Notes on KZG ccommitment scheme

Karthik Inbasekar[a]

[a] *Ingonyama*

E-mail: karthik@ingonyama.com

1 KZG opening proofs for evaluation at random points in $\mathbb{F}_p$

Let us begin by reviewing the KZG commitment scheme [1], but albeit in a slightly more explicit form not discussed elsewhere. The problem we are considering is how to commit to a given polynomial

$$f(X) = \sum_{i=0}^{n-1} a_i \cdot X^i \tag{1.1}$$

where $a_i \in \mathbb{F}$ and convince the verifier of the commitment by opening the commitment at a challenge value provided the verifier. This requires a trusted setup, we assume that there is a setup process that created a string of secret scalars $\{1, s, s^2, \ldots, s^{n-1}\}$ which are then used to generate the Structured Reference String (SRS) defined as

$$\mathtt{SRS}^{G_1} \equiv \{s, s \cdot g, s^2 \cdot g, \ldots, s^{n-1} \cdot g\} \tag{1.2}$$

$$\mathtt{SRS}^{G_2} \equiv \{g', s \cdot g'\} \tag{1.3}$$

In the above G_1 is a prime order subgroup of points in an elliptic curve $E/\mathbb{F}_p$ with a generator g. Similarly G_2 is a prime order subgroup of points in an elliptic curve $E/\mathbb{F}_{p^k}$ and we will assume $k = 2$ without loss of generality (This is the case for curves like BN254, BLS12-377, BLS12-381 etc). Given the SRS, we can now define the commitment to the polynomial (1.1) as

$$\begin{aligned}[f(s)]_1 \equiv f(s) \cdot g &= \sum_{i=0}^{n-1} a_i s^i \cdot g \\ &= \sum_{i=0}^{n-1} a_i \cdot \mathtt{SRS}_i^{G_1}\end{aligned} \tag{1.4}$$

Note that $f(s)$ is a formal expression to denote evaluation at a secret scalar, no one knows s, in general the commitment refers to the second line. In general we will use the definition

$$[A]_1 \equiv A \cdot g \tag{1.5}$$

$$[A]_2 \equiv A \cdot g' \tag{1.6}$$

to denote evaluations at the secret scalar multiplied by the group element and hence protected by the DLOG.

Given the definition of the commitment, how can the verifier check that this is indeed the claimed polynomial. The verifier generates a challenge (is it seeded by the commitment?) $\beta \in \mathbb{F}_p$ and asks the prover to evaluate the polynomial at this point.

$$f(\beta) = \sum_{i=0}^{n-1} a_i \beta^i \tag{1.7}$$

The prover sends the evaluation $f(\beta)$ to the verifier, and furthermore if the prover is honest the following check will pass

$$t_s(\beta) \equiv \frac{f(\beta) - f(s)}{\beta - s} \tag{1.8}$$

i.e $(\beta - s)$ will divide the function $f(\beta) - f(s)$. Note that s is a secret scalar that no one has access to. So how does one compute this equation? Before we get into that, we intuitively explain why $(\beta - s)$ divides $f(\beta) - f(s)$. First consider the function

$$f(X) - f(Y) = a_1 \cdot (X - Y) + a_2 \cdot (X^2 - Y^2) + \ldots + a_{n-1}(X^{n-1} - Y^{n-1}) \tag{1.9}$$

Plugging the relation

$$X^{r-1} - Y^{r-1} = (X - Y) \cdot X^{r-2} \cdot \sum_{i=0}^{r-2} \left(\frac{Y}{X}\right)^i \quad \forall r \geq 2 \tag{1.10}$$

into (1.9) we see that every term in (1.9) is proportional to $(X - Y)$, i.e $(X - Y)$ divides (1.9). Thus if the prover is honest, he/she should be able to compute the polynomial

$$\begin{aligned} t_Y(X) &\equiv \frac{f(X) - f(Y)}{X - Y} \\ &\equiv \sum_{r=1}^{n-1} a_r X^{r-1} \cdot \sum_{i=0}^{r-1} \left(\frac{Y}{X}\right)^i \end{aligned} \tag{1.11}$$

i.e if the prover is honest, (1.11) is indeed a polyomial and not a rational function. We denote as the commitment to the t polynomial evaluated on the challenge β as

$$\begin{aligned} [t(\beta)]_1 &\equiv t_s(\beta) \cdot g \\ &= \sum_{r=1}^{n-1} a_r \beta^{r-1} \cdot \sum_{i=0}^{r-1} \left(\frac{s}{\beta}\right)^i \cdot g \\ &= \sum_{r=1}^{n-1} a_r \beta^{r-1} \cdot \sum_{i=0}^{r-1} \beta^{-i} \cdot \mathtt{SRS}_i^{G_1} \end{aligned} \tag{1.12}$$

where the last line is the actual computation performed by the prover.

