



Haym @SalomonCrypto

Oct 21 • 23 tweets • [SalomonCrypto/status/1583542847759753217](https://twitter.com/SalomonCrypto/status/1583542847759753217)

Tr

## (1/22) KZG Commitments Part 3: Verify

After the prover has committed to the data, the verifier issued a challenge: "prove yourself based on this value." The prover has obliged, generating and transmitting a proof... now it's time to verify.

The conclusion of the KZG scheme.

### KZG Polynomial Commitments

Red: Secret   Green: Public   Blue: Elliptic Curve (Public)   Pink: Data (Varies)

#### Step 3: Verify

Prover	Verifier
1) Commit to $[f(S)] = C$	2a) Request proof of $z$
2b) Calculate $f(z)$ , $[h(S)] = H$	3) Verify $e(H, [S - z]) = e(C - f(z), [1])$

#### What is $e(C - f(z)) = e(H, [S - z])$ ?

Goal: verify that the prover actually did the polynomial division to create  $h(x)$ , and that  $[h(S)]$  the commitment of  $h(x)$  at  $S$ .

##### What is $h(x)$ ?

$$h(x) = \frac{f(x) - f(z)}{x - z}$$

$h(x)$  is a polynomial created by the prover through the (relatively) simple process of polynomial division.

##### Curve Points

$[f(S)]$  and  $[h(S)]$  are points on an elliptic curve

##### Elliptic Curve Pairings

Pairings are functions that act as the elliptic point-equivalent of multiplication

$$h(x) = \frac{f(x) - f(z)}{x - z} \Rightarrow (x - z) \cdot h(x) = f(x) - f(z) \Rightarrow e([x - z], [h(x)]) = e([f(x)] - [f(z)], [1])$$
$$e([S - z], H) = e(C - f(z), [1])$$

Identify point, known by all (equivalent to multiplying by 1)

Provided by prover:  $e([S - z], H)$

Calculated by verifier:  $e(C - f(z), [1])$

**Conclusion:**  $e([S - z], H) = e(C - f(z), [1])$

(2/22) We're in the home stretch! This is the last piece we need to understand KZG commitments.

I am assuming that you've read the threads before this one. If you missed them, no worries I've linked them below.

**Haym**  
@SalomonCrypto · [Follow](#)



Replying to @SalomonCrypto

(2/18) Preface:

This is part of a series on elliptic curve cryptography and its applications for [@ethereum](#). This is simplified to a MINIMAL level, aiming at ~high-school math.

[@VitalikButerin](#) [@dankrad](#) [@danboneh](#) [@chaseklvk](#), if you read this, I'm sorry for what I did to the math.


4:03 PM · Oct 15, 2022 

 20  Reply  Copy link

[Read 2 replies](#)

(3/22) A polynomial commitment scheme allows one party to publicly commit to a specific dataset without giving away any info about the underlying data.

Once a commitment has been calculated, the data is locked in; any changes to the data will result in invalid proof generation.

 **Haym**  
@SalomonCrypto · [Follow](#)

(1/17) Cryptography Basics: Polynomial Commitments

Creating unique digital fingerprints is core to cryptography. We use tools like hashing to provide cryptographic-certainty without revealing any info.

But hashing is so destructive, is there a better way to commit to data?

### Polynomial Commitments

Encode data to polynomial form by applying a Lagrange Interpolation

Index	Letter	Value
1	S	83
2	T	84
3	U	85
4	A	65
5	R	82
6	T	84


Derive a new, non-sensitive point on the polynomial

**PRIVATE** (points 1-6) | **PUBLIC** (point 7)

Post the commitment to this specific polynomial (and therefore to the original data)

**(4.5, 69.5)**

1:50 AM · Oct 16, 2022

 [Read the full conversation on Twitter](#)

370 ❤️ Reply Copy link


[Read 15 replies](#)

(4/22) KZG commitments use the magic of elliptical curve cryptography to create something special: a commitment that can be checked at ANY position without revealing any other information.


