m-1] \end{pmatrix} \tag{12}$$

For the remainder of the document, let us denote the forward DFT relation, going from the coefficient-domain (time-domain) to the evaluation-domain (frequency-domain) as the Number Theoretic Transform (NTT) and its inverse the Inverse NTT (INTT).

The claim that $\mathbf{W}$ and $\mathbf{U}$ are NTT matrices is not completely trivial and requires some clarification, due to the utilized reverse-bit-ordering. For this, with no loss of generality, let us examine $\mathbf{W}$. Denote by $\widetilde{\mathbf{W}}$ the natural-ordered NTT matrix

$$\widetilde{\mathbf{W}} = \frac{1}{\sqrt{m}} \begin{pmatrix} 1 & 1 & ... & 1 \\ 1 & w & ... & w^{m-1} \\ . & . & ... & . \\ 1 & w^{m-1} & ... & w^{(m-1)^2} \end{pmatrix} \tag{13}$$

and its corresponding INTT matrix

$$\widetilde{\mathbf{W}}^{-1} = \widetilde{\mathbf{W}}^{H} = \frac{1}{\sqrt{m}} \begin{pmatrix} 1 & 1 & ... & 1 \\ 1 & w^{-1} & ... & w^{-(m-1)} \\ . & . & ... & . \\ 1 & w^{-(m-1)} & ... & w^{-(m-1)^2} \end{pmatrix} \tag{14}$$

where H denotes the conjugate transpose operator[1]. It's easy to verify that $\widetilde{\mathbf{W}}$ is unitary since $|\widetilde{\mathbf{W}}| = 1$ and $\widetilde{\mathbf{W}}^H\widetilde{\mathbf{W}} = \mathbf{I}$ where $\mathbf{I}$ is the identity matrix, hence it is a valid NTT. The reverse-bit-ordering can be defined as a permutation matrix $\mathbf{P}$ therefore

$$\mathbf{W} = \mathbf{P}\widetilde{\mathbf{W}} \tag{15}$$

[1]Conjugate for finite field-elements is defined in the usual way as w^* s.t. $ww^* = w^*w = |w|^2$.

and thus $\mathbf{W}$ is also unitary and a valid NTT.

Extraction of the 2D-polynomial coefficients, based on (11), can thus be described as

$$\mathbf{C} = \mathbf{U}^H \mathbf{D}^{\text{in}} \mathbf{W}^H \tag{16}$$

Note that (16) is equivalent to performing INTT over the rows and then INTT over the columns of the result, or visa-versa.

4.2 1D Coefficients Extraction

The 1D row and column polynomials representing the input data $d(u_\mu, w_\eta)$ can be depicted as

$$d_\mu^{\text{row}}(w_\eta) = \frac{1}{\sqrt{m}} \sum_{j=0}^{m-1} c_\mu^{\text{row}}[j] w_\eta^j \tag{17}$$

$$d_\eta^{\text{col}}(u_\mu) = \frac{1}{\sqrt{n}} \sum_{i=0}^{n-1} c_\eta^{\text{col}}[i] u_\mu^i \tag{18}$$

where $d_\mu^{\text{row}}(w_\eta) = d_\eta^{\text{col}}(u_\mu) = d(u_\mu, w_\eta)$. The total number of product-code polynomials is $n + m$. The input data matrix can thus be extracted from the 1D-polynomials as

$$\mathbf{D}^{\text{in}} = \mathbf{C}^{\text{rows}} \mathbf{W} = \mathbf{U} \mathbf{C}^{\text{cols}} \tag{19}$$

which immediately leads to

$$\mathbf{C}^{\text{rows}} = \mathbf{D}^{\text{in}} \mathbf{W}^H \tag{20}$$

$$\mathbf{C}^{\text{cols}} = \mathbf{U}^H \mathbf{D}^{\text{in}} \tag{21}$$

where the rows of $\mathbf{C}^{\text{rows}}$ are the n row product-codes, and the columns of $\mathbf{C}^{\text{cols}}$ are the m column product-codes. Note that the size of the 1D coefficient representation is double the size of the 2D representation.

5 Data Interpolation

The full (interpolated) data matrix can be described as the concatenation of four sub-matrices as follows:

$$\mathbf{D} = \begin{pmatrix} \mathbf{D}_{n\times m}^{\text{in}} & \mathbf{D}_{n\times m}^{\text{rows}} \\ \mathbf{D}_{n\times m}^{\text{cols}} & \mathbf{D}_{n\times m}^{\text{both}} \end{pmatrix} \tag{22}$$

5.1 2D Data Interpolation

The 2D-interpolation is achieved by

$$\mathbf{D} = \mathcal{F}_{2n} \left\{ \mathcal{F}_{2m} \left\{ c[i,j] \right\}_{j=0}^{2m-1} \right\}_{i=0}^{2n-1} \tag{23}$$

where $c[i,j] \equiv 0,\ \forall i \geq n, j \geq m$, $\mathcal{F}_{2m}$ is a length $2m$ row-wise NTT, and $\mathcal{F}_{2n}$ is a length $2n$ column-wise NTT.