Thus the verifier can check (1.8) using the pairing equation

$$e\left([f(\beta) - f(s)]_1, [1]_2\right) \stackrel{?}{=} e\left([t(\beta)]_1, [\beta - s]_2\right) \tag{1.13}$$

where all the quantities are calculated using

$$[f(\beta)-f(s)]_1 = [f(\beta)]_1 - [f(s)]_1 \equiv f(\beta)\cdot \mathtt{SRS}_0^{G_1} - \sum_{i=0}^{n-1} a_i \cdot \mathtt{SRS}_i^{G_1}$$

$$\begin{aligned} [1]_2 &= \mathtt{SRS}_0^{G_2} \\ [\beta - s]_2 &\equiv [\beta]_2 - [s]_2 \equiv \beta \cdot \mathtt{SRS}_0^{G_2} - \mathtt{SRS}_0^{G_2} \end{aligned} \tag{1.14}$$

In order to evaluate (1.8) the verifier needs the prover to send a KZG proof

$$\pi_{KZG} = \{[f(s)]_1, [t(\beta)_1], f(\beta)\} \tag{1.15}$$

that consists of $2G_1$ elements and $1\mathbb{F}$ element. A summary of the protocol is given in table 1.

Round	Prover $\mathcal{P}$	Communication	Verifier $\mathcal{V}$
1	compute $[f(s)]_1$ (MSM)	$[f(s)]_1 \longrightarrow$ $\longleftarrow \beta \in \mathbb{F}_p$	Store $[f(s)]_1$
2	Evaluate $f(\beta) = \sum_{i=0}^{n-1} a_i \cdot \beta^i$ Define $t(X,Y) \equiv \frac{t(X)-t(Y)}{X-Y}$ Compute $[t(\beta)]_1$	$f(\beta) \longrightarrow$ $[t(\beta)]_1 \longrightarrow$	compute $[f(\beta))]_1 \equiv f(\beta)\cdot \mathtt{SRS}_0^{G_1}$ define $[P_L]_1 \equiv [f(\beta)-f(s)]_1$ define $[P_R]_1 \equiv [t(\beta)]_1$ compute $[P_R]_2 = \beta \cdot \mathtt{SRS}_0^{G_2} - \mathtt{SRS}_1^{G_2}$ check $e([P_L]_1, [1]_2) \stackrel{?}{=} e([P_R]_1, [P_R]_2)$

Table 1. KZG Protocol

We now address the complexity of the prover to compute (1.12). Using the equation, expanding it and writing out a few terms we see that

$$\begin{aligned} [t(\beta)]_1 = \; & a_1 \cdot \mathtt{SRS}_0^{G_1} \\ & + a_2\beta \cdot (\mathtt{SRS}_0^{G_1} + \beta^{-1} \cdot \mathtt{SRS}_1^{G_1}) \\ & + a_3\beta^2 \cdot (\mathtt{SRS}_0^{G_1} + \beta^{-1} \cdot \mathtt{SRS}_1^{G_1} + \beta^{-2} \cdot \mathtt{SRS}_2^{G_1}) \\ & \vdots \qquad\qquad \vdots \\ & + a_{n-1}\beta^{n-2} \cdot (\mathtt{SRS}_0^{G_1} + \beta^{-1} \cdot \mathtt{SRS}_1^{G_1} + \ldots + \beta^{-(n-2)} \cdot \mathtt{SRS}_{n-2}^{G_1}) \end{aligned} \tag{1.16}$$

