MPC ZK

zkHub is built in conjunction with 2 cutting edge technologies, MPC and zk-proofs

We will break down the architecture in this article and try to explain each of these technologies and briefly explain how they work and how they come together.

Multi Party Computation

Multi-party computation (MPC) is a cryptographic tool that allows multiple parties to make calculations using their combined data, without revealing their individual input.

To understand the intuition behind MPC, let's take an example.

Say we have a group of 5 people (A,B,C,D,E) with the same job in different companies, who want to determine their average salary for future negotiations without disclosing their individual salaries for privacy reasons.

Can they do this in a way that allows each individual to find out what the average of all of their salaries is, but in such a way that no one learns any more than that?

The key to doing this is realising that all we really need is the sum of all the salaries. Once we have that, we can simply divide that by the number of participants to get the average.

This is where secret sharing comes in.

Secret-sharing

For a participant to enter their salary(S) into the computation, they do the following -

- Choose 5 random numbers x1,.., x5 such that

  S = x1+ x2 +x3 +x4 +x5

  This can be done by choosing 4 random numbers first and then choosing the last one such that the sum is correct.

  Now, privately share x1 to A, x2 to B, and so on ().

  Note that these are just random numbers to the participants, revealing nothing about the salary.

After each participant has done the same, we come to the second phase - Secure Computation.

Secure Computation

Now that everyone has received a share from each other, we need to perform the computation(sum of salaries). The table below shows what each participant has.

A	B	C	D	E
a1	a2	a3	a4	a5
b1	b2	b3	b4	b5
c1	c2	c3	c4	c5
d1	d2	d3	d4	d5
e1	e2	e3	e4	e5

Now, each participant adds up all the shares and sends them to all other participants, ie. the sum of each column in the table is computed and published.

Let's call this s1, s2,s3, s4, s5.

Now, it turns out that the sum of all the salaries is simply s1+s2+s3+s4+s5.

What we have seen is a good example of a general principle that MPC constructions often follow.

First the inputs are secret shared. And now, instead of computing on the actual data, participants do a corresponding computation on the shares they have. And from those (partial) results they obtain, we can finally reconstruct the actual output we wanted. This keeps the inputs private, because we only compute on shares that do not reveal anything about the actual data.

Multiplications can be done in a similar way.

All computations can be expressed in terms of additions and multiplications. This is what makes MPC possible.

In practice, these additions and multiplications are done in **Prime Fields**

Prime fields are mathematical structures that consist of a set of numbers where the number of elements is a prime number.

For example, consider the prime number 5. The prime field with 5 elements, would include the numbers 0, 1, 2, 3, and 4. So, in the prime field of 5,

4 + 4 = 8 (mod 5) = 3

All computations in MPC are done in prime fields of extremely large prime numbers, since they are predictable and easier to operate in.

In real life MPC, more advanced techniques are used where

- we can do general computation, not just sums.
- we can complete the job, even if some participants crash or malfunction.
- we can at least identify those that did not do what they were supposed to.

Let's see how it is different in real life.

Shamir Secret Sharing

We just saw that secret sharing is an integral part of MPC. The idea is to share a secret s between n participants (s1, s2, s3, …. sn). This is a very robust way to store *s* because it gives you some assurance that *s* will remain secret, but also that *s* will not be lost.

This is because the shares are constructed with respect to a threshold value *t*, a number between 1 and n. The idea is that given any *t* shares or less, you will learn nothing at all about s, but given at least t+1 shares, you can easily compute s.

Secret sharing in practice is a bit more complex. It's called Shamir Secret Sharing

