1

TÉCNICO LISBOA

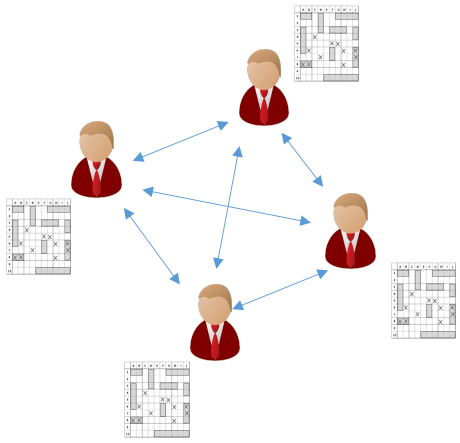
Multiparty Battleship Game

Carlos.Ribeiro@tecnico.ulisboa.pt

2

The Game

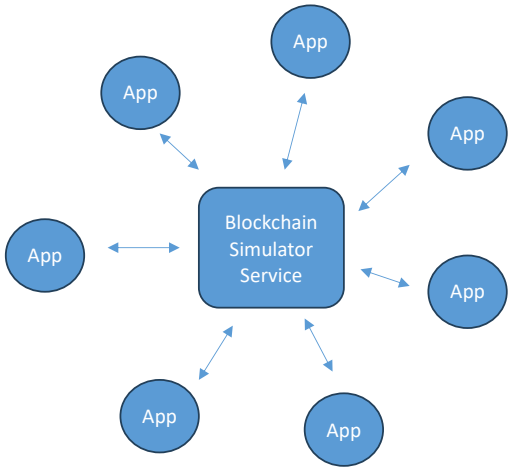
- One player creates the game by telling others the position of her fleet.
- The other players enroll in the game by telling everyone that they have positioned their fleet.
- The player that created the game starts the game by shooting at the fleet of some other player.
- The player receiving the shot reports to have received a shot into a vessel or that the shot missed all of its vessels.
- The player receiving the shot also receives the turn to fire.
- The player with a sunk fleet cannot fire. It must wave her turn.
- A player claims victory by proving that its fleet is not sunk and no one else can do the same.



3

The architecture

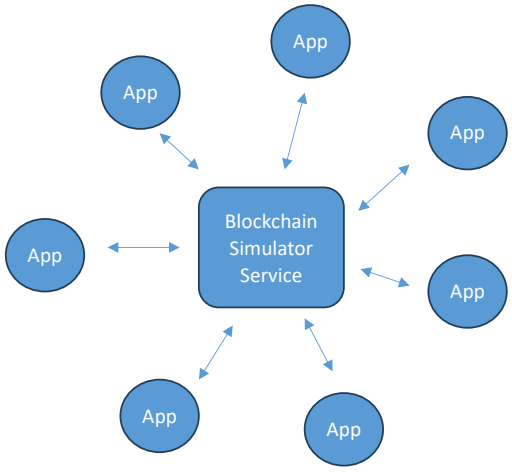
- There is a central service with no private information.
- The central service simulates a blockchain service run by each player's node.
- Each player runs a copy of a specific App
- The IP:UDP port of the service is known by everyone
- The App registers for a game by sending a message with that game's name.
- The initial App to register a name creates the game.
- The server lists the player's IP:UDP Port for each game.
- Each player shoots at someone else fleet in turn.
- A player receives the turn when receives a fire.



4

The Proofs

- At registration, a player must prove that she positioned the fleet correctly.
- When receiving a shot a player must prove that her answer is correct.
- When firing a shot a player must prove that not all of her fleet is sunk.



5

Proving correct positioning of fleet

- Let's define a function
  - $f(nonce, fleet) := \langle check(fleet), Hash(nonce, fleet)\rangle$
  - *fleet* is a vector with the position of the fleet.
  - *check()* is a function that verifies the boundaries of the fleet.
  - *nonce* is a random number that is never disclosed to anyone.
  - Hash() is a collision-resistant hash function.
- We want to prove that we know a $nonce$ and a $fleet$ for which $f(nonce, fleet) = \langle True, h\rangle$ without ever revealing the $nonce$ or $fleet$.
- At the registration phase, h does not prove anything; we are just proving that h is an hash for some fleet that checks true.