The prover commits, then the verifier can request a proof around any data point.

(5/22) The scheme has two types of preparation: hiding a secret number  $S$  in an elliptic curve (creating a SRS) and building a Lagrange polynomial from the data.

Using these two pieces, the prover generates a commitment.



**Haym**  
@SalomonCrypto · [Follow](#)



(1/23) KZG Commitments Part 1: Commit

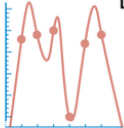
Our goal: 1) prove we are committed to specific data and 2) allow others to verify specific points within that dataset.

Before we get started, we need to prepare the data and elliptic curve.... Then, it's time for some magic!

### KZG Polynomial Commitments

Red: Secret   Green: Public   Blue: Elliptic Curve (Public)   Pink: Data (Varies)

#### Step 0: Preperation


shared between all commitments		specific to each commitment	
Trusted Setup		Data	
Secret Number $S$	Elliptic Curve: $y^2 = x^3 + ax + b$	Plaintext Data STUART	UTF-8 Encoding: 83, 84, 85, 65, 84
$f(S^0) = [S^0] = S^0 G$ $f(S^1) = [S^1] = S^1 G$ $f(S^2) = [S^2] = S^2 G$ $f(S^3) = [S^3] = S^3 G$ $\vdots$ $f(S^n) = [S^n] = S^n G$	Public Structured Reference String (SRS)		Lagrange Polynomial: $f(x) = a_0 x^0 + a_1 x^1 + a_2 x^2 + a_3 x^3 + a_4 x^4 + a_5 x^5$




#### Step 1: Commit

**Commitment:**  $[f(S)]$  — single value that serves as the polynomial commitment

$$[f(S)] = [a_0 S^0 + a_1 S^1 + a_2 S^2 + a_3 S^3 + a_4 S^4 + a_5 S^5]$$
$$[f(S)] = a_0 [S^0] + a_1 [S^1] + a_2 [S^2] + a_3 [S^3] + a_4 [S^4] + a_5 [S^5]$$

1:06 AM · Oct 19, 2022

[Read the full conversation on Twitter](#)

 194    Reply    Copy link

[Read 6 replies](#)

(6/22) The commitment is a specific value around which the future KZG proofs will be built.

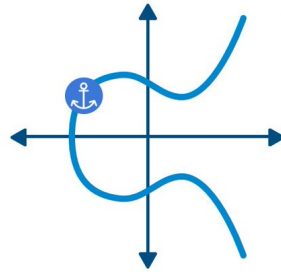
By virtue of the scheme, the commitment binds the prover to that data; if the data changes, the prover will no longer be able to generate valid proofs.

## KZG Commitment Scheme

*First, the prover commits to data by providing a point on the elliptic curve. If the data changes, the prover cannot create valid proofs.*

**Prover**


1)   
Commit



**Verifier**


(7/22) The verifier (the party challenging the prover) knows at least one piece of data in the committed data set, and so requests a proof around data point z.

The prover calculates  $[h(S)]$  and  $f(z)$ , returning the values to the verifier.



Haym

@SalomonCrypto · Follow



(1/13) KZG Commitments Part 2: Open

Previously, we committed to our data by evaluating a polynomial using an elliptic curve. Now, it's time to test that commitment; it's time for the verifier to see if the prover knows a specific piece of data.

It's time to open the commitment.