A more efficient way to achieve this is by noting that the interpolated evaluations are a time-shifted version of the original evaluations which translates to frequency-modulation in the coefficient-domain. This immediately translates to the following scheme

$$\mathbf{D}^{\text{rows}} = \mathbf{U}\mathbf{C}\Lambda_w\mathbf{W} \tag{24}$$

$$\mathbf{D}^{\text{cols}} = \mathbf{U}\Lambda_u\mathbf{C}\mathbf{W} \tag{25}$$

$$\mathbf{D}^{\text{both}} = \mathbf{U}\Lambda_u\mathbf{C}\Lambda_w\mathbf{W} \tag{26}$$

where $\Lambda_w = \text{diag}([1\, w\, w^2 \,...\, w^{m-1}])$ and $\Lambda_u = \text{diag}([1\, u\, u^2 \,...\, u^{n-1}])$ are the frequency-modulation matrices.

5.2 1D Data Interpolation

Similar arguments translates to the following scheme for the 1D case

$$\mathbf{D}^{\text{rows}} = \mathbf{C}^{\text{rows}}\Lambda_w\mathbf{W} \tag{27}$$

$$\mathbf{D}^{\text{cols}} = \mathbf{U}\Lambda_u\mathbf{C}^{\text{cols}} \tag{28}$$

$$\mathbf{D}^{\text{both}} = \mathbf{U}\Lambda_u\mathbf{U}^H\mathbf{D}^{\text{rows}} = \mathbf{D}^{\text{cols}}\mathbf{W}^H\Lambda_w\mathbf{W} \tag{29}$$

6 KZG Commitments

From this stage and on, each row of the the interpolated data matrix $\mathbf{D}$ is treated as evaluations of a 1D-polynomial of degree $m-1$, as was done previously for the 1D case, namely $d_\mu(x)$. This allows us to reuse the row-polynomials $\mathbf{C}^{\text{rows}}$ from (20). For the 2D case, one can calculate an equivalent as $\mathbf{C}^{\text{rows}} = \mathbf{U}\mathbf{C}$. This conclusion is easily reached from simple comparison of (11) and (19).

Each row-polynomial is committed to individually. The commitment scheme is polynomial-KZG [7] with a single predetermined setup of length m

$$[1], [s], ..., [s^{m-2}], [s^{m-1}] \tag{30}$$

where $[s^k] \equiv s^k \cdot \mathcal{G}$ for some secret field-element s, and generator group-element $\mathcal{G}$. In general, we use the form $[f(s)]$ to denote the evaluation of a polynomial $f(x) = \sum_i f_i x^i$ at group-element $[s]$, i.e. $[f(s)] = \sum_i f_i[s^i]$. A commitment to row $\mu \in [0, 2n-1]$ is an evaluation of its polynomial $d_\mu(x)$ at $[s]$, i.e. $\mathcal{K}_\mu = [d_\mu(s)]$. Commitments for rows $[0, n-1]$ are calculated according to

$$\boldsymbol{\mathcal{K}}_{0:n-1} = \mathbf{C}^{\text{rows}}[\mathbf{s}] \tag{31}$$

where $[\mathbf{s}] \equiv \left[[1], [s], ..., [s^{m-2}], [s^{m-1}]\right]^T$ and T is the transpose operator. Note that the above is equivalent to n, m-long Multi-Scalar Multiplications (MSM). Commitments for rows $[n, 2n-1]$ can be interpolated following (29) as

$$\boldsymbol{\mathcal{K}}_{n:2n-1} = \mathbf{U}\Lambda_u\mathbf{U}^H\boldsymbol{\mathcal{K}}_{0:n-1} \tag{32}$$

The implied NTTs in (32) are performed over EC group-element vectors and are termed ECNTT[2]. The interpolation is valid due to linearity. More specifically, linearity applies since $\mathbb{F}_r$ and $\mathbb{G}_1$ are a linear bijection, i.e. $[s_0 + s_1] = [s_0] + [s_1]$ and $|\mathbb{F}_r| = |\mathbb{G}_1|$.

[2]Note that optimization of ECNTT is different from NTT since field-multiplications are traded for scalar-multiplications whose computational complexity is typically proportional to $log_2(s)$ where s is the scalar.