Rearranging the terms in order of powers of β we get

$$\begin{aligned} [t(\beta)]_1 = \; & a_{n-1}\mathtt{SRS}_{n-2} + a_{n-2}\mathtt{SRS}_{n-3} + \ldots + a_3\mathtt{SRS}_2 + a_2\mathtt{SRS}_1 + a_1\mathtt{SRS}_0 \\ & + (a_{n-1}\mathtt{SRS}_{n-3} + a_{n-2}\mathtt{SRS}_{n-4} + \ldots + a_4\mathtt{SRS}_2 + a_3\mathtt{SRS}_1 + a_2\mathtt{SRS}_0)\beta \\ & \vdots \quad \vdots \\ & + (a_{n-1}\mathtt{SRS}_1 + a_{n-2}\mathtt{SRS}_0)\beta^{n-3} \\ & + (a_{n-1}\mathtt{SRS}_0)\beta^{n-2} \end{aligned} \tag{1.17}$$

which can be written as a matrix product

$$[t(\beta)]_1 = \begin{bmatrix} 1 & \beta & \beta^2 \dots & \beta^{n-2} \end{bmatrix} \cdot \begin{bmatrix} a_{n-1} & a_{n-2} & a_{n-3} & \dots & \dots & a_1 \\ 0 & a_{n-1} & a_{n-2} & \dots & \dots & a_2 \\ 0 & 0 & a_{n-1} & \dots & \dots & a_3 \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \dots & a_{n-1} \end{bmatrix} \cdot \begin{bmatrix} \texttt{SRS}_{n-2} \\ \texttt{SRS}_{n-3} \\ \texttt{SRS}_{n-4} \\ \vdots \\ \texttt{SRS}_0 \end{bmatrix} \tag{1.18}$$

One can view the eq (1.18) as a Matrix representation of the equation of equation 4 in [2]. Note that the Matrix is of Toeplitz form and the matrix multiplication can be done in $\mathcal{O}((n-1)\log(n-1))$ time.

2 KZG point proofs in Danksharding: Batch opening of the committed polynomial in roots of unity

In the case of KZG point proofs $[t(\beta_i)]_1$ where for the same polynomial $n-1$ opening proofs are constructed, $i = 0, 1, ..n-1$ batching is possible when the evaluations are on roots of unity $\beta_i = \omega^i$ where ω is a root of unity in the domain of size $|N|$. This is for example the case in Danksharding [3]. Substituting $\beta = \omega^i$ in (1.18) we get

$$[t(\omega^i)]_1 = \begin{bmatrix} 1 & \omega^i & \omega^{2i} \dots & \omega^{(n-2)i} \end{bmatrix} \cdot \begin{bmatrix} a_{n-1} & a_{n-2} & a_{n-3} & \dots & \dots & a_1 \\ 0 & a_{n-1} & a_{n-2} & \dots & \dots & a_2 \\ 0 & 0 & a_{n-1} & \dots & \dots & a_3 \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \dots & a_{n-1} \end{bmatrix} \cdot \begin{bmatrix} \texttt{SRS}_{n-2} \\ \texttt{SRS}_{n-3} \\ \texttt{SRS}_{n-4} \\ \vdots \\ \texttt{SRS}_0 \end{bmatrix} \tag{2.1}$$

and each of the batch proof can be organized as part of a vector

$$\begin{bmatrix} [t(\omega^0)]_1 \\ [t(\omega^1)]_1 \\ [t(\omega^2)]_1 \\ \vdots \\ [t(\omega^{n-2})]_1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & \dots & \dots & 1 \\ 1 & \omega & \omega^2 & \dots & \dots & \omega^{n-2} \\ 1 & \omega^2 & \omega^4 & \dots & \dots & \omega^{2(n-2)} \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ 1 & \omega^{(n-2)} & \omega^{2(n-2)} & \dots & \dots & \omega^{(n-2)^2} \end{bmatrix} \underbrace{\cdot \begin{bmatrix} a_{n-1} & a_{n-2} & a_{n-3} & \dots & \dots & a_1 \\ 0 & a_{n-1} & a_{n-2} & \dots & \dots & a_2 \\ 0 & 0 & a_{n-1} & \dots & \dots & a_3 \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \dots & a_{n-1} \end{bmatrix} \cdot \begin{bmatrix} \texttt{SRS}_{n-2} \\ \texttt{SRS}_{n-3} \\ \texttt{SRS}_{n-4} \\ \vdots \\ \texttt{SRS}_0 \end{bmatrix}}_{\text{precompute}} \tag{2.2}$$

We note that the batch proofs (2.2) are uniquely represented as a product of a DFT matrix, times multiplication by a Toeplitz matrix. The product (MSM) of the Toeplitz matrix with the group elements can be done in $\mathcal{O}((n-1)\log_2(n-1))$ and the DFT in another $\mathcal{O}((n-1)\log_2(n-1))$.