### KZG Polynomial Commitments

Red: Secret   Green: Public   Blue: Elliptic Curve (Public)   Pink: Data (Varies)

#### Step 2: Open

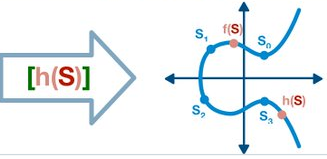
Prover	Verifier
Data → Polynomial → Commitment	Step 0 & Step 1
Step 2a	Given commitment, create proof for z
With z, calculate $[h(S)]$ and $f(z)$ and return the values $\langle z, [h(S)], f(z) \rangle$	Step 2b

Proof:  $\langle z, [h(S)], f(z) \rangle$


#### Caclulating $[h(S)]$


$h(x)$  is a polynomial that can be generated by an honest prover with quick and (relatively) simple math

$$h(x) = \frac{f(x) - f(z)}{x - z}$$





2:27 AM · Oct 21, 2022






Read the full conversation on Twitter

 105

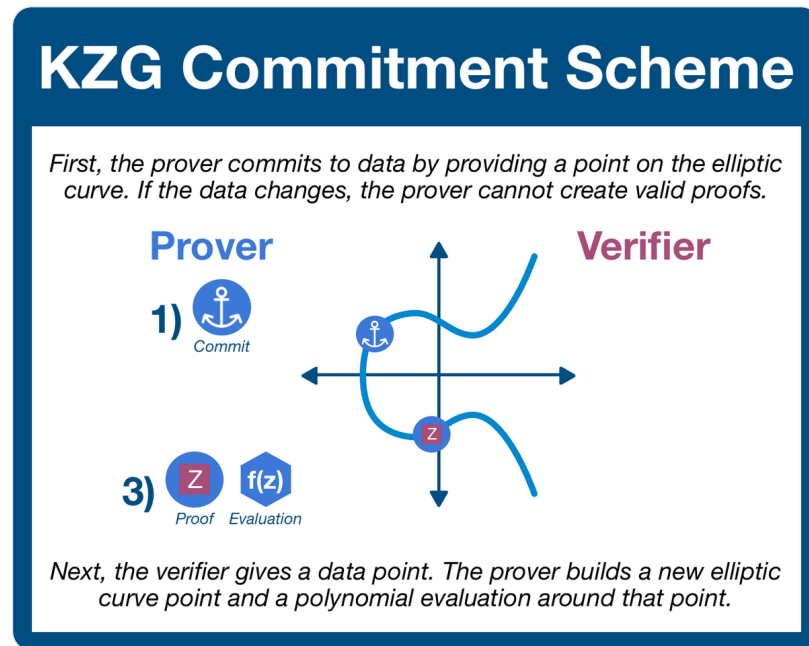
 Reply

 Copy link

Read 2 replies


(8/22) Just like our original commitment is simply  $f(x)$  evaluated through the elliptic curve at  $S$ , our proof is going to be an evaluation of  $h(x)$  at  $S$ ... ALSO an elliptic curve point!

The question: how can we use our  $f(S)$ ,  $f(z)$ ,  $h(S)$  and  $z$  to prove our prover is being honest?




(9/22) The core of KZG commitment is this " $h(x)$ " polynomial. If the prover is honest and is still committed to the original data, this is a simple polynomial to derive and to evaluate.

And so, we are simply going to test this polynomial by having the prover evaluate it at  $S$ .



**Haym**  
@SalomonCrypto · Follow




Replying to @SalomonCrypto

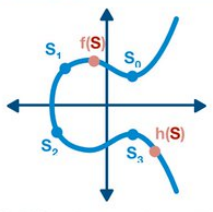
(11/13) Because the prover has built the polynomial, it is very easy for them to create the new polynomial  $h(x)$ .  $h(x)$  is the quotient polynomial built off of our proof data  $z$ .

