



Haym @SalomonCrypto

Oct 19 • 24 tweets • [SalomonCrypto/status/1582538525517369344](https://twitter.com/SalomonCrypto/status/1582538525517369344)

Tr

(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

Trusted Setup

Secret Number
S

Elliptic Curve:
 $y^2=x^3+ax+b$

$f(S^0) = [S^0] = S^0G$
 $f(S^1) = [S^1] = S^1G$
 $f(S^2) = [S^2] = S^2G$
 $f(S^3) = [S^3] = S^3G$
 \vdots
 $f(S^n) = [S^n] = S^nG$

Public
Structured
Reference
String (SRS)

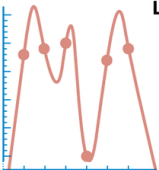
specific to each commitment

Data

Plaintext Data
STUART

UTF-8 Encoding:
83, 84, 85, 65, 84

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

$$[f(S)] = [a_0S^0 + a_1S^1 + a_2S^2 + a_3S^3 + a_4S^4 + a_5S^5]$$
$$[f(S)] = a_0[S^0] + a_1[S^1] + a_2[S^2] + a_3[S^3] + a_4[S^4] + a_5[S^5]$$

single value generated during polynomial creation — a_i — single value generated during trusted setup

(2/23) Before we begin, a moment for all the mathematicians that labored vigorously to develop the world we have today...

Now let's proceed to simplify their work far beyond what they'd be comfortable with!



(3/23) A cryptographic commitment scheme is a structured process that allows one party to bind themselves to a particular value (keeping it hidden).

A good commitment scheme protects the underlying data for as long as desired, but can be revealed at some future date.

(4/23) Example: Alice bets Bob he can't guess the number she is thinking of. But Alice can change her number after Bob makes his guess, and so Bob isn't likely to take the bet.

Instead, Alice writes down the number, committing to it. After Bob's guess, she can reveal it.

(5/23) There are many ways to commit to a specific piece of data, the key is to commit in a way that only obscures the data (instead of destroying it).

The most basic example of a commitment scheme is one based on a hash function.

(6/23) Think of a hash function like a black box that takes some input data and irrevocably transforms it into a unique string of nonsense.

Imagine if Alice picks her number, hashes it and provides it to Bob.

**Haym**
@SalomonCrypto · [Follow](#)



(1/7) Computer Science 101: Hash Functions

What is a hash function? What are the characteristics of a good hash function? Where do hash functions appear and why do I hear about them all the time?

If you want to understand the fundamental tool of crypto, this guide is for you!

Hash Functions

A hash function transforms any amount of data into a compact value of uniform length.

INPUT	OUTPUT
Hello World	0x829cd824b016326a401d083b
Hello Wold	0xabd6bd33983cb06776e89273
Social Security Number: ***-**-****	0xad22b653d2d85490c0147dfa1
	0x91bfa44d98f1d3e2v2d098d5ff
	0x299c0ce9763c53debb12a87e1

A good hash function is quick and efficient to compute, but difficult (if not impossible) run in reverse, and distribute values uniformly (randomly) across all possible outputs.

3:54 PM · Sep 7, 2022

[Read the full conversation on Twitter](#)

 117  Reply  Copy link

[Read 1 reply](#)

(7/23) By receiving the hash Bob won't have any new info to help him guess. Once he gets Alice's answer (after guessing) he can hash it and check it against the original hash Alice provided.

The hash commits Alice to the underlying number, but does not leak any information.

(8/23) The problem with hashing is that it is a very blunt and destructive tool. Hashing creates a unique, non-reversible fingerprint - the same property that makes it a good commitment candidate also limits how useful it can be.

(9/23) For example, let's say Carol has a list of 500 numbers and she wants to see if Dennis knows even one of them.

If Carol hashes the whole list, all 500 values get collapsed into a single hash. You can't tell if just one value is in that hash; it's all or nothing.


(10/23) This is the problem space:

How can we commit to a chunk of data in a way that allows us to check specific data points without revealing any extra data.

If I ask "if $x = 1$, does $y = 4$?" We want to answer YES or NO; we do not want to leak any other information.

(11/23) Fortunately, we already know our solution: polynomial commitments!