6

Proving correct report on a shot

- Let's define a function
  - $g(X, Y, nonce, fleet) \coloneqq \langle Report(X, Y, fleet), Hash(nonce, fleet) \rangle$
  - *fleet* is a vector with the position of the fleet.
  - *Report()* is a function that computes hit-or-miss.
  - *nonce* is the same random number used before
  - Hash() is a collision-resistant hash function
- We want to prove that we know a $nonce$ and a $fleet$ for which $g(X, Y, nonce, fleet) = \langle r, h \rangle$ without ever revealing the $nonce$ or $fleet$
- In this case h must be the same h reported in the registration and r the reported result.

7

Proving the fleet is not sunk

- Must extend the other two proofs.
- Besides the fleet, each player must keep a state with the shots received at the fleet.
- The *Check()* function must take the initial status and check that it is empty: not shots on vessels.
- The report check is more complex. It must provide two hashes instead of one, and it should receive two statuses, one after the shot and another before the shot.
  - The first hash is calculated with the first status and the second with the second status.
  - The *Report()* function should check that the second status equals the first status plus the new shot.
- When the report is received, the blockchain simulator also receives that player's new commitment (from the new status).
- Proving the fleet is not sunk is just a check on the status to check that not every boat is sunk.

8

TÉCNICO LISBOA

zkSNARK

9

What is a zkSNARK

- This chapter is a summary of the excellent video by Dan Boneh
  - https://www.youtube.com/watch?v=gcKCW7CNu_M
- zkSNARKs are proof systems for function executions
- A verifier can check the output of a function
  - without running the function,
  - without knowing all of its input parameters

10

zkSNARK

zk Zero-Knowledge

S Succinct

N Non-Interactive

AR Argument of

K Knowledge

Blockchain: zkRollups
Miners validate transactions, place them in a block, and mine that block (i.e. find r $s.t.\ H(Block\|r) = a\|0000$)
Every other wallet manager must verify each transaction in the block and that the r found by the miner is correct.
This is too much for the average wallet, therefore miners actually produce short proofs that every transaction in the block and the value r is correct so that every wallet user is able to verify it easily.

Blockchain: zkBridge
Generates a proof of consensus in one chain to be sent to another chain, so that a bridge between the 2 may occur.

Outsource Computation
The cloud generates proof of correct computation.

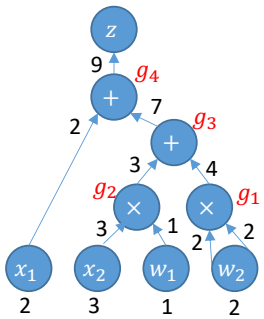
Blockchain: zCash
Have private transactions that may be verified (balance pre- and post-transaction) without being read. Each one is generated with a zkProof.

Fake News Images
Some cameras already sign (with a cryptographic key) the images with their metadata. The problem is that when they are pre-processed to e.g. reduce the pixel size and be published, the signature gets corrupted. But zkSNARK may be generated stating that the only transformation applied to the photo was resolution reduction.

11

Provable functions

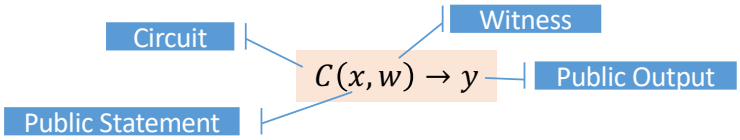
$C(\vec{x}, \vec{w}) = x_1 + w_1 x_2 + w_2^2$



