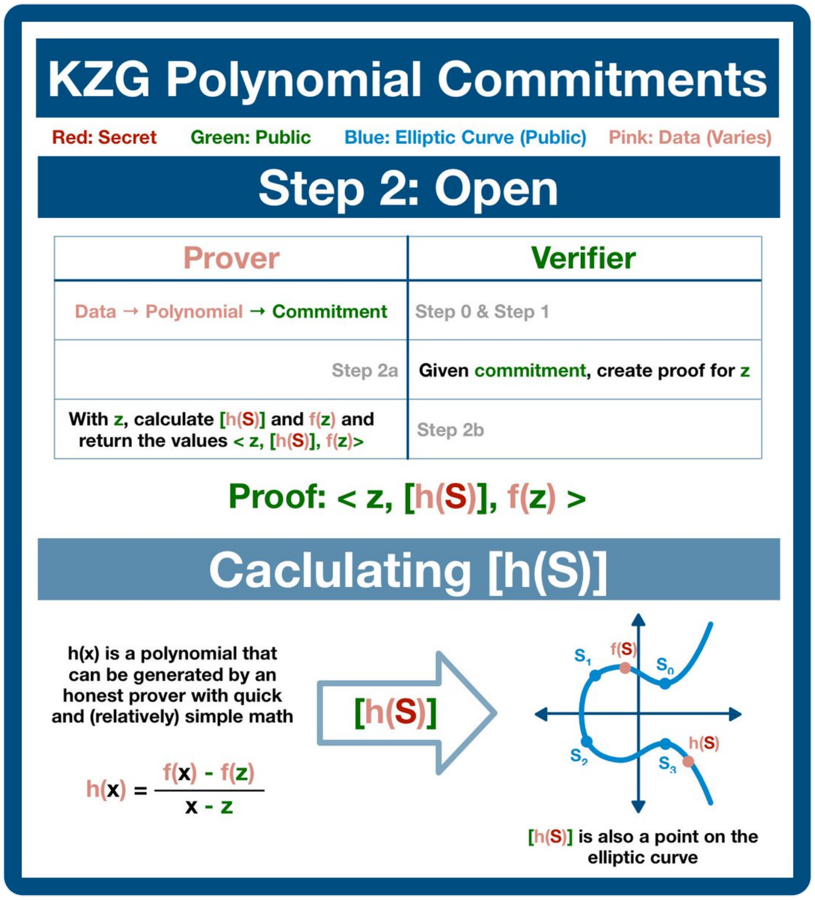
Haym @SalomonCrypto
Oct 21 • 14 tweets • SalomonCrypto/status/1583283784761233408

(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 < z, [h(S)], f(z)>	Step 2b

Proof: < z, [h(S)], f(z) >

Caclulating [h(S)]

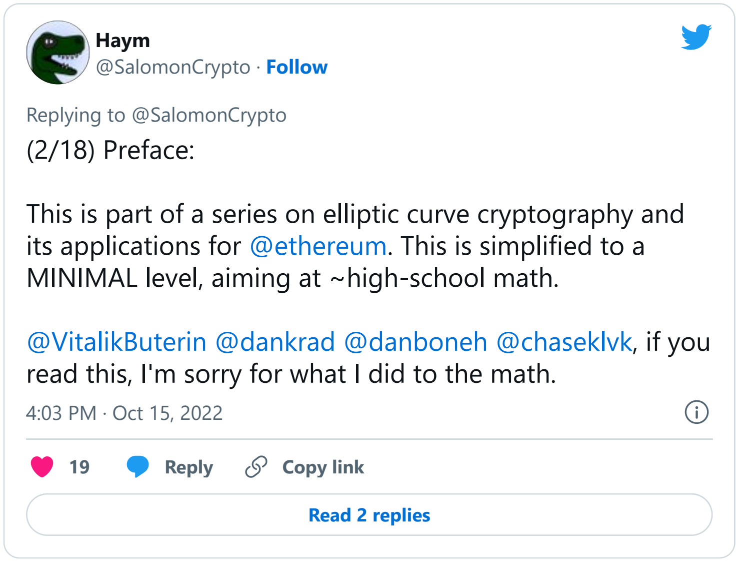
h(x) is a polynomial that can be generated by an honest prover with quick and (relatively) simple math

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

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

(2/13) The deeper we go, the more we are forced to simplify. Nevertheless, let's solider on!