### Caclulating $[h(S)]$


$h(x)$  is a polynomial that can be generated by an honest prover with quick and (relatively) simple math




$$h(x) = \frac{f(x) - f(z)}{x - z}$$





$[h(S)]$  is also a point on the elliptic curve

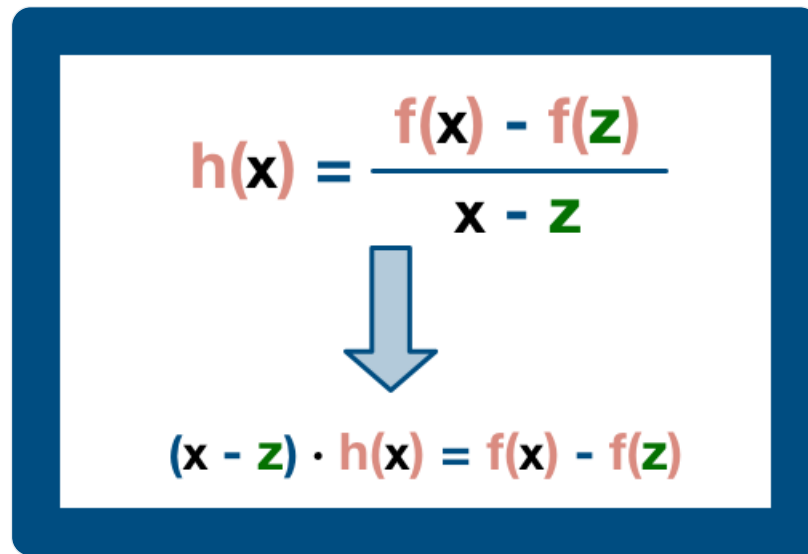
2:27 AM · Oct 21, 2022

 4 Reply Copy link

[Read 1 reply](#)

(10/22) Remember, the verifier may or may not have access to the full original data set and therefore the verifier cannot build the polynomial and check the provers value.

Instead, we are going to check a slightly different equation:


$$h(x) = \frac{f(x) - f(z)}{x - z}$$

↓

$$(x - z) \cdot h(x) = f(x) - f(z)$$

(11/22) But remember, we aren't working with standard functions; we are working with elliptic curves. And so, we have a problem.

We have seen that we can easily add and subtract two points on an elliptic curve, but we haven't seen multiplication.


(12/22) In fact, you CAN'T multiply two elliptic curve points, at least in any coherent way... not in any way that will non-destructively obscure data in the way we need.


Fortunately, we have a solution: elliptic curve pairings.



(13/22) Pairings are perhaps the most advanced math you'll ever come across. Super exciting, cutting edge research-type stuff.

For our purposes, they are actually pretty easy: imagine pairings as the single use, irreversible version of elliptic curve point multiplication.

**Haym**  
@SalomonCrypto · [Follow](#)



(1/24) Degen's Handbook: A Practical Guide to Elliptic Curve Pairings

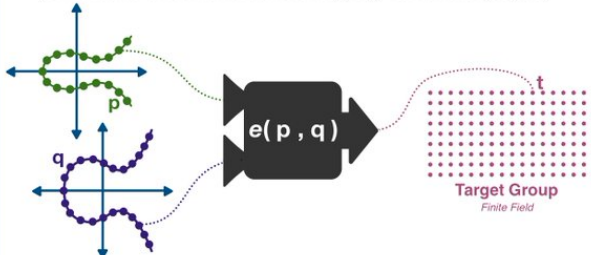
Magic is real, elliptic curve pairings are proof. Through incredibly advanced math, pairings allow us to multiply two polynomials... through elliptic curves!

Here's all you need to know.

**Elliptic curve pairings are the elliptic point equivalent of multiplication**

$$e(p, q) = t \approx p * q = t$$

An elliptic curve pairing is a function that takes a pair of elliptic curve points and returns an element of some other group, called the target group




Think of a pairing as a black box that takes elliptic points. Pairings cannot be used consecutively; the target group points don't match the input points