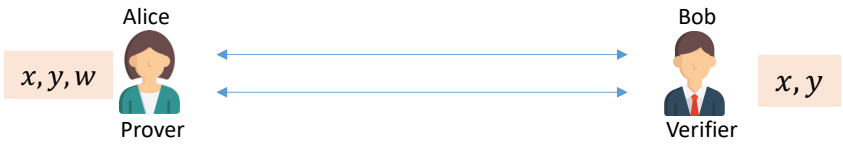
- Functions are arithmetic circuits
  - $C: \mathbb{F}_p^n \rightarrow \mathbb{F}_p$
  - $\mathbb{F}_p$={0,…,p-1} is a finite field for some prime p
- Can be represented by DAGs
  - Internal nodes are operations
  - Leafs are inputs
  - Root is the output
- Functions are usually NP problems
  - $C_{SHA256}(h, m)$:= h==SHA256(m)
  - $C_{Sig}(pk, m, \sigma) \coloneqq Verify(pk, m, \sigma)$

12

Proof (Argument) Systems



Alices convinces Bob that $\exists w : C(x, w) = y$



Interactive vs non-interactive systems

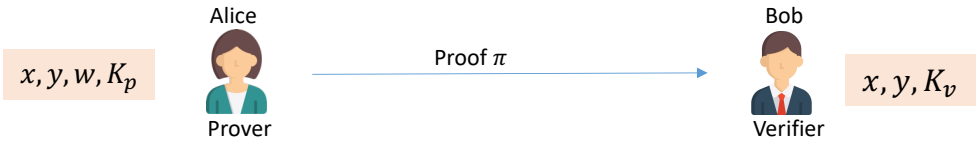
13

Pre-processing non-interactive Argument Systems

Three Algorithms S, P, V

Someone runs a pre-processing algorithm $S(C) \rightarrow (K_p, K_v)$

Alices convinces Bob that $\exists w : C(x, w) = y$



$P(x, y, w, K_p) \rightarrow \pi$

$V(x, y, \pi, K_v) \rightarrow boolean$

14

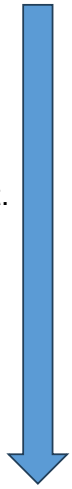
SNARK requirements

- Complete
  - $\forall x, y, w: C(x, w) = y \implies P[V(x, y, \pi, K_v) = true] = 1$
- Argument of Knowledge
  - $V(x, y, \pi, K_v) = true \implies Prover\ knows\ w\ st\ C(x, w) = y$
  - $Prover\ does\ not\ know\ w \implies P[V(x, y, \pi, K_v) = true] < negligible$
- Succinct
  - $P(x, y, w, K_p) \rightarrow short\ proof\ \pi \qquad |\pi| = O(\log(|C|), \lambda)$
  - $V(x, y, \pi, K_v) \rightarrow boolean \qquad time(V) = O(|x|, \log(|C|), \lambda)$

15

Types of Setups

- Trusted setup per circuit
  - $S(C)$ generates some random data (aka radioactive waste)
  - Whoever has this data can generate false proofs
  - Need a ceremony per circuit to ensure that no radioactive waste is kept.
- Trusted but universal setup
  - Two algorithms $S = (S_u, S_c)$: $S_u(\lambda) \rightarrow U \qquad S_c(U, C) \rightarrow (K_p, K_v)$
  - S_u also generates radioactive waste but is only run once
  - S_c is safe and is run once per circuit
- Transparent Setup
  - $S(C)$ does not generate random waste