2.1 Batch opening at random field elements

However if each opening point were some random $\beta_i \in \mathbb{F}$ there is no DFT structure and batch opening of the same polynomial at k points will require repetition of (1.18) k times or in other words

$$\begin{bmatrix} [t(\beta_0)]_1 \\ [t(\beta_1)]_1 \\ [t(\beta_2)]_1 \\ \vdots \\ [t(\beta_{k-1})]_1 \end{bmatrix} = \begin{bmatrix} 1 & \beta_0 & \beta_0^2 & \dots & \dots & \beta_0^{n-2} \\ 1 & \beta_1 & \beta_1^2 & \dots & \dots & \beta_1^{n-2} \\ 1 & \beta_2 & \beta_2^2 & \dots & \dots & \beta_2^{n-2} \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ 1 & \beta_{k-1} & \beta_{k-1}^2 & \dots & \dots & \beta_{k-1}^{n-2} \end{bmatrix} \cdot \underbrace{\begin{bmatrix} a_{n-1} & a_{n-2} & a_{n-3} & \dots & \dots & a_1 \\ 0 & a_{n-1} & a_{n-2} & \dots & \dots & a_2 \\ 0 & 0 & a_{n-1} & \dots & \dots & a_3 \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \dots & a_{n-1} \end{bmatrix}}_{\text{precompute}} \cdot \begin{bmatrix} \texttt{SRS}_{n-2} \\ \texttt{SRS}_{n-3} \\ \texttt{SRS}_{n-4} \\ \vdots \\ \texttt{SRS}_0 \end{bmatrix} \tag{2.3}$$

while this retains the same structure as (2.2) it loses the NTT structure that gave a $n \log n$ advantage. This problem compounds, if we have many polynomials $t_r(X)$ and we are forced to repeat (2.3) r times. On top of all this, one must always have the polynomial in monomial basis for the whole discussion to work.

3 KZG evaluation proofs in Lagrange basis

In this section we ask the question if it is cheaper to produce evaluation proofs in the Lagrange basis. The evaluation form or the Lagrange basis is a more natural representation of data, and in order to go to the monomial basis one does an NTT, which costs $O(n \log n)$, which can be saved if there are a large number of polynomials. Our goal is to find a way of batching, analogous to the batching [4] and provide a single opening proof that covers all openings.

3.1 Lagrange basis and some identities

For this we define the Lagrange basis in a domain H spanned by the set of roots of unity $x_i \equiv \{1, \omega, \omega^2, \dots \omega^{n-1}\}$ as

$$L_{H,x_i}(X) = \frac{\prod_{k=0,k\neq i}^{n-1}(X - x_k)}{\prod_{k=0,k\neq i}^{n-1}(x_i - x_k)} \tag{3.1}$$

that satisfy the relations

$$L_{H,x_i}(X = x_j) = \delta_{ji} = \begin{cases} 1 \ , \ j = i \ \forall \ i,j = 0,1,\dots n-1 \\ 0 \ , \ \text{otherwise} \end{cases} \tag{3.2}$$

Note that each basis polynomial (3.1) is at most of degree $n-1$. We define a (Low Degree Extension) LDE of any given set of n points v_i in terms of the Lagrange polynomial (3.1)

$$F(X) = \sum_{i=0}^{n-1} v_i \cdot L_{H,x_i}(X) \tag{3.3}$$

which is a polynomial of at most $deg \leq n-1$. This is referred to as the evaluations form of the polynomial. We recollect that the vanishing polynomial in a domain H is defined as the monic

$$v_H(X) = X^H - 1 = \prod_{k=0}^{n-1}(X - x_k) \ \forall \ x_k \in H \tag{3.4}$$

that vanishes only on all points within H, and is non-vanishing outside. The formal derivative of the vanishing polynomial (independent of H) is defined as

$$V'_H(X,Y) = \begin{cases} \frac{V_H(X)-V_H(Y)}{X-Y} & X \neq Y \\ |H|X^{|H|-1} & X = Y \end{cases} \tag{3.5}$$

Since $V_H(X)\big|_H = 0$, the derivative $V'_H(X,Y)\big|_H$ vanishes on all square $H \times H$ except the diagonal where the non vanishing entries are of degree $|H|-1$.