All you need is high-school math and a willingness to read the images and linked tweets.



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

(3/13) 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.

(4/13) A KZG polynomial commit can be "opened" with any data point without revealing the entire dataset.

Let's say I am thinking of 10 words. I write them each down on different pieces of paper. If you guess correct, i ONLY show you the correct 1 (keeping the other 9 hidden).

(5/13) First, we generate a polynomial from our data using a mathematical process called a Lagrange Interpolation.

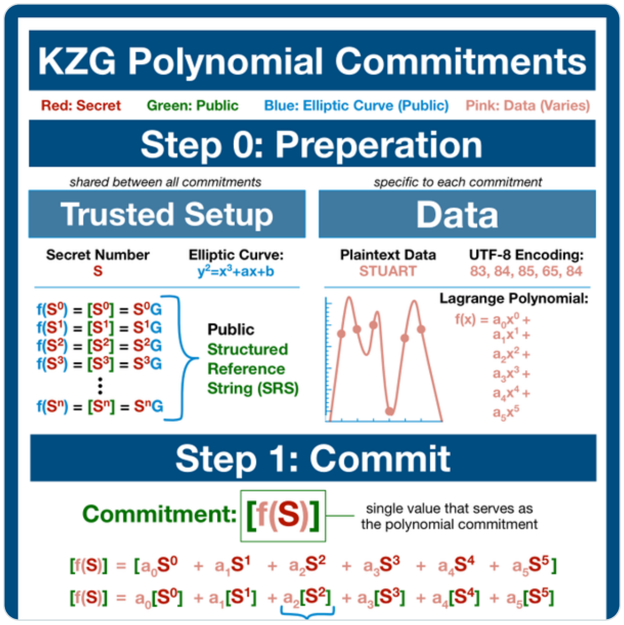
Then, we calculate the value of this polynomial evaluated with a secret number S (hidden within an elliptic curve).

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
$f(S^0) = [S^0] = S^0G$
$f(S^1) = [S^1] = S^1G$
$f(S^2) = [S^2] = S^2G$
$f(S^3) = [S^3] = S^3G$
$\vdots$
$f(S^n) = [S^n] = S^nG$
Public
Structured
Reference
String (SRS)
Lagrange Polynomial:
$f(x) = a_0x^0 + a_1x^1 + a_2x^2 + a_3x^3 + a_4x^4 + a_5x^5$
Step 1: Commit
Commitment: $\boxed{[f(S)]}$
single value that serves as the polynomial commitment
$[f(S)] = [a_0S^0 + a_1S^1 + a_2S^2 + a_3S^3 + a_4S^4 + a_5S^5]$
$[f(S)] = a_0[S^0] + a_1[S^1] + \underbrace{a_2[S^2]} + a_3[S^3] + a_4[S^4] + a_5[S^5]$

1:06 AM · Oct 19, 2022

Read the full conversation on Twitter

191 Reply Copy link

Read 6 replies

(6/13) Once the commitment is calculated, the prover is bound to this specific data. The scheme is going to pivot around this specific value, derived from this specific polynomial.

A change to even a single bit will result in a brand new polynomial (and therefore commit value).

(7/13) < PAUSE >

At this point our graphic has become too cluttered to be useful, so we will simplify in the name of readability. Nothing has changed, we are still working in modular arithmetic.