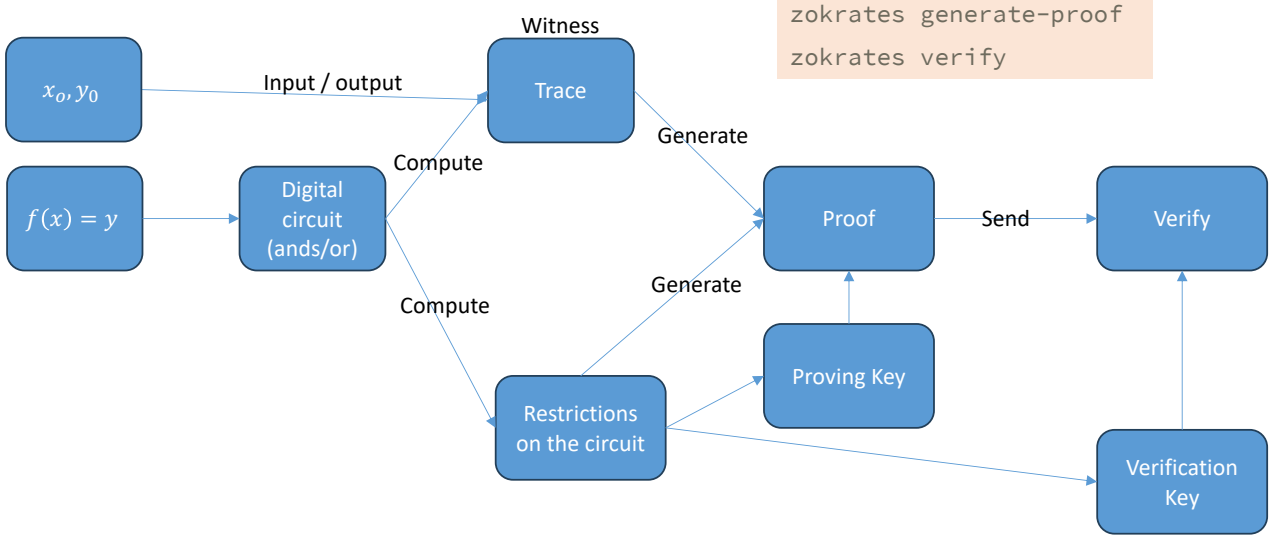


Longer and Slower Proofs

16

Prove a Circuit

```
zokrates compile test.zok
zokrates setup
zokrates compute-witness -a x y
zokrates generate-proof
zokrates verify
```



17

Proof Setup

https://play.zokrat.es

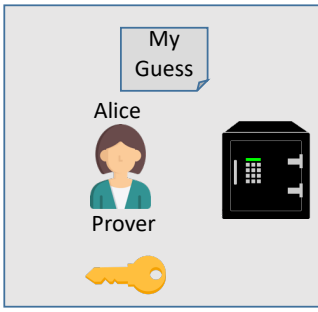
18

TÉCNICO LISBOA

B.2 Commitments

19

Commitments

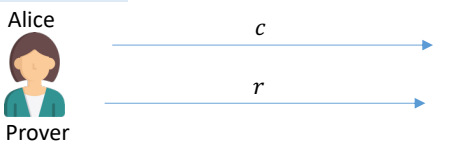


Alice wants to convince Bob that she can guess any coin flip that he makes

Bob Checks that the value of the coin flip is equal to what was written

Again Bob is convinced with probability $P = 1 - \left(\frac{1}{2}\right)^n$, where n is the number of tries.

$c = H(m\|r)$



$b \xleftarrow{R} \{0,1\}$

Accepts if $H(b\|r) = c$

Usages

Auctions and Bids
Zero-Knowledge Proofs
Voting Protocols
Zero-Knowledge Proofs
Many more

The hash function can be replaced by an Encryption function $c = E(k, m)$ and send the key k to open the commitment

20

Properties

- Hiding - The verifier should not be able to know anything about the commitment before it is open, except with negligible probability
- Binding – The committer should not be able to open the commitment to another value except with negligible probability

Perfect Hiding – Also called statistical Hiding for opposition to computationally biding
The commitment is hiding even from an exponentially powerful adversary.

Perfect Binding – Also called statistical Biding for opposition to computationally biding
The commitment is biding even from an exponentially powerful adversary.

Fact: A commitment scheme cannot be perfectly binding and hiding simultaneously

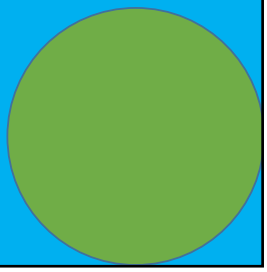
The example with the Hash and Encryption in the previous slide are perfectly hiding but only computational binding. Why?