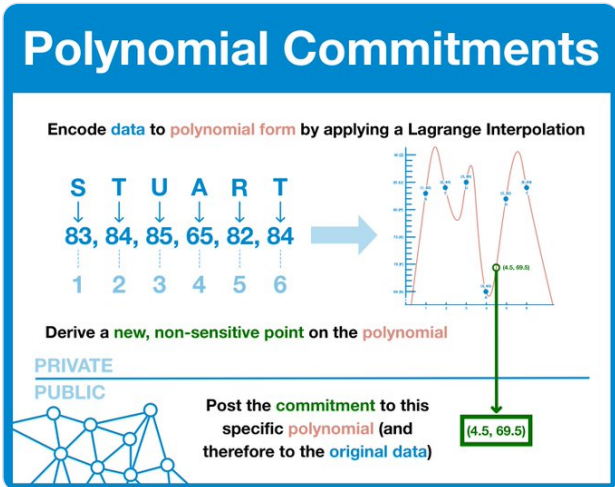
Quick refresher: any data can be represented in polynomial form through a well-understood process called a Lagrange interpolation.

**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

S T U A R T
↓ ↓ ↓ ↓ ↓ ↓
83, 84, 85, 65, 82, 84
1 2 3 4 5 6


Derive a new, non-sensitive point on the polynomial




PRIVATE
PUBLIC

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](#)

 368  Reply  Copy link

[Read 13 replies](#)

(12/23) Data is discrete - it has a finite size with specific values. To form the polynomial, we break the data into chunks, taking each chunk in order (x) and plotting our data (y).

Polynomial form allows us to derive y-values for ANY x-value, even if it doesn't really exist.

(13/23) If our data is the word "STUART," we plot the following points:

[1, S]

[2, T]


[3, U]

[4, A]

[5, R]

[6, T]

Polynomial form allows us to calculate a value for $x = 4.5$, even though the result is a non-real, nonsense answer.

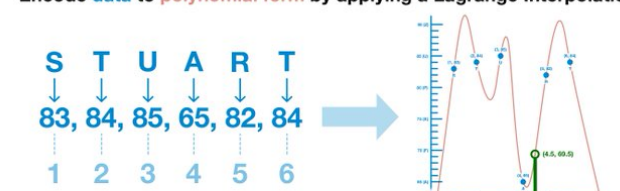
 **Haym**
@SalomonCrypto · Follow

Replying to @SalomonCrypto

(14/17) We call this non-sensitive point a "polynomial commitment" as its mere existence serves as cryptographic-proof that the creator is committed to working with the same polynomial.

A polynomial commitment acts a unique fingerprint for a specific set of data.

Encode **data** to **polynomial form** by applying a Lagrange Interpolation



Derive a new, non-sensitive point on the polynomial

PRIVATE
PUBLIC

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

(4.5, 69.5)

1:50 AM · Oct 16, 2022

9 ❤️ Reply ↻ Copy link

Read 1 reply

(14/23) This is going to form the basis of our commitment scheme; we are going to commit to data by:

- 1) converting it to polynomial form
- 2) evaluating the polynomial at some non-sensitive point
- 3) sharing the evaluated value (referred to as the commitment)

(15/23) Here's the important question: which non-sensitive point are we going to use for our commitment?

If the inputs (x) we use to generate our commitment are publicly known, we run into a lot of issues. Most importantly, attackers can easily break and/or overcome the system.


(16/23) A random number would be perfect; one that neither party knows beforehand (and therefore can't craft attack, producing correct values from the wrong underlying data).


But this is the internet, how are we going to get a random number like that?

(17/23) Once again, we've got our solution: an elliptic curve trusted setup!

By the end of the setup, we will have irrevocably hidden a secret number (lost forever) in an elliptic curve.

And though the number is forever hidden, it's hidden in a particularly useful way.

**Haym**
@SalomonCrypto · [Follow](#)

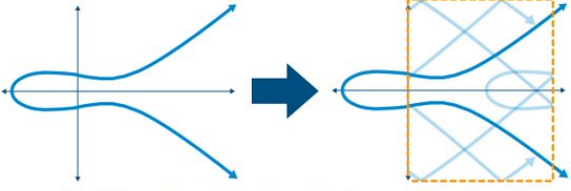


(1/19) Elliptic Curve Cryptography: Trusted Setups

Successful cryptography is cryptography that transforms legible data into digital-static. Before we go big, let's wrap our mind around something simple.

An instruction manual for creating a permanently secret number.