< RESUME >

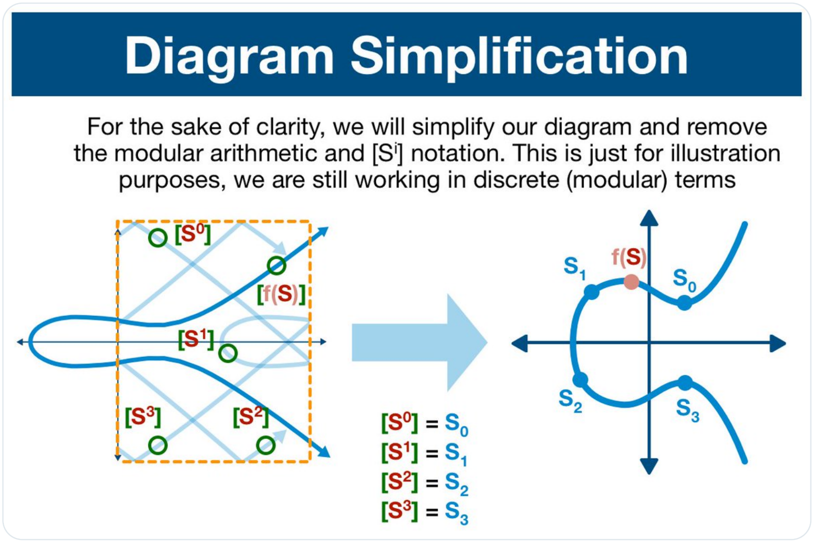


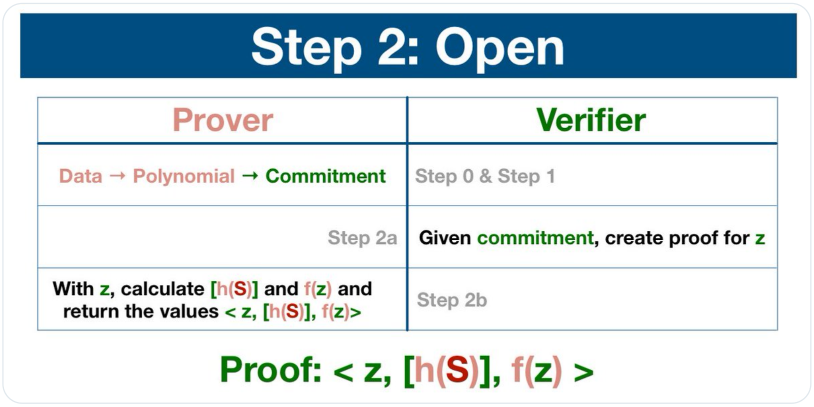
Diagram Simplification
For the sake of clarity, we will simplify our diagram and remove the modular arithmetic and [S^i] notation. This is just for illustration purposes, we are still working in discrete (modular) terms
[S⁰]
[f(S)]
[S¹]
[S³]
[S²]
[S⁰] = S₀
[S¹] = S₁
[S²] = S₂
[S³] = S₃
S₁
f(S)
S₀
S₂
S₃

(8/13) Now that the prover has been bound to this specific set of data, we can test the commitment by opening a proof at singular piece of data.

The verifier, an independent entity who may or may not have access to the entire dataset, now chooses a piece of data to verify.

(9/13) Let z be the piece of data the verifier is going to test... and so he sends z over to the prover.

First, the prover calculates f(z). This is very easy for the prover; he has already built the polynomial in order to generate the commitment. He simply runs z through.



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 < z, [h(S)], f(z)>	Step 2b

Proof: < z, [h(S)], f(z) >

(10/13) < Second wall break to paper over some math >

tl;dr you can divide polynomials. It's not a particularly complicated operation, in fact it (can) look like long division.

CUEMATH

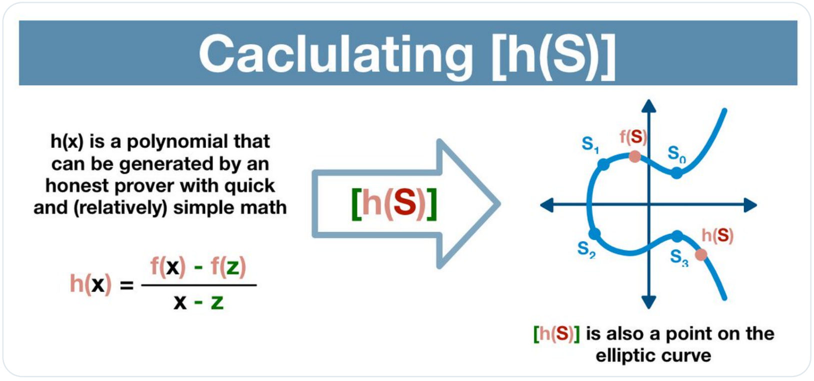
Dividing Polynomials - Definition, Synthetic Division, Long Division, E…