7 KZG Proofs

The proofs are Multi-reveal KZG proofs (MKZG) following the scheme from [8]. Each proof is constructed for a sample of $l = 16$ data symbols from $\mathbf{D}$. Each sample corresponds to polynomial evaluations over a coset of the subgroup $[1, \psi, \psi^2, ..., \psi^{l-1}]$, where ψ is an l'th root-of-unity and each sample is taken from a distinct row $\mu \in [0, 2n-1]$. Each sample proof is constructed to prove that

$$\begin{pmatrix} d_\mu(\omega^k) \\ d_\mu(\omega^k\psi) \\ . \\ d_\mu(\omega^k\psi^{l-1}) \end{pmatrix} = \begin{pmatrix} z_0 \\ z_1 \\ . \\ z_{l-1} \end{pmatrix} \tag{33}$$

where $k \in [0, \frac{2m}{l} - 1]$. To this end, we compute the quotient polynomial

$$q_{\mu,k}(x) = \frac{d_\mu(x) - r_{\mu,k}(x)}{x^l - \omega^{kl}} = \frac{d_\mu(x) - r_{\mu,k}(x)}{x^l - \phi^k} \tag{34}$$

where $\phi = \omega^l$ is a $\frac{2m}{l}$'th root-of-unity, $r_{\mu,k}(x) = d_\mu(x) \mod (x^l - \phi^k)$ is the remainder polynomial which is the unique polynomial of degree $l-1$ that obeys (34) (see [9] for a direct construction method), and $(x^l - \omega^{kl}) = (x - \omega^k)(x - \omega^k\psi)\cdots(x - \omega^k\psi^{l-1})$ by definition of a roots-of-unity group [10]. The MKZG proof is simply an evaluation of the quotient polynomial at $[s]$, i.e. $\Pi_{\mu,k} = [q_{\mu,k}(s)]$. Verification is achieved using

$$e(\mathcal{K}_\mu - [r_{\mu,k}(s)],\ [1]) = e(\Pi_{\mu,k},\ [s^l] - [\phi^k]) \tag{35}$$

where $e(\cdot,\cdot)$ is a pairing operator.

Equation (34) can be rewritten as

$$\begin{aligned} q_{\mu,k}(x) &= \frac{d_\mu(x) - r_{\mu,k}(x) + \phi^k q_{\mu,k}(x)}{x^l} && (36)\\ &= \left\lfloor \frac{d_\mu(x) + \phi^k q_{\mu,k}(x)}{x^l} \right\rfloor && (37)\\ &= \left\lfloor \frac{d_\mu(x)}{x^l} \right\rfloor + \left\lfloor \frac{d_\mu(x)}{x^{2l}} \right\rfloor \phi^k + \left\lfloor \frac{d_\mu(x)}{x^{3l}} \right\rfloor \phi^{2k} + \cdots + \left\lfloor \frac{d_\mu(x)}{x^{(\frac{m}{l}-1)l}} \right\rfloor \phi^{(\frac{m}{l}-2)k} && (38) \end{aligned}$$

where the floor operator signifies truncated division, i.e. terms with negative exponents $x^i,\ \forall i < 0$ are discarded. This can be presented more compactly in vector form as

$$\begin{aligned} q_{\mu,k}(x) &= \boldsymbol{\phi}_k^T \cdot \mathbf{d}_\mu(x) && (39)\\ \boldsymbol{\phi}_k^T &= \left[1, \phi^k, \phi^{2k}, \cdots, \phi^{(\frac{m}{l}-2)k}, \phi^{(\frac{m}{l}-1)k}\right] && (40)\\ \mathbf{d}_\mu(x) &= \left[\left\lfloor \frac{d_\mu(x)}{x^l} \right\rfloor, \left\lfloor \frac{d_\mu(x)}{x^{2l}} \right\rfloor, \left\lfloor \frac{d_\mu(x)}{x^{3l}} \right\rfloor, \cdots, \left\lfloor \frac{d_\mu(x)}{x^{(\frac{m}{l}-1)l}} \right\rfloor, 0\right]^T && (41) \end{aligned}$$

where we extend both vectors to length $\frac{m}{l}$, adding one redundant element for convenient NTT sizing. The last two required operations are described in the following subsections.

7.1 Calculating $[\mathbf{d}_\mu(s)]$

Evaluating (41) at $[s]$ (i.e. $[\mathbf{d}_\mu(s)]$) can be done as follows:

$$[\mathbf{d}_\mu(s)] = \left[\sum_{j=tl}^{m-1} c_\mu^{\text{row}}[j][s^{j-tl}]\right]_{t\in[1,\frac{m}{l}]} \tag{42}$$

where $\mathbf{c}_\mu^{\text{row}}$ is row-μ of $\mathbf{C}^{\text{rows}}$. One method by which this can be achieved is by performing $(\frac{m}{l}-1)$ MSMs of varying sizes (note that the last element in the vector does not need to be calculated and is by definition the EC point-at-infinity).