Begin with your elliptic curve function $y^2 = x^3 + ax + b$,



and bind it to a **min and max bounds** using **modular arithmetic**.

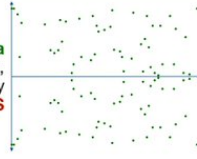
Secret Number **S** **Begin with $S^0 = 1$**
[S] **Set $x = S^0$ and evaluate $y^2 = x^3 + ax + b$**
 Let the solution be denoted as $[S^0]$
 Set $S^1 = S^0 * S$ and repeat n times (until S^n)


[] denotes secret; after setup no one will ever know any value inside **[]**


$f(S^0) = [S^0]$
 $f(S^1) = [S^1]$
 $f(S^2) = [S^2]$
 $f(S^3) = [S^3]$
 \vdots
 $f(S^n) = [S^n]$




**Public
Structured
Reference
String (SRS)**

The **SRS data**
appears random,
but is actually
related by **S**



4:27 AM · Oct 17, 2022 

 [Read the full conversation on Twitter](#)

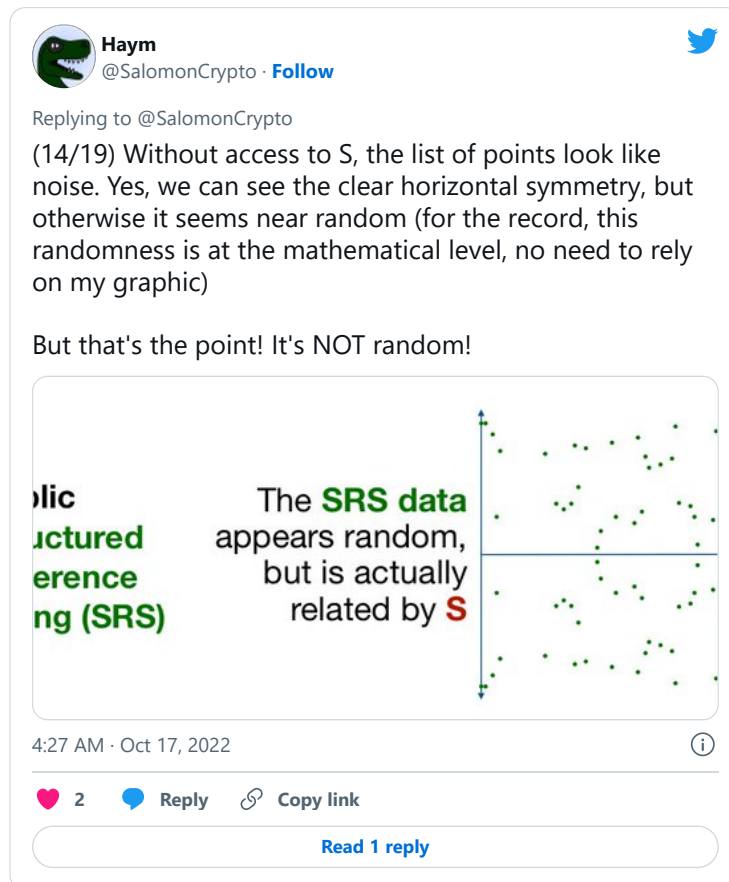
 244  Reply  Copy link

[Read 6 replies](#)

(18/23) Our setup provides us a list of numbers where each number is:

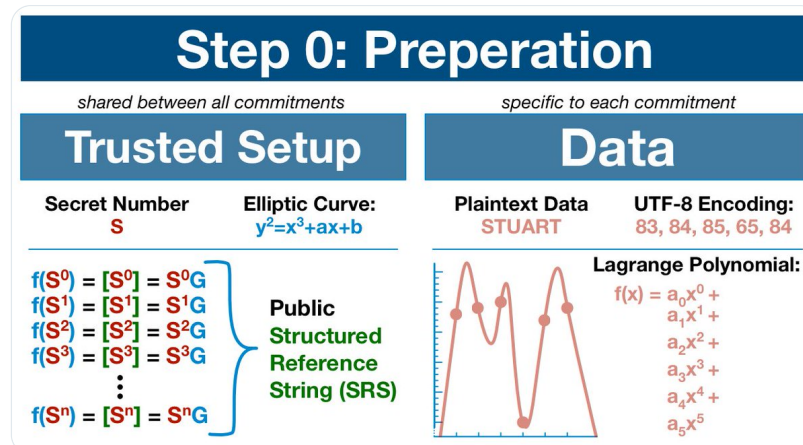
- 1) the secret number S
- 2) multiplied by itself i times
- 3) run through our elliptic curve function

S^i is the fundamental building block of the SRS... the exact kind of term we need for our polynomial.