$$\begin{aligned} \frac{\partial}{\partial X}(V_H(X))\Big|_{X=x_i} &= \frac{\partial}{\partial X}\prod_{k=0}^{n-1}(X-x_k)\Big|_{X=x_i} \\ &= \sum_{r=0}^{n-1}\prod_{k=0,k\neq r}^{n-1}(X-x_k)\Big|_{X=x_i} \\ V'_H(x_i) &= \prod_{k=0,k\neq i}^{n-1}(x_i-x_k) \end{aligned} \tag{3.6}$$

Using the above relations (3.6) and (3.4) one can rewrite the Lagrange polynomial as

$$L_{H,x_i}(X) = \frac{v_H(X)}{X-x_i}\cdot\frac{1}{v'_H(x_i)} \tag{3.7}$$

Thus the evaluation representation (3.3) takes the form

$$F(X) = \sum_{i=0}^{n-1}\frac{v_H(X)}{X-x_i}\cdot\frac{v_i}{v'_H(x_i)} \tag{3.8}$$

3.2 Divisibility in Lagrange basis

Similar to opening proofs in monomial basis, we would like to compute the quotient polynomial at some random point $\beta \in \mathbb{F}$. Using (3.8)

$$F(\beta)-F(X) = \sum_{i=0}^{n-1}\left[\frac{v_H(\beta)}{\beta-x_i}-\frac{v_H(X)}{X-x_i}\right]\cdot\frac{v_i}{v'_H(x_i)} \tag{3.9}$$

The term in [] on the RHS can be expressed as

$$\frac{v_H(\beta)}{\beta-x_i}-\frac{v_H(X)}{X-x_i} = \frac{-x_i\cdot(\beta^n-X^n)+X\cdot\beta\cdot(\beta^{n-1}-X^{n-1})-(\beta-X)}{(X-x_i)(\beta-x_i)} \tag{3.10}$$

using

$$X^{r-1}-Y^{r-1} = (X-Y)\cdot X^{r-2}\cdot\sum_{i=0}^{r-2}\left(\frac{Y}{X}\right)^i \quad \forall r \geq 2 \tag{3.11}$$

we get

$$\begin{aligned} \frac{v_H(\beta)}{\beta-x_i}-\frac{v_H(X)}{X-x_i} &= \frac{-x_i\cdot(\beta^n-X^n)+X\cdot\beta\cdot(\beta^{n-1}-X^{n-1})-(\beta-X)}{(X-x_i)(\beta-x_i)} \\ &= \frac{(\beta-X)}{(X-x_i)(\beta-x_i)}\left(-x_i\cdot\beta^{n-1}\sum_{j=0}^{n-1}\left(\frac{X}{\beta}\right)^j+X\cdot\beta^{n-1}\sum_{j=0}^{n-2}\left(\frac{X}{\beta}\right)^j-1\right) \end{aligned} \tag{3.12}$$

Substituting this equation in (3.9) we get

$$\frac{F(\beta)-F(X)}{(\beta-X)}=\sum_{i=0}^{n-1}\frac{v_i}{(X-x_i)(\beta-x_i)v'_H(x_i)}\left(-x_i\cdot\beta^{n-1}\sum_{j=0}^{n-1}\left(\frac{X}{\beta}\right)^j+X\cdot\beta^{n-1}\sum_{j=0}^{n-2}\left(\frac{X}{\beta}\right)^j-1\right) \tag{3.13}$$

which shows that $F(\beta)-F(X)$ in evaluation form is indeed divisible by $(\beta-X)$.

To write the above equation using the SRS and Lagrange polynomials evaluated in secret scalar

3.3 A simple protocol for opening a polynomial in multiple points (not at roots of unity) in Lagrange basis

In this section we will consider only polynomials in Lagrange form. Thus in order to commit them, we need the SRS in the Lagrange basis. We deonote commitment in Lagrange basis as $[F(s)]_1=\sum_{i=0}^{n-1}v_i\cdot\texttt{SRS}_i$, where s is the secret scalar which cannot be extracted from the SRS due to the DLOG assumption.

This work is inspired by [4] where there are multiple polynomials $F_i(X)$ with all the evaluations $F_i(z_i)=y_i$ are known apriori. The protocol checks that each of the evaluations are indeed that of the corresponding committed polynomial by combining all the quotient polynomials with random coefficients. Then a KZG opening check is applied to the final quotient polynomial, that proves the opening of all the committed polynomials.