What about the first example?

21

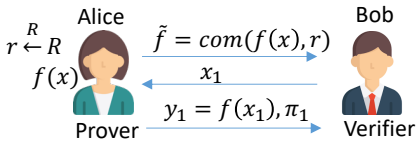
TÉCNICO LISBOA

B.4 Functional Commitments



22

Functional Commitments



π_1 - proves that $y_1 = f(x_1)$ and that $\tilde{f} = com(f(x))$

The verifier may ask the prover to open the commitment in more points

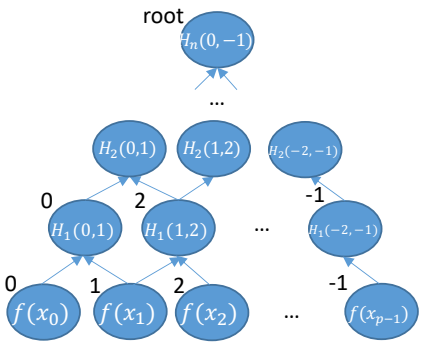
As with any other commitment, it should be:
- Binding – The verifier cannot open x_1 to another value y_1'
- Hiding [optional] – The commitment $\tilde{f}$ should not leak anything about $f(x)$

Most problems require a setup phase that sets global public parameters $pp = setup(\lambda)$
In same cases these parameters are generated from a *"Common Reference String (CRS)"* $pp = setup(\lambda, crs)$ that must be forgotten to ensure security

23

Simplified Example

Merkle Tree



$x_i \in \mathbb{F}_p$

Commit: $\tilde{f} = com(f(x)) := H_n(0,-1)$

Open $f(x_1)$: $f(x_1), \pi = (f(x_2), H_1(0,1), H_2(1,2) \dots)$

Verify: Checks that $H_n(0,-1) \stackrel{?}{=} \tilde{f}$

Size of opening: $n = \log_2 p$

Problems:
- The prover needs to generate the Merkle Tree for a very large $p = 2^{128}$
- The prover commits to a specific vector of values that it cannot change but the verifier does not know if they came from a polynomial of degree $\leq d$

24

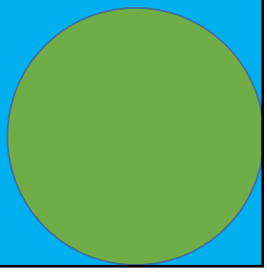
Some PCS examples
(Polynomial Commitment Schemes)

PCS	Base Technique	Quantum Safe	Setup	Proof Size
Groth'16	Linear PCP		Individual Trusted Setup	200 Bytes
KZG	Bilinear Groups		Universal Trust Setup	400 Bytes
Bulletproofs	Discrete Logarithm		Transparent	1.5 KB
FIR	Hash	Yes	transparent	100 KB

