Securing Elections using Homomorphic Cryptography

David Florness

February 10th, 2020

Voting

- still most done via paper ballots

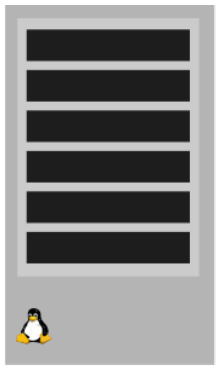
Voting

- still most done via paper ballots
- why is that?

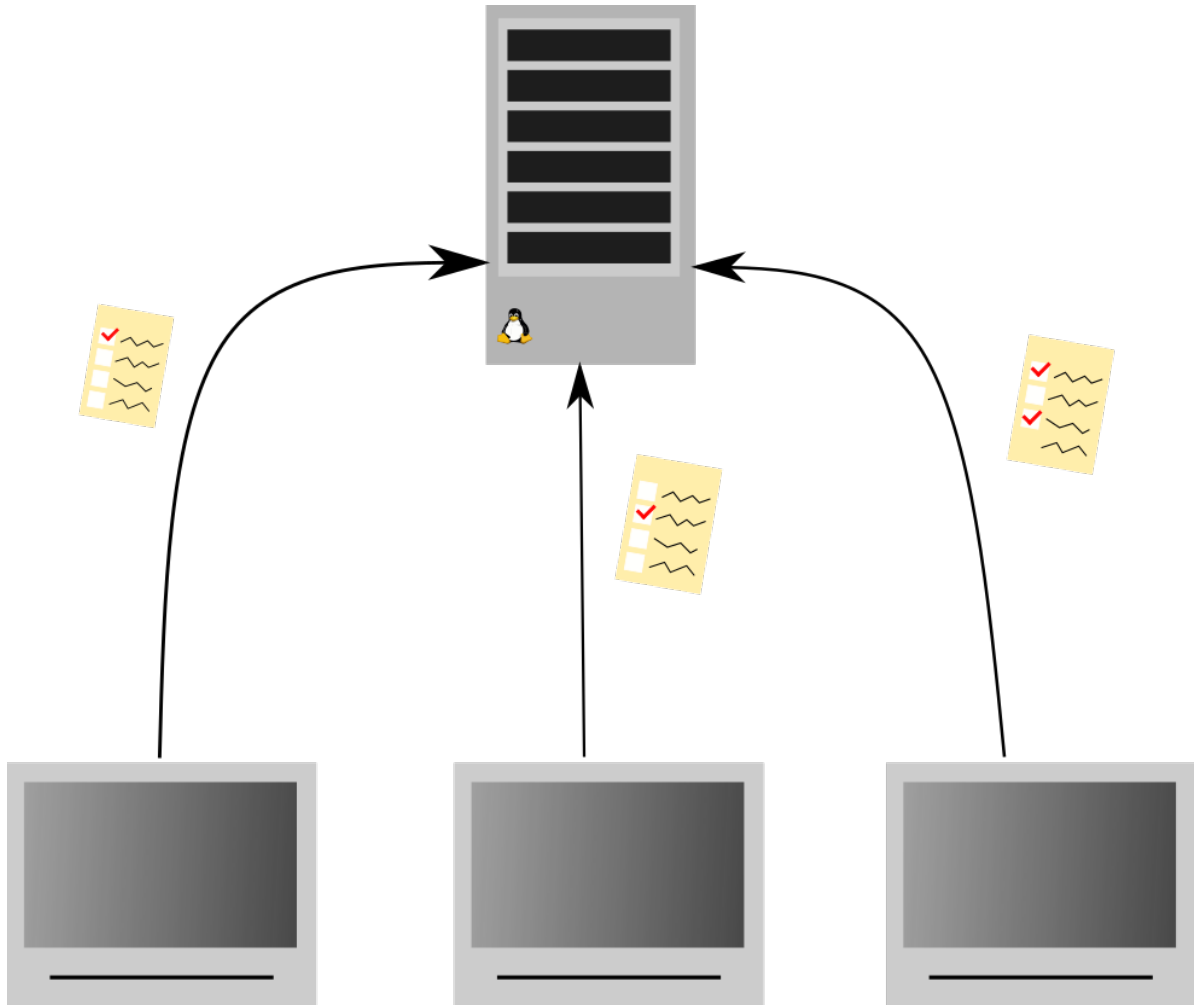
The Traditional Approach in Electronic Terms I