(19/23) Before we get to the big moment, let's take one final look at the pieces.

Notice how the terms created by the trusted setup (specifically the ascending exponent) resemble requirements of the data's polynomial (again, the ascending exponent).



(20/23) Now we are ready, we can commit to our data. We simply check how many terms our polynomial has, grab an equivalent number of points from the SRS and process the polynomial.

The result: you've been able to compute the value of $f(S)$, even though S is permanently a secret!

Step 1: Commit

Commitment: $[f(S)]$

single value that serves as the polynomial commitment

$[f(S)] = [a_0S^0 + a_1S^1 + a_2S^2 + a_3S^3 + a_4S^4 + a_5S^5]$
 $[f(S)] = a_0[S^0] + a_1[S^1] + a_2[S^2] + a_3[S^3] + a_4[S^4] + a_5[S^5]$

single value generated during polynomial creation

a_i
 $[S^i]$

single value generated during trusted setup

(21/23) In fact, our commitment is actually just another point on our elliptic curve!

I've written out the (aggressively simplified) math below, but you don't need to worry about it.

The important takeaway is simple: a KZG commitment is a point on the underlying elliptic curve.

Commitment: $[f(S)]$

$[f(S)] = [a_0S^0 + a_1S^1 + a_2S^2 + a_3S^3 + a_4S^4 + a_5S^5]$
 $[f(S)] = a_0[S^0] + a_1[S^1] + a_2[S^2] + a_3[S^3] + a_4[S^4] + a_5[S^5]$
 $[f(S)] = a_0S^0G + a_1S^1G + a_2S^2G + a_3S^3G + a_4S^4G + a_5S^5G$
 $[f(S)] = (a_0S^0 + a_1S^1 + a_2S^2 + a_3S^3 + a_4S^4 + a_5S^5)G$

Commitment


$[f(S)] = f(S) \cdot G$

Point on elliptic curve

Commitment: $[f(S)]$

(22/23) Remember, our cryptographic scheme is built around the elliptic curve discrete logarithm problem: in an elliptic curve system, it is EXTRAORDINARY difficult to find X with only X^i .


And so our commitment can be publicly shared - any information stays secure.

**Haym**
@SalomonCrypto · Follow

Replying to @SalomonCrypto

(22/25) We have reached the impenetrable foundation of elliptic curve cartography: the elliptic curve discrete logarithm problem.

The discrete logarithm problem (factoring modular numbers) is provably difficult, the extra layer of ambiguity makes it SIGNIFICANTLY more difficult.




**Haym** @SalomonCrypto
Replying to @SalomonCrypto
(24/25) Arjen Lenstra framed it best in Universal Security

Tl;dr breaking a 228-bit RSA key requires less energy to than it takes to boil a teaspoon of water. Breaking a 228-bit elliptic curve key requires enough energy to boil all the water on earth.

eprint.iacr.org/2013/635.pdf

RSA key – size (in bits)	ECC key – size (in bits)
512	106
768	132
1024	160
2048	210
21000	600

5:17 PM · Oct 16, 2022

 3  Reply  Copy link


Read 1 reply

(23/23) And there we have our KZG commitment! Using the magic of elliptical curve cryptography, we are able to evaluate a function using a secret number. One that no adversary could know before hand, because NO ONE knows it before hand.

Now, it's time to put our point 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



(1/23) 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

Secret Number
 S

Elliptic Curve:
 $y^2 = x^3 + ax + b$

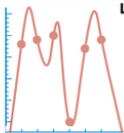
Data

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: $[f(S)]$

single value that serves as the polynomial commitment


$[f(S)] = [a_0S^0 + a_1S^1 + a_2S^2 + a_3S^3 + a_4S^4 + a_5S^5]$
 $[f(S)] = a_0[S^0] + a_1[S^1] + a_2[S^2] + a_3[S^3] + a_4[S^4] + a_5[S^5]$


1:06 AM · Oct 19, 2022






Read the full conversation on Twitter

 22

 Reply

 Copy link

Read 2 replies

...