Another method is as follows. We start by noting that equation (42) is an l-downsample of the output of a linear convolution. The linear convolution can be achieved by element-wise multiplication in the evaluations-domain. The coefficient-domain sequences ($\mathbf{c}_\mu^{\text{row}}$ and $[\mathbf{s}]$) must be zero padded prior to the NTT in order that the cyclic-convolution nature of the NTT does not alias the desired linear-convolution onto itself. Zero-padding to length $2m$ is sufficient. The frequency-domain of the coefficients is already available as row μ of $\mathbf{D}$. Since the setup does not change, its interpolated frequency-domain $\boldsymbol{\mathcal{S}}$ can be pre-calculated as

$$\boldsymbol{\mathcal{S}}_{0:m-1} = \overleftarrow{[\mathbf{s}]}^T\mathbf{W} \tag{43}$$

$$\boldsymbol{\mathcal{S}}_{m:2m-1} = \overleftarrow{[\mathbf{s}]}^T\Lambda_w\mathbf{W} \tag{44}$$

where $\overleftarrow{[\mathbf{s}]}$ is the reversed-order vector of $[\mathbf{s}]$. This is required since the r.h.s. of (42) is technically a correlation and not a convolution. The two frequency-domain vectors are element-wise multiplied and l-fold aliased as

$$[\boldsymbol{\delta}_\mu(s)] = \left[\frac{1}{l}\sum_{j=0}^{l-1} d_\mu^{\text{row}}\left[j\frac{m}{l}+t\right]\boldsymbol{\mathcal{S}}\left[j\frac{m}{l}+t\right]\right]_{t\in[0,\frac{m}{l}-1]} \tag{45}$$

where the l-fold aliasing is the evaluation-domain equivalent of the l-downsampling in the coefficient-domain. Finally, $[\mathbf{d}_\mu(s)]$ is the $\frac{m}{l}$-size INTT of $[\boldsymbol{\delta}_\mu(s)]$.

7.2 Calculating Row Proofs $[\mathbf{q}_\mu(s)]$

Going back to (39), note that all row-μ proofs

$$[\mathbf{q}_\mu(s)] \equiv \left[[q_{\mu,0}(s)], [q_{\mu,1}(s)], \cdots, [q_{\mu,\frac{2m}{l}-1}(s)]\right] \tag{46}$$

can be calculated by a $\frac{2m}{l}$-size NTT, since $\boldsymbol{\phi}_k^T$ is the NTT vector for frequency ϕ^k.

Interpolation tricks, similar to the ones used before, can be utilized to calculate only the first $\frac{m}{l}$ proofs directly and interpolate the rest.

Extending the proofs for all rows can be done directly by repeating this process for all $\mu \in [0, 2n-1]$ or again by calculating the first n and interpolating the rest.

Acknowledgement

The author would like to thank Karthik Inbasekar, Dmytro Tymokhanov, and Omer Shlomovits for helpful discussions. Additionally, a special thanks to Kevaundray Wedderburn at Ethereum Foundation for finding a small mistake that was subsequently corrected in the second revision.

References

[1] Vitalik Buterin. Proto-danksharding faq. https://notes.ethereum.org/@vbuterin/proto_danksharding_faq#:~:text=is%20just%20data).-,What%20is%20proto%2Ddanksharding%20(aka.,yet%20actually%20implementing%20any%20sharding.

[2] Vitalik Buterin. An explanation of the sharding + das proposal. https://hackmd.io/@vbuterin/sharding_proposal.

[3] Discrete fourier transform. https://en.wikipedia.org/wiki/Discrete_Fourier_transform.

[4] Ben Edgington. Bls12-381 for the rest of us. https://hackmd.io/@benjaminion/bls12-381.

[5] Dankrad Feist. Data availability encoding. https://notes.ethereum.org/@dankrad/danksharding_encoding.

[6] E. R. Berlekamp. *Algebraic coding theory.* Aegean Park Press, Laguna Hills, CA, USA, 1984.

[7] Dankrad Feist. Kzg polynomial commitments. https://dankradfeist.de/ethereum/2020/06/16/kate-polynomial-commitments.html, 2020.

[8] Dankrad Feist and Dmitry Khovratovich. Fast amortized kate proofs. https://github.com/khovratovich/Kate/blob/master/Kate_amortized.pdf, 2020.

[9] Vitalik Buterin. Quadratic arithmetic programs: from zero to hero. https://medium.com/@VitalikButerin/quadratic-arithmetic-programs-from-zero-to-hero-f6d558cea649, 2016.

[10] Ray Li. Roots of unity. http://theory.stanford.edu/~rayyli/static/contest/lectures/Ray%20Li%20rootsofunity.pdf, 2021.