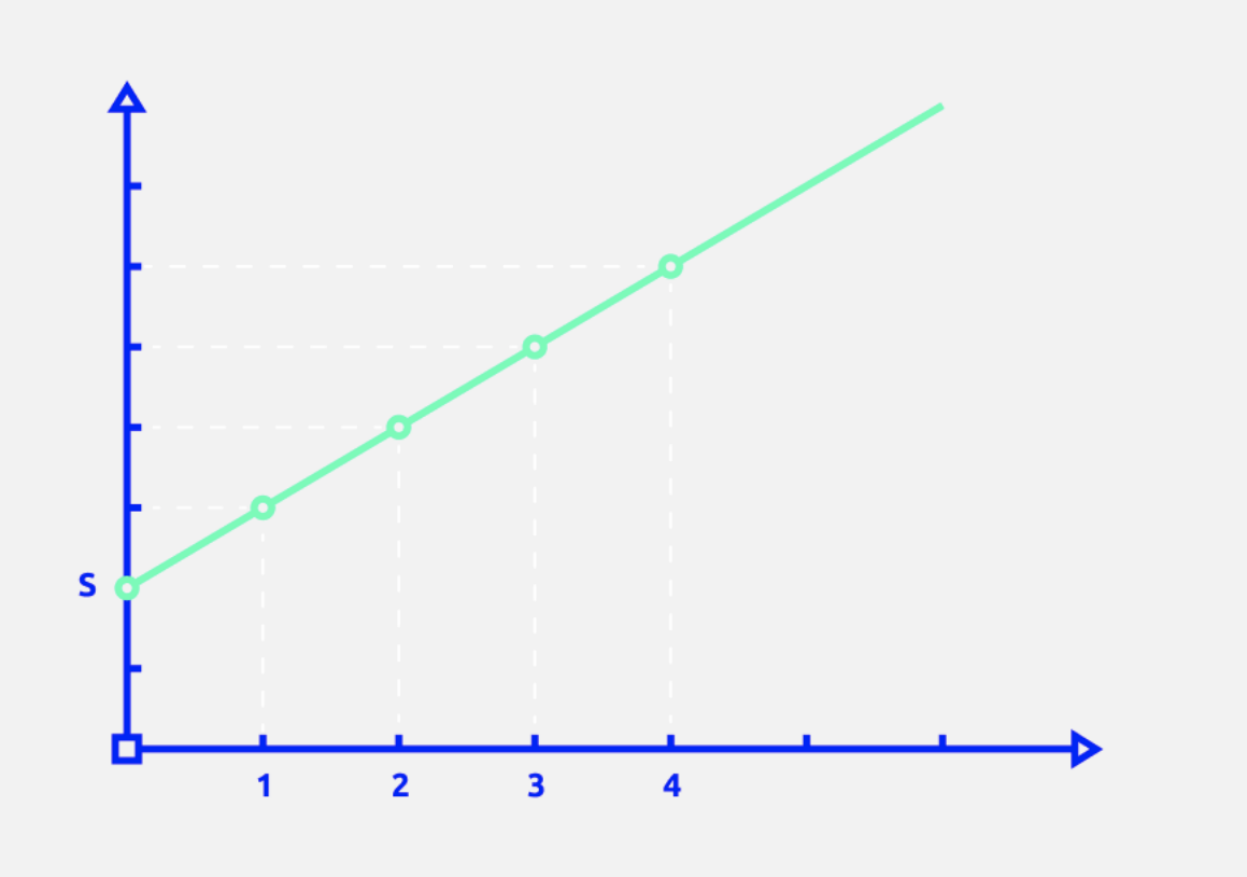
It is an elegant way to do secret sharing for any n and t.

Let's understand how it works with an example -

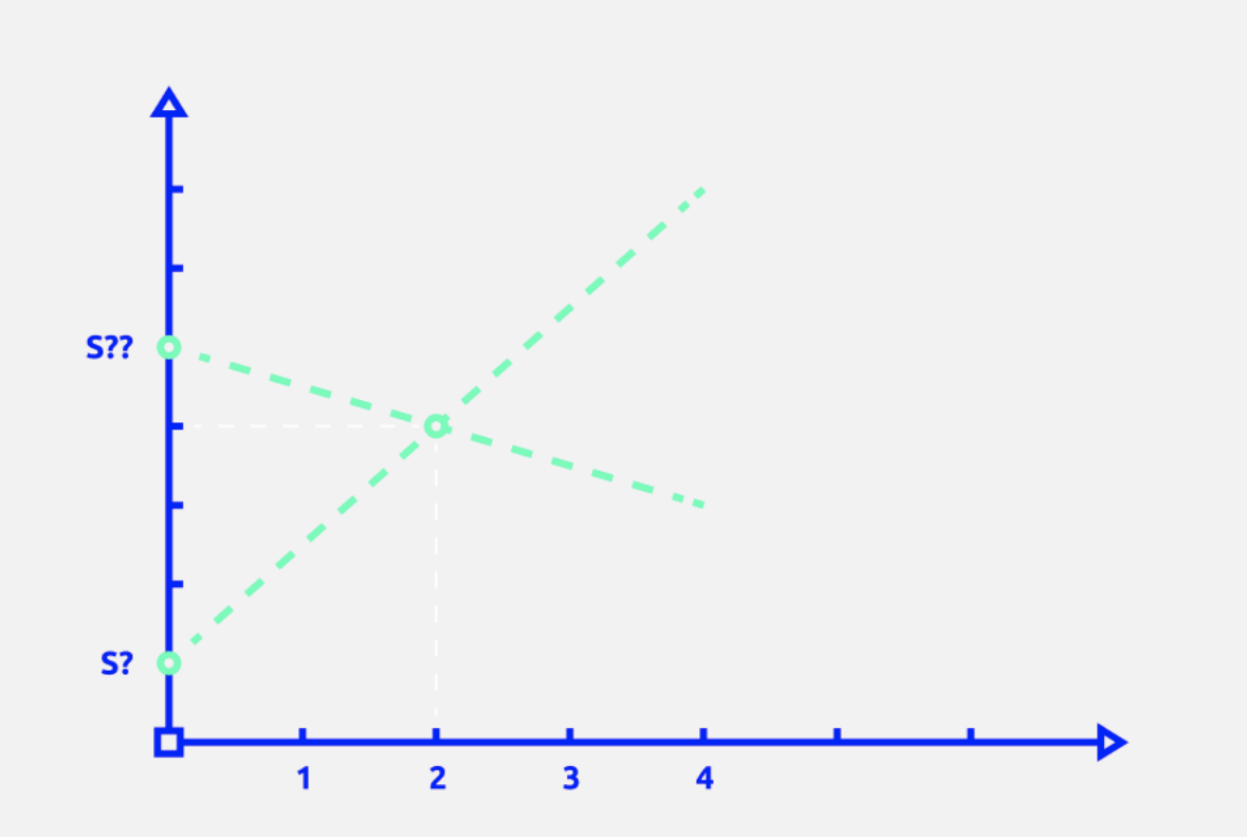
Let n = 4 & t = 1 ie. 1 share would not result in information being leaked, however, any 2 would be enough to result in a leak.