25

TÉCNICO LISBOA

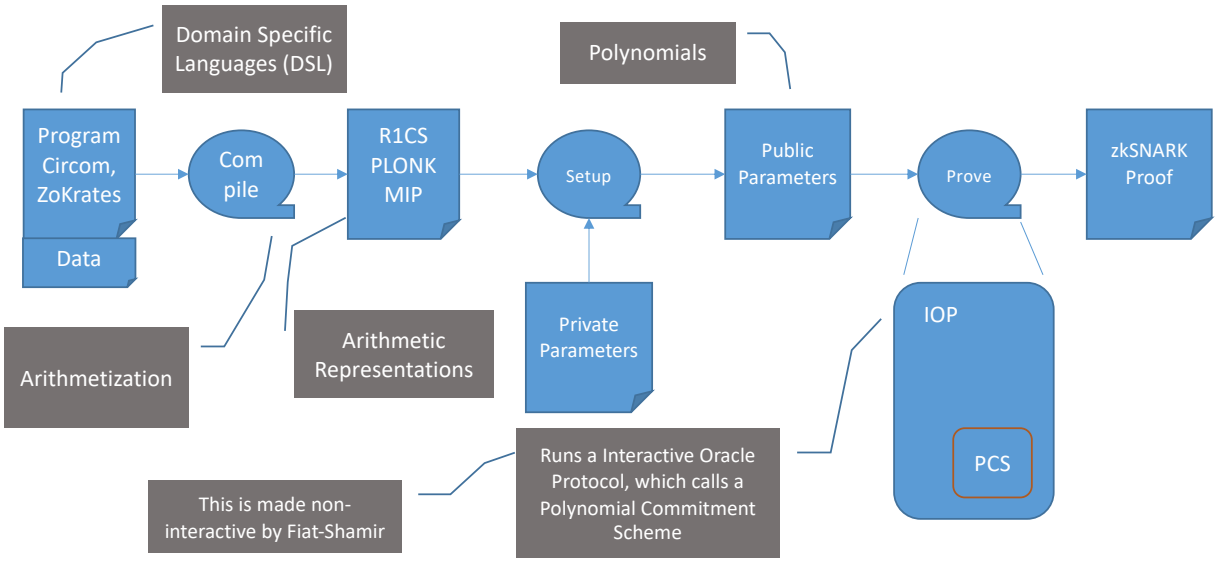
B.5 zkSNARKS



26

Overall Picture

The remaining slides will provide an intuition on the whole process



27

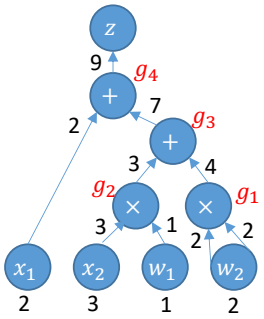
Example - Encoding

$C(\vec{x}, \vec{w}) = x_1 + w_1 x_2 + w_2^2$ — Public known Circuit

Prover wants to prove that she knows a secret $\vec{w} \in \mathbb{Z}_p^2$ s.t. $C(\vec{x}, \vec{w}) = z$ for $\vec{x} = \{2,3\} \in \mathbb{Z}_p^2$

Let $\vec{w} = \{1,2\}$ and therefore $z = 9$

1 - Mult
0 - Add



Gates

g_1	2	2	4	1
g_2	3	1	3	1
g_3	3	4	7	0
g_4	2	7	9	0

Inputs

x_1	x_2	w_1	w_2
2	3	1	2

Assume that $|\vec{t}| = n$ divides $p - 1$ then
There is a multiplicative subgroup $\Omega = \{1, \omega, \omega^2, \dots, \omega^{n-1}\}$ s.t. $\omega^n = 1 \in \mathbb{Z}_p$ and $\omega^{i<n} \neq 1 \in \mathbb{Z}_p$

ω is called a primitive nth root of unity

Primitive roots of unity generate all other roots of unity

Example:
Let $p = 97$ and $n = 16$ then there are 8 primitive roots of unity [8, 12, 18, 27, 70, 79, 85, 89]
Let $\omega = 8$ then
$\Omega = \{1, 8, 64, 27, 22, 79, 50, 12, 96, 89, 33, 70, 75, 18, 47, 85\}$

Let $f(x) \in \mathbb{F}_p^{n-1}[X]$ and interpolate at
$f(1) = t_0, f(\omega) = t_1, f(\omega^2) = t_2, f(\omega^{n-1}) = t_{n-1}$

$f(1) = 2, f(8) = 2, f(64) = 4, f(27) = 3, \dots f(85) = 2$

Fact: $\forall y \in \Omega : f(y) = t_i \Rightarrow f(\omega^j y) = t_{i+j \bmod n}$

$= \vec{t}$

Then use PCS to commit to $f(x)$ and prove some properties about the polynomial without evaluating it again or reveal w

28

Example - Properties

1. The output was correctly encoded
2. The public inputs were correctly encoded
3. All the gates were computed correctly
4. All the wires were encoded correctly