Dividing polynomials is an arithmetic operation where we divide a polynomial by another polynomial, generally with a lesser degree as compared to the dividend. Learn everything you need to know about…

https://www.cuemath.com/algebra/dividing-polynomials/

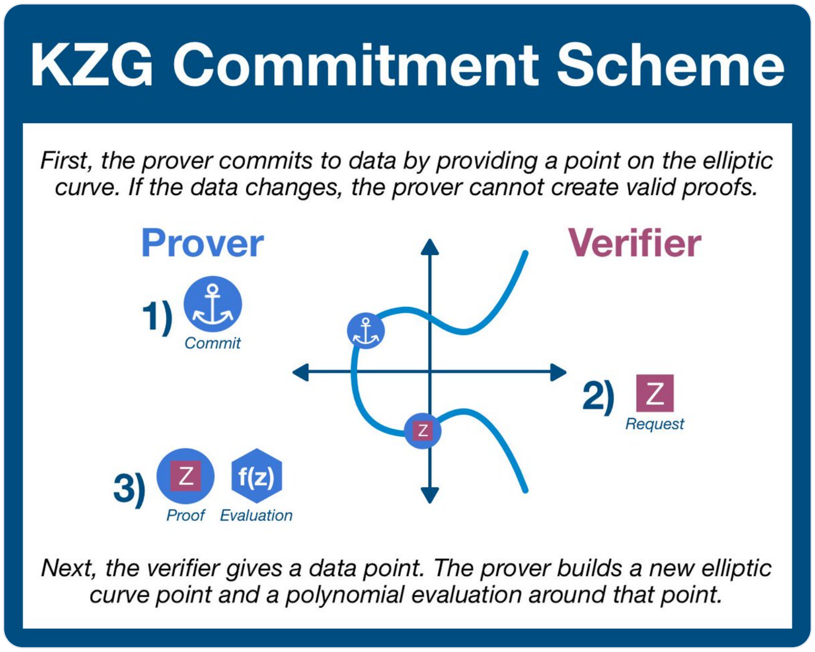
< Back to the thread >

(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)]
S_1
f(S)
S_0
S_2
S_3
h(S)
[h(S)] is also a point on the elliptic curve

(12/13) This is it one more time, as simplified as possible.

The important takeaway: the verifier provides a value (a piece of data in the dataset to which the prover is committed) around which the prover is going to create an elliptic curve point and one other value.


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
Verifier
1) Commit
2) Z Request
3) Z Proof
f(z) Evaluation
Next, the verifier gives a data point. The prover builds a new elliptic curve point and a polynomial evaluation around that point.

(13/13) And that's it! Our commitment is open and the prover has created all the pieces that the verifier needs to verify the proof!

Stay tuned for part 3, when we use these piece to unequivocally prove that the prover is staying truthful and fulfilling its commitment.

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

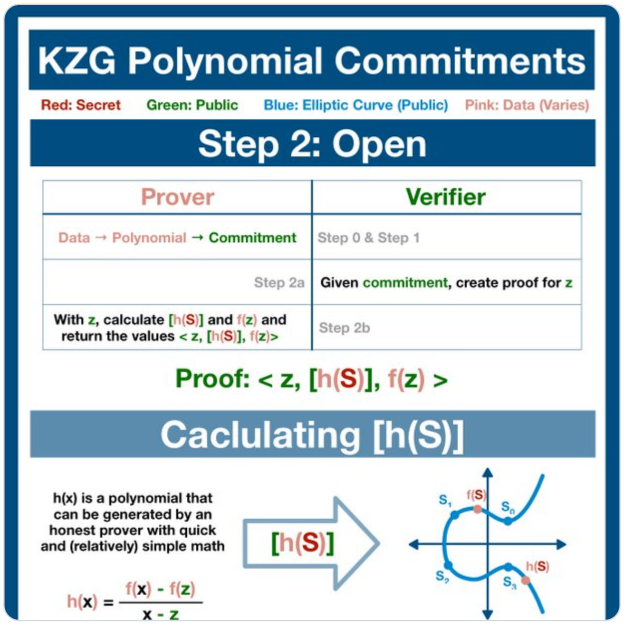
Follow me for more explainers and as much alpha as I can possibly serve.

Haym
@SalomonCrypto · **Follow**

(1/13) 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 < z, [h(S)], f(z)>	Step 2b

Proof: < z, [h(S)], f(z) >

Caclulating [h(S)]

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

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

2:27 AM · Oct 21, 2022

Read the full conversation on Twitter

4 Reply Copy link

Read 1 reply

• • •