Elliptic curve pairings are bilinear, holding to the following property:

$$e(p+r, q) = e(p, q) * e(r, q)$$
$$e(p, q+r) = e(p, q) * e(p, r)$$

Translation: you can pull additive component out of a pairing by multiplication

6:50 PM · Oct 20, 2022

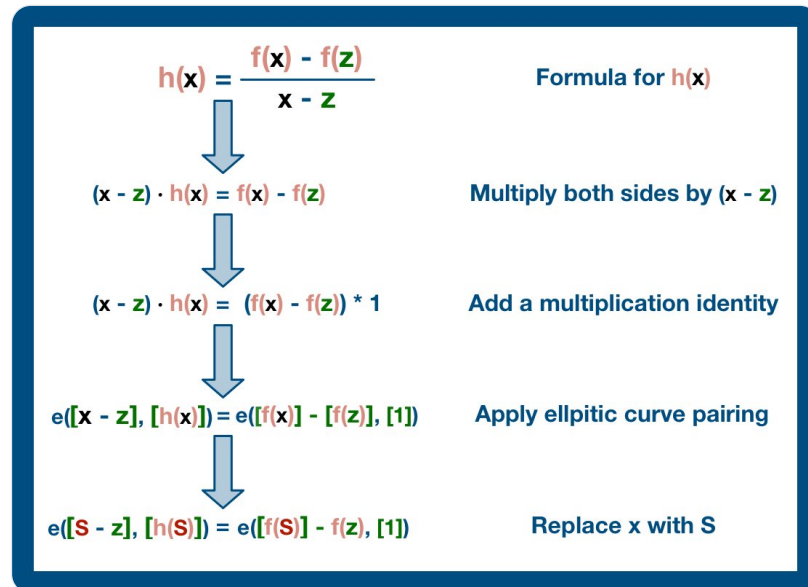
 [Read the full conversation on Twitter](#)

 353  Reply  Copy link

[Read 13 replies](#)

(14/22) Now we have all the pieces we need to complete our KZG proof verification; let's finish creating verification conditions.

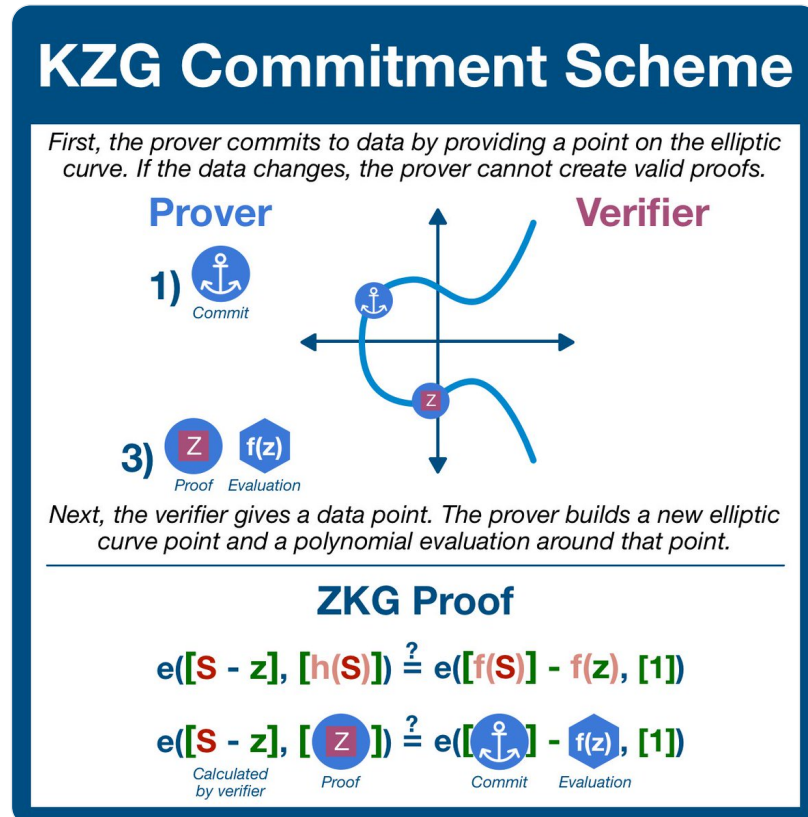
At the end of this process we will have our KZG verification expression. All we need to is evaluate each side and make sure they are equivalent.