On the graph, we have assigned x-coordinates 1,2,3 and 4 to the 4 participants.

When the dealer wants to share a secret *s*, they think of the secret number s as a y-coordinate and draw a random line that passes through the point (0,s). The shares are now defined as points on this line. For instance, participant number 3 gets a number s3, such that (3, s3) is on the line.



Consider now that you are given only 1 share. You have only one point and no idea which line the dealer chose, except that it passes through your point. This also means you have no idea where it intersects the y-axis. Any guess at the intersection point would define a line which is just as likely as any other line as shown below



However, if you are given 2 points, you can draw the correct line, and compute the intersection with the y-axis.

This can be done with any value of n.

For values of t greater than 1, higher degree polynomial curves can be used (eg - for a value of t = 2, the dealer would draw a random parabola intersecting y at a random point. In this case 3 shares of secret values would be needed to determine the exact value of the curve).

Once values have been secret-shared, we can perform computation on them while they are shared.

SPDZ Protocol

SPDZ Protocol is an advanced MPC Protocol. It is also used for MPC in zkHub. A few characteristics of the protocol are -

- It allows any number of parties, call it n, to perform MPC.
- It tolerates up to n-1 corrupted parties.

- It provides active security e.g. it is secure even if the corrupt parties misbehave in the protocol
- It allows the evaluation of arithmetic circuits ,boolean circuits, finite fields, arithmetic modulo power of 2s, etc.

The SPDZ protocol is divided into two phases: a preprocessing phase and an online phase.

Preprocessing phase

The preprocessing phase of SPDZ is a pre-computation phase that is performed before the online phase of the protocol. The goal of the preprocessing phase is to generate some intermediate values that can be used to speed up the computation in the online phase.

The preprocessing phase can be performed offline, which means that it can be done before the parties need to communicate with each other. This makes the preprocessing phase a valuable, as it allows the parties to perform some of the computation offline, which can improve the efficiency of the protocol.

Online phase

The online phase of SPDZ is the phase of the protocol where the parties actually perform the computation. The goal of the online phase is to compute the function on the input values, without revealing any information about the input values or the output value to any of the parties.

How to speed up MPC in zk-proofs

Using MPC causes a 1000x slowdown in computations. However it is an important technique adopted by the industry to perform private computations.

Generating a zk-proof also causes a similar slowdown when compared to just sharing values without proofs.

Using MPC with zk-proofs naively would cause a whopping 1000 x 1000 times slowdown in performing computations which is unacceptable.

zkHub cleverly uses both of these together, reducing the overhead from 1,000,000 times to 2000 times.

To understand how, let's look at a small part of how zk-proofs are evaluated.

To create a zk-proof of a computation, we represent it as a “circuit”. A circuit is a combination of inputs, gates and outputs, where the gates represent addition or multiplication.

The circuit is evaluated over an **elliptic curve**.

An elliptic curve is a mathematical curve which has a few special characteristics -

1. Group structure: The points on an elliptic curve form a mathematical group. This means that there are well-defined operations for adding and subtracting points on the curve.
2. Symmetry: Elliptic curves possess a symmetry property. If a point (x, y) is on the curve, then its reflection across the x-axis (x, -y) is also on the curve. This property is used in various operations involving elliptic curves.

To evaluate a circuit over an elliptic curve, we first need to convert the circuit into a form that can be evaluated over an elliptic curve. This is done by replacing the basic operations in the circuit with elliptic curve operations. Once the circuit has been converted, we can then evaluate it by following the edges of the circuit and performing the corresponding elliptic curve operations.

Elliptic curve operations are extremely efficient. This is because elliptic curves can be used to represent points on a curve, and operations of points on a curve can be performed using a simpler algorithm than traditional integer operations.

By making the SPDZ protocol run over the circuit's elliptic curve, these efficiencies are made possible.