Here we restrict to the situation that the verifier is simply interested in checking if the prover committed to a given polynomial $F(X)$ and if multiple openings of the same polynomial at random field elements can be batched together with KZG opening of one effective quotient polynomial. The main difference is that the verifier has to rely on the prover to query all the evaluations. Nevertheless the trick used in [4] applies here as well.

The main idea of the protocol is to open a given committed polynomial $F(X)$ in Lagrange basis, in multiple points (two here for simplicity) and combine the KZG quotient polynomials with random coefficients. The resultant quotient polynomial $T(X)$ can be opened at a point, and successful check of this opening, proves all the openings of the original polynomial $F(X)$. i.e a sum of polynomials is indeed a polynomial, even if one opening in the sum gives a rational function the result of the sum is rational.

We denote in color coding: challenges, computed by prover, computed by verifier, public. The prover protocol is

1. The prover commits to the claimed evaluation form polynomial

$$F(X)=\sum_{i=0}^{n-1}v_i\cdot\mathbb{L}_i(X) \tag{3.14}$$

2. The prover receives $\beta\leftarrow H(([F(s)]_1||0))$, $\gamma\leftarrow H(([F(s)]_1||1))$

3. The prover evaluates $F(\gamma)=\sigma, F(\beta)=\zeta$ and sends it to verifier.

4. The prover recieves $r \leftarrow H([F(s)]_1, \sigma, \zeta)$

5. the prover computes the quotient polynomial

$$T(X) = \frac{F(X) - \zeta}{X - \beta} + r \cdot \frac{F(X) - \sigma}{X - \gamma} \tag{3.15}$$

and commits to $[T(s)]_1$.

6. The prover gets random challenge $\tau \leftarrow H([F(s)]_1, [T(s)]_1, \sigma, \zeta)$.

7. Prover computes

$$T_1(X) = F(X) \cdot \left(\frac{1}{\tau - \beta} + \frac{r}{\tau - \gamma} \right) \tag{3.16}$$

and $\rho = T_1(\tau) - T(\tau)$ and uses it to compute the KZG opening proof

$$[\pi_{KZG}]_1 = \left[\frac{T_1(s) - T(s) - \rho}{s - \tau} \right]_1 \tag{3.17}$$

8. The proof elements are

$$\pi = ([F(s)]_1, [T(s)]_1, [\pi_{KZG}]_1, \sigma, \zeta) \tag{3.18}$$

The verifier does

1. The verifier knows r, β, γ, τ and uses it to compute $\mu_1 = \left(\frac{1}{\tau - \beta} + \frac{r}{\tau - \gamma} \right)$. With this the verifier computes the commitment

$$[T_1(s)]_1 \equiv [F(s)]_1 \cdot \mu_1 \tag{3.19}$$

2. Pairing Check: The verifier independently computes $\rho = \left(\frac{\zeta}{\tau - \beta} + \frac{r \cdot \sigma}{\tau - \gamma} \right)$ and checks [1]

$$e([\pi_{KZG}]_1, [s - \tau]_2) = e([T_1(s) - T(s) - \rho]_1, [1]_2) \tag{3.20}$$

If the prover did not commit to the correct $F(X)$ or attempts to fudge (3.15) by choosing say different $F(X)$ for each term. Then the commitment $[T_1(s)]_1$ (3.19) computed by the verifier will give a different result, following which then (3.20) will fail.

Acknowledgments

We stand on the shoulders of giants. We thank Suyash Bagad and Yuval Domb for discussions.

[1] Where s is just a formal notation for the secret scalar. If the SRS string in $G2$ is $\{g_0, g_1, \ldots\}$ then, $[s - \tau]_2 = g_0 - \tau \cdot g_0$

References

[1] A. Kate, G. M. Zaverucha, and I. Goldberg, *Constant-size commitments to polynomials and their applications*, in *Advances in Cryptology - ASIACRYPT 2010* (M. Abe, ed.), (Berlin, Heidelberg), pp. 177–194, Springer Berlin Heidelberg, 2010.

[2] D. Khovratovich and D. Feist, "Fast amortized kzg proofs." https://github.com/khovratovich/Kate/blob/master/Kate_amortized.pdf.

[3] Y. Domb, "A mathematical theory of danksharding." https://github.com/ingonyama-zk/papers/blob/20d4a951a497a8e580573eade1e0c330475d3ccf/danksharding_math.pdf.

[4] D. Feist, "Pcs multiproofs using random evaluation." https://dankradfeist.de/ethereum/2021/06/18/pcs-multiproofs.html).