(15/22) The prover supplies 3 of 4 terms needed in the verification expression:

- $[f(S)]$ , the commitment to the data
- $[h(S)]$ , a commitment to  $h(S)$ , where  $h(x)$  is polynomial constructed around  $z$
- $f(z)$ , the original polynomial evaluated at  $z$

The verifier can calculate  $[S - z]$ .



(16/22) And so, the verifier computes both sides of the equation and compares the results. If the results match, then we can be mathematically certain the prover was honest!

Need to unpack that one more time?

(17/22) The purpose of the verification is to work out whether the prover successfully divided the polynomial  $f(x)$  by  $(x-z)$ . If any of the data was altered, the prover won't have access to  $f(x)$ . He'll be able to generate a polynomial based off the new data, but it won't verify.

(18/22) Think like this:

Step 1: the prover committed to a dataset by creating a polynomial and evaluating it using the SRS

Step 2: the verifier challenges the prover by giving a data point within the original data set

(continued)

(19/22)

Step 3: the prover comes back with two items, a curve point chosen using the prover's requested data point and an evaluation

Step 4: the prover uses a pairing to double check that these two items were correctly calculated, using the commitment as a reference.

(20/22) That's it, the entire KZG commitment scheme! Once the verifier has run both sides of the equation and checked for equality, the process is complete.

We've successfully verified both the original commitment and the specific data point within the committed dataset.

(21/22) In fact, KZG commitments are so powerful that we can actually verify more than one point at a time.

A KZG multiproof can prove huge amounts of data points (million+) just by providing a single group element.

But that's a topic for some other theadoor.


(22/22) And so, there you have it: The KZG Polynomial Commitment Scheme!

Now, let's go find somewhere to put them to use...




Like what you read? Help me spread the word by retweeting the thread (linked below).

Follow me for more explainers and as much alpha as I can possibly serve.



**Haym**  
@SalomonCrypto · [Follow](#)



(1/22) KZG Commitments Part 3: Verify

After the prover has committed to the data, the verifier issued a challenge: "prove yourself based on this value." The prover has obliged, generating and transmitting a proof... now it's time to verify.

The conclusion of the KZG scheme.

### KZG Polynomial Commitments

Red: Secret   Green: Public   Blue: Elliptic Curve (Public)   Pink: Data (Varies)

#### Step 3: Verify

Prover	Verifier
1) Commit to $[f(S)] = C$	2a) Request proof of $z$
2b) Calculate $f(z)$ , $[h(S)] = H$	3) Verify $e(H, [S - z]) = e(C - f(z), [1])$

#### What is $e(C - f(z)) = e(H, [S - z])$ ?

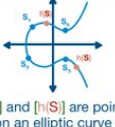
Goal: verify that the prover actually did the polynomial division to create  $h(x)$ , and that  $[h(S)]$  the commitment of  $h(x)$  at  $S$ .

##### What is $h(x)$ ?

$$h(x) = \frac{f(x) - f(z)}{x - z}$$

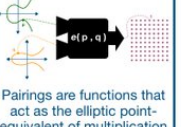
$h(x)$  is a polynomial created by the prover through the (relatively) simple process of polynomial division.

##### Curve Points



$[f(S)]$  and  $[h(S)]$  are points on an elliptic curve


##### Elliptic Curve Pairings






Pairings are functions that act as the elliptic point-equivalent of multiplication

$$h(x) = \frac{f(x) - f(z)}{x - z} \Rightarrow (x - z) \cdot h(x) = f(x) - f(z) \Rightarrow e[x - z], [h(x)] = e[f(x) - f(z)], [1]$$
$$e([S - z], H) = e(C - f(z), [1])$$

7:36 PM · Oct 21, 2022

[Read the full conversation on Twitter](#)

 3    Reply    Copy link

[Read 2 replies](#)

...