1. voters register with a server
2. once authenticated, voters send their ballots to the server
3. the server counts the votes
4. the aggregate result is published

The Traditional Approach in Electronic Terms II



The Traditional Approach in Electronic Terms III



- Pros

- Pros
  1. easy and simple
  2. reliable
  3. allows registration

- Pros
  1. easy and simple
  2. reliable
  3. allows registration
- Cons
  1. trust in a single authority; such a system would be rife with corruption

- We want an election system that

- We want an election system that
  - requires registration / authentication, allowing us to limit who can vote

- We want an election system that
  - requires registration / authentication, allowing us to limit who can vote
    - we may want to limit voting to members of a club, citizens of a jurisdiction, etc.

- We want an election system that
  - requires registration / authentication, allowing us to limit who can vote
    - we may want to limit voting to members of a club, citizens of a jurisdiction, etc.
    - without such authentication, stopping double-voting would be a nightmare

- We want an election system that
  - requires registration / authentication, allowing us to limit who can vote
    - we may want to limit voting to members of a club, citizens of a jurisdiction, etc.
    - without such authentication, stopping double-voting would be a nightmare
  - gives accurate aggregate results

- We want an election system that
  - requires registration / authentication, allowing us to limit who can vote
    - we may want to limit voting to members of a club, citizens of a jurisdiction, etc.
    - without such authentication, stopping double-voting would be a nightmare
  - gives accurate aggregate results
  - protects voter privacy

- We want an election system that
  - requires registration / authentication, allowing us to limit who can vote
    - we may want to limit voting to members of a club, citizens of a jurisdiction, etc.
    - without such authentication, stopping double-voting would be a nightmare
  - gives accurate aggregate results
  - protects voter privacy
    - no individual voter's vote should be knowable by anyone except said voter

- We want an election system that
  - requires registration / authentication, allowing us to limit who can vote
    - we may want to limit voting to members of a club, citizens of a jurisdiction, etc.
    - without such authentication, stopping double-voting would be a nightmare
  - gives accurate aggregate results
  - protects voter privacy
    - no individual voter's vote should be knowable by anyone except said voter

...these sound like conflicting goals...

- We want an election system that
  - requires registration / authentication, allowing us to limit who can vote
    - we may want to limit voting to members of a club, citizens of a jurisdiction, etc.
    - without such authentication, stopping double-voting would be a nightmare
  - gives accurate aggregate results
  - protects voter privacy
    - no individual voter's vote should be knowable by anyone except said voter

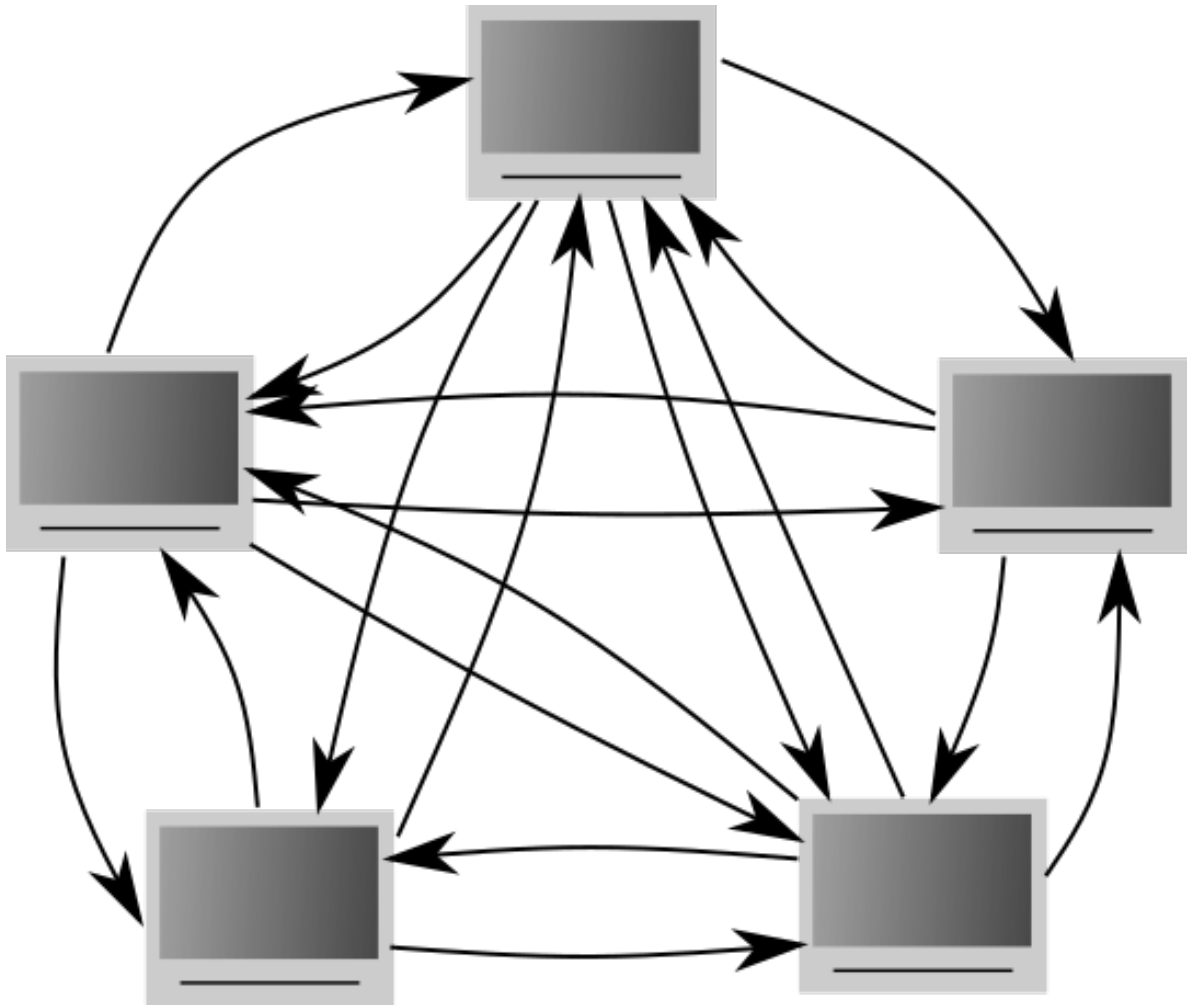
...these sound like conflicting goals...but they're not!

Disclaimer: I did not come up with any of this.

A New Approach I

- instead of a client-server approach, let's have a peer-to-peer (p2p) network where voters connect to each other directly

A New Approach II



- Authentication is essentially the same as before: voters use standard means to identify themselves (think passwords, kerberos, LDAP, SSH/GPG finderprints, multipass, etc.)

The big question

- **Question**: how and to whom do we submit ballots?

The big question

- **Question**: how and to whom do we submit ballots?
- **Answer**: we "share" pieces of our secret ballot with everyone!

Introducing secret sharing

- ***Secret sharing*** *(also called secret splitting) refers to methods for distributing a secret amongst a group of participants, each of whom is allocated a share of the secret. The secret can be reconstructed only when a sufficient number, of possibly different types, of shares are combined together; individual shares are of no use on their own.* [1]

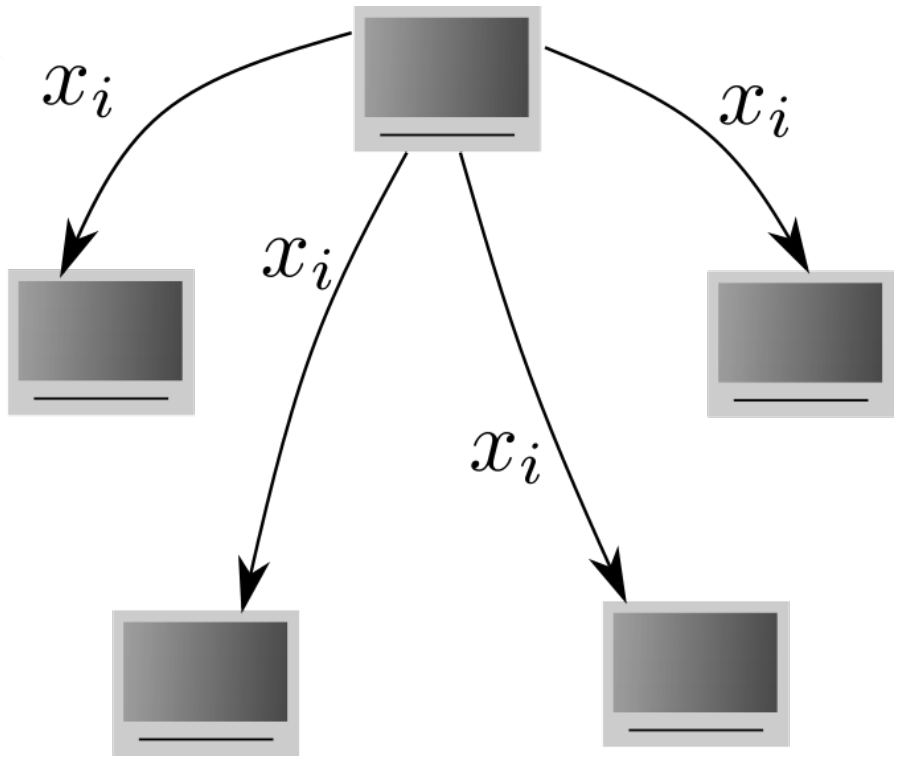
[1] `https://en.wikipedia.org/wiki/Secret_sharing`

Preliminaries

- First, let's have every voter generate and share a random natural number that will serve as said voter's public key. Let x_i denote this random number for voter i.

Preliminaries

- First, let's have every voter generate and share a random natural number that will serve as said voter's public key. Let x_i denote this random number for voter i.



Polynomials

- We're now going to leverage plain old polynomials to conduct secret sharing.

Polynomials

- We're now going to leverage plain old polynomials to conduct secret sharing.
- Let's have every voter, say voter i, create a secret polynomial P_i of degree $k-1$ where k is the total number of voters:

$$P_i(x) = c_i + a_{(i,1)}x + a_{(i,2)}x^2 + \cdots + a_{(i,k-1)}x^{k-1}$$

Polynomials

- We're now going to leverage plain old polynomials to conduct secret sharing.
- Let's have every voter, say voter i, create a secret polynomial P_i of degree $k-1$ where k is the total number of voters:

$$P_i(x) = c_i + a_{(i,1)}x + a_{(i,2)}x^2 + \cdots + a_{(i,k-1)}x^{k-1}$$

- Every $a_{(i,j)}$ is a random natural number.

Polynomials

- We're now going to leverage plain old polynomials to conduct secret sharing.
- Let's have every voter, say voter i, create a secret polynomial P_i of degree $k-1$ where k is the total number of voters:

$$P_i(x) = c_i + a_{(i,1)}x + a_{(i,2)}x^2 + \cdots + a_{(i,k-1)}x^{k-1}$$

- Every $a_{(i,j)}$ is a random natural number.
- c_i is the numerically-encoded contents of voter i's ballot. Every c_i must be encoded in such a way that summing every $\{c_j\}_{j\in[1,k]}$ results in the desired election result. For example, in a simple "yes/no" election, 1 could represent "yes" and -1 could represent "no".

Polynomials as sharable secrets

Polynomials as sharable secrets

- To know everything about a line (degree 1 polynomial), we only need two points on the it.

Polynomials as sharable secrets

- To know everything about a line (degree 1 polynomial), we only need two points on the it.
- To know everything about a parabola curve (degree 2 polynomial), we only need three points on the curve.

Polynomials as sharable secrets

- To know everything about a line (degree 1 polynomial), we only need two points on the it.
- To know everything about a parabola curve (degree 2 polynomial), we only need three points on the curve.
- ...

Polynomials as sharable secrets

- To know everything about a line (degree 1 polynomial), we only need two points on the it.
- To know everything about a parabola curve (degree 2 polynomial), we only need three points on the curve.
- ...
- As you'd expect, this pattern continues: to know everything about a k-degree polynomial, we need only $k+1$ points.

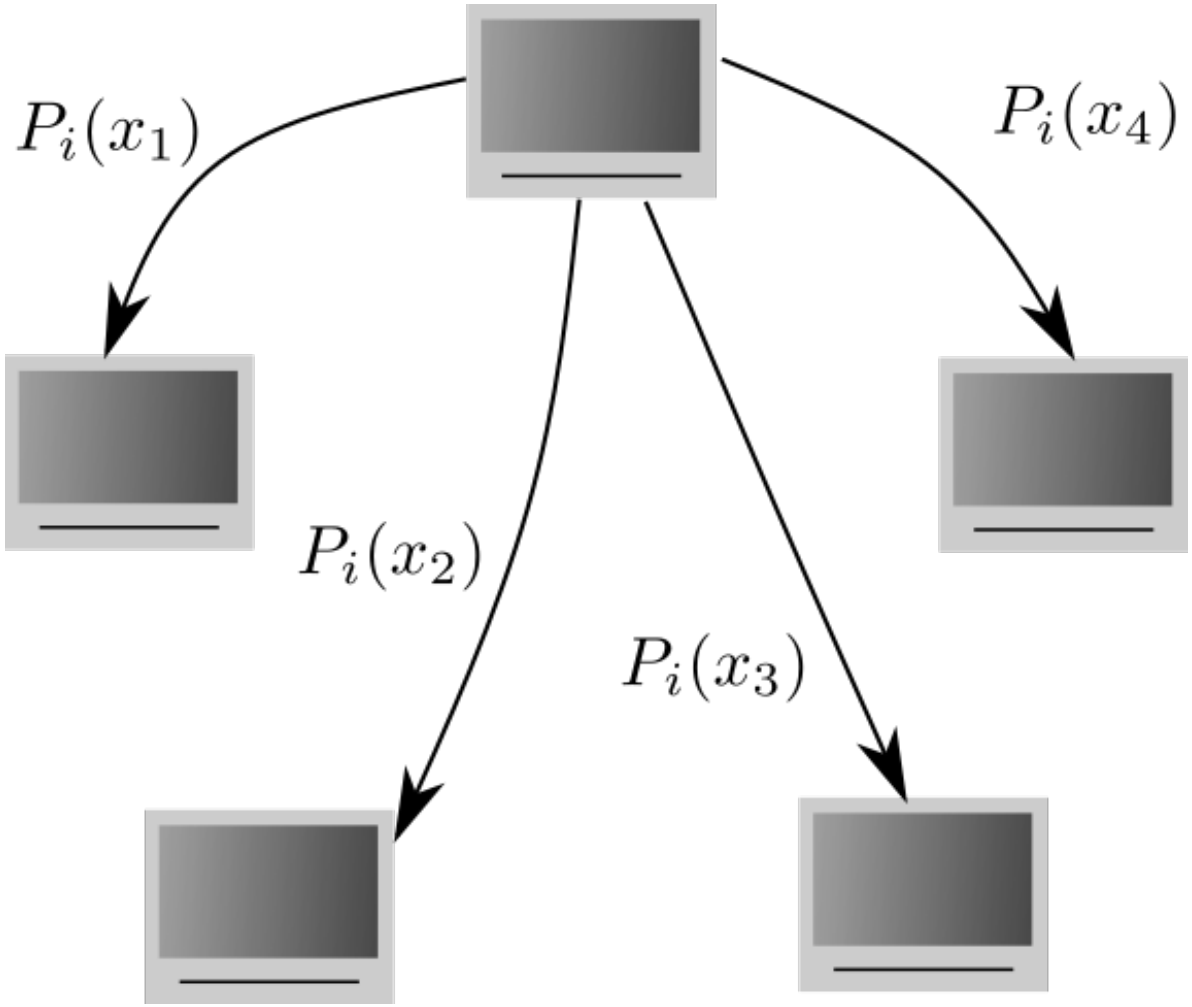
Polynomials as sharable secrets

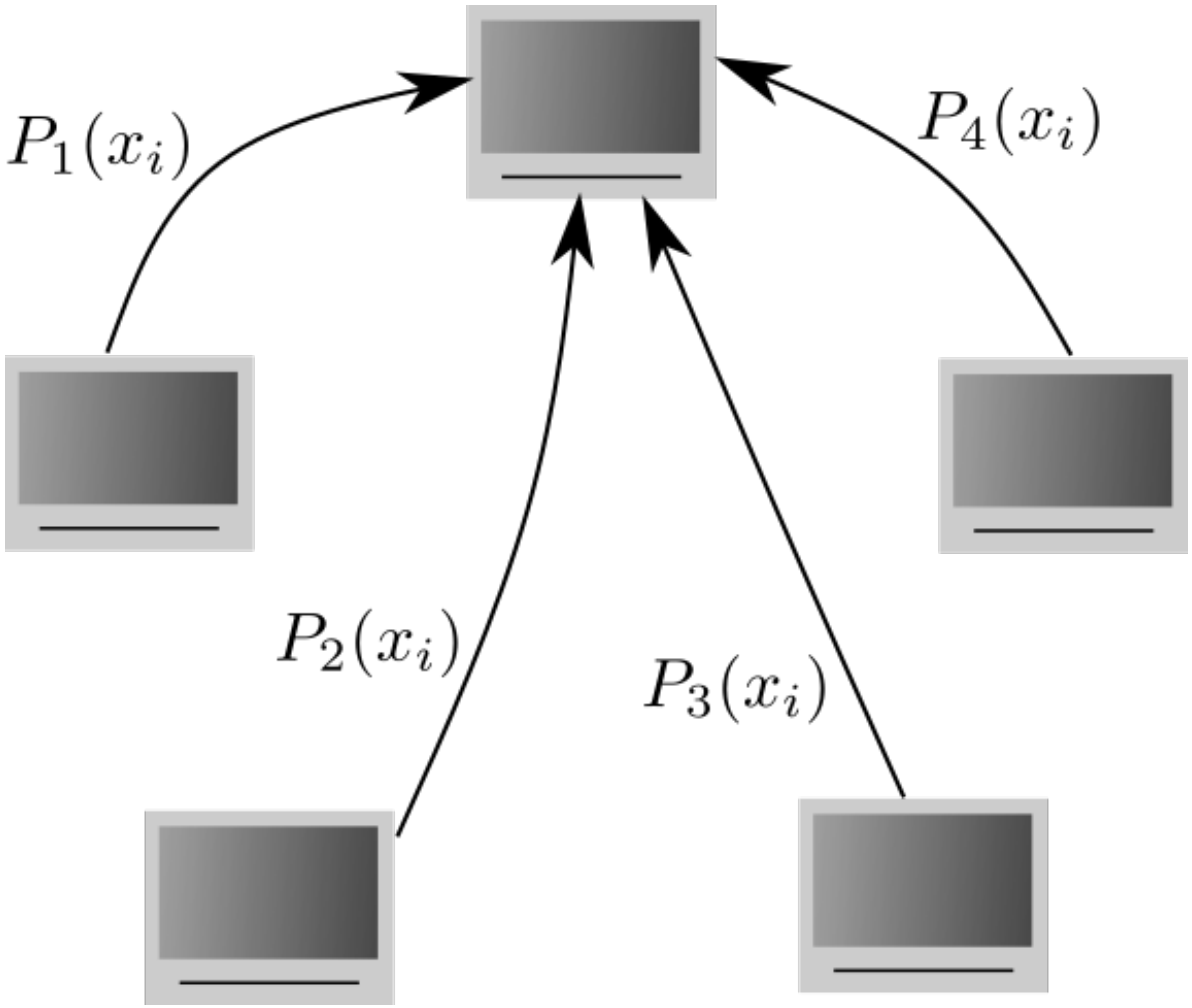
- To know everything about a line (degree 1 polynomial), we only need two points on the it.
- To know everything about a parabola curve (degree 2 polynomial), we only need three points on the curve.
- ...
- As you'd expect, this pattern continues: to know everything about a k-degree polynomial, we need only $k+1$ points.
- Thus, knowing everything about our $k-1$ degree polynomial P_i is equivalent to knowing k points on it.

Polynomials as sharable secrets

- To know everything about a line (degree 1 polynomial), we only need two points on the it.
- To know everything about a parabola curve (degree 2 polynomial), we only need three points on the curve.
- ...
- As you'd expect, this pattern continues: to know everything about a k-degree polynomial, we need only $k+1$ points.
- Thus, knowing everything about our $k-1$ degree polynomial P_i is equivalent to knowing k points on it.
- So, we're going to distribute our secret by giving everyone just one point.

- Every voter sends every other voter the result of evulating their polynomial at the given voter's public input:



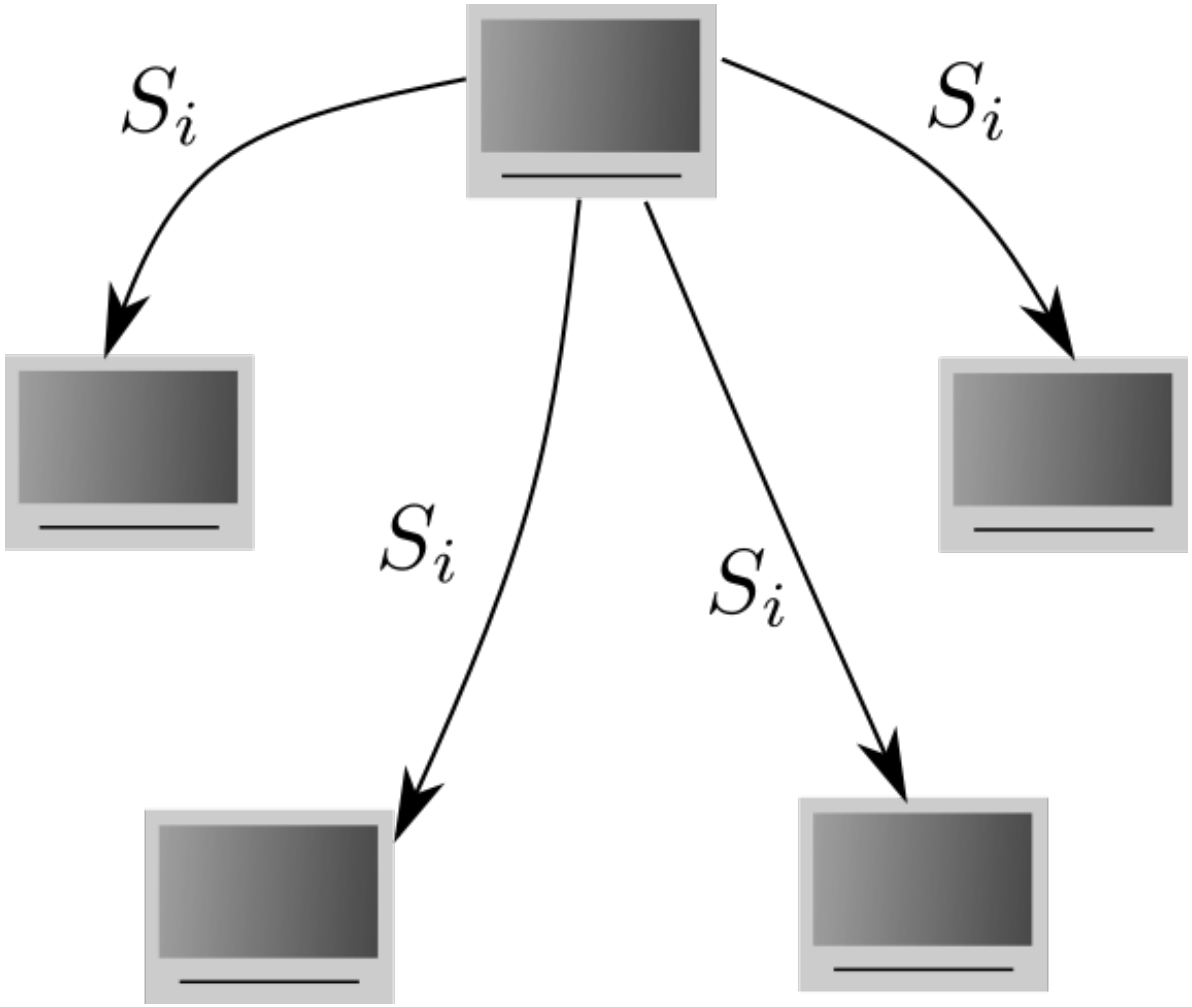

$P_1(x_i)$
$P_4(x_i)$
$P_2(x_i)$
$P_3(x_i)$

- Voter i now has:
  - $P_1(x_i)$
  - $P_2(x_i)$
  - $\vdots$
  - $P_k(x_i)$

Let's sum those values!

$$
\begin{aligned}
S_i =& P_1(x_i) + P_2(x_i) + \cdots + P_k(x_i) \\
=& \left(c_1 + a_{(1,1)}x_i + a_{(1,2)}x_i^2 + \cdots + a_{(1,k-1)}x_i^{k-1}\right) + \\
& \left(c_2 + a_{(2,1)}x_i + a_{(2,2)}x_i^2 + \cdots + a_{(2,k-1)}x_i^{k-1}\right) + \\
& \cdots \\
& \left(c_k + a_{(k,1)}x_i + a_{(k,2)}x_i^2 + \cdots + a_{(k,k-1)}x_i^{k-1}\right) \\
=& \sum_{j=1}^{k} c_j + x_i \sum_{j=1}^{k} a_{(j,1)} + x_i^2 \sum_{j=1}^{k} a_{(j,2)} + \cdots + x_i^{k-1} \sum_{j=1}^{k} a_{(j,2)}
\end{aligned}
$$

Let's have every voter share their sum with everyone else.



- Everyone now has S_1, S_2, ..., S_k:

- Everyone now has S_1, S_2, $\ldots$, S_k:

$$S_1 = \sum_{j=1}^{k} c_j + x_1 \sum_{j=1}^{k} a_{(j,1)} + x_1^2 \sum_{j=1}^{k} a_{(j,2)} + \cdots + x_1^{k-1} \sum_{j=1}^{k} a_{(j,k-1)}$$

$$S_1 = \sum_{j=1}^{k} c_j + x_2 \sum_{j=1}^{k} a_{(j,1)} + x_2^2 \sum_{j=1}^{k} a_{(j,2)} + \cdots + x_2^{k-1} \sum_{j=1}^{k} a_{(j,k-1)}$$

$$\vdots$$

$$S_k = \sum_{j=1}^{k} c_j + x_k \sum_{j=1}^{k} a_{(j,1)} + x_k^2 \sum_{j=1}^{k} a_{(j,2)} + \cdots + x_k^{k-1} \sum_{j=1}^{k} a_{(j,k-1)}$$

- Everyone now has S_1, S_2, ..., S_k:

$$S_1 = \sum_{j=1}^{k} c_j + x_1 \sum_{j=1}^{k} a_{(j,1)} + x_1^2 \sum_{j=1}^{k} a_{(j,2)} + \cdots + x_1^{k-1} \sum_{j=1}^{k} a_{(j,k-1)}$$

$$S_1 = \sum_{j=1}^{k} c_j + x_2 \sum_{j=1}^{k} a_{(j,1)} + x_2^2 \sum_{j=1}^{k} a_{(j,2)} + \cdots + x_2^{k-1} \sum_{j=1}^{k} a_{(j,k-1)}$$

$$\vdots$$

$$S_k = \sum_{j=1}^{k} c_j + x_k \sum_{j=1}^{k} a_{(j,1)} + x_k^2 \sum_{j=1}^{k} a_{(j,2)} + \cdots + x_k^{k-1} \sum_{j=1}^{k} a_{(j,k-1)}$$

- These are k points all on the same $k-1$ degree polynomial:

$$(x_1, S_1), (x_2, S_2), \ldots, (x_k, S_k)$$

- Since we know k points of the $k-1$ degree polynomial, we can find the coefficients and constant term of it with a little linear algebra:

- Since we know k points of the $k-1$ degree polynomial, we can find the coefficients and constant term of it with a little linear algebra:

$$\begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^{k-1} \\ 1 & x_2 & x_2^2 & \cdots & x_2^{k-1} \\ \vdots & & & \ddots & \\ 1 & x_k & x_k^2 & \cdots & x_k^{k-1} \end{bmatrix} \begin{bmatrix} \sum_{j=1}^{k} c_j \\ \sum_{j=1}^{k} a_{(j,1)} \\ \sum_{j=1}^{k} a_{(j,2)} \\ \vdots \\ \sum_{j=1}^{k} a_{(j,k-1)} \end{bmatrix} = \begin{bmatrix} S_1 \\ S_2 \\ \vdots \\ S_k \end{bmatrix}$$

$$\text{RREF}\left(\begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^{k-1} & S_1 \\ 1 & x_2 & x_2^2 & \cdots & x_2^{k-1} & S_2 \\ \vdots & & & \ddots & & \\ 1 & x_k & x_k^2 & \cdots & x_k^{k-1} & S_k \end{bmatrix}\right)$$

Notice that the constant term