Use PCS to open $f(x)$ at $x = \omega^{11}$ if $f(\omega^{11}) = 9$ accepts

Could do the same for $x = \omega^{12}, \omega^{13}$ but it is more efficient to build a polynomial $v(x) \in \mathbb{F}_p^2[X]$ s.t. $v(\omega^{12}) = x_1$ and $v(\omega^{13}) = x_2$ and perform a ZeroTest on

$$g(y) = f(y) - v(y) = 0 \qquad \forall\, y \in \{\omega^{12}, \omega^{13}\}$$

g_1	2 (0)	2 (1)	4 (2)	1
g_2	3 (3)	1 (4)	3 (5)	1
g_3	3 (6)	4 (7)	7 (8)	0
g_4	2 (9)	7 (10)	9 (11)	0

x_1	x_2	w_1	w_2
2 (12)	3 (13)	1 (14)	2 (15)

Want to prove that the univariate polynomial $f(y) \odot f(\omega y) = f(\omega^2 y)$ for all rows of the table, where $y = \omega^{3r}$ is the the first cell of each row r, and $\odot$ may be addition or multiplication depending on the last column of the matrix.

Build a selector polynomial $S(X) \in \mathbb{F}_1^4$ using the last column of the transcript table, s.t. $S(\omega^{3r}) = 1$ if row r is a multiplication gate and $S(\omega^{3r}) = 0$ if is an addition gate, and then do a ZeroTest on

$$g(y) = S(y).f(y).f(\omega y) + \big(1 - S(y)\big)\big(f(y) + f(\omega y)\big) - f(\omega^2 y) = 0 \quad \forall y \in \{1, \omega^3, \omega^6, \omega^9\}$$

The wires being correct implies that some of the values need to be equal. The output of g_1 should be equal to the right input of g_3 therefore we must prove that $f(\omega^2) = f(\omega^7)$ and so on $f(\omega^{15}) = f(1) = f(\omega)$, … . There is an efficient method to prove this.

How to build a ZeroTest with a PCS?

29

ZeroTest

It is just one of many different tests that can be performed on polynomials

Let $f(x) \in \mathbb{F}_p^{\leq d}[X], \quad S \subset \mathbb{F}_p^n, \quad n < d$
$f(x) = 0\ \forall x \in S$

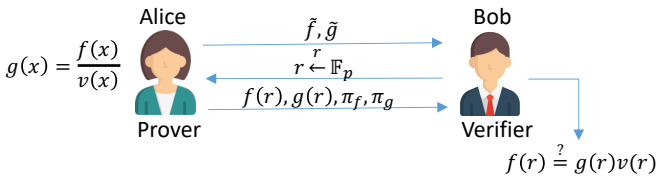
Let $S = \{s_1, .., s_n\}$
Define $v(x) \in \mathbb{F}_p^{n-1} : v(x) \coloneqq \prod_{i=1}^{n}(x - s_i)$

If $f(x) = 0\ \forall x \in S$ then $f(x) = g(x)v(x)$

Let $f(x) \in \mathbb{F}_p^{\leq d}[X]$
For $r \xleftarrow{R} \mathbb{F}_p$: $Prob[f(r) = 0] \leq \frac{d}{p}$
If $p \approx 2^{256}$ and $d \leq 40$ the d/p is negligible

So if we test a polynomial $f(x)$ in a random point r and $f(r) = 0$ the polynomial $f(x) \equiv 0\ w.h.p.$

Moreover if we test two polynomials $f(x), g(x) \in \mathbb{F}_p^{\leq d}$ in one random point r and $f(r) = g(r)$ then $f(x) \equiv g(x)$ w.h.p.



Ask the prover to calculate and send a PCS commitment of $g(x)$ together with the original commitment of $f(x)$.

The verifier will check request the prover to open $g(x)$ and $f(x)$ at a random point r (using the PCS scheme), calculate $v(r)$ and check that

$$f(r) \stackrel{?}{=} g(r)v(r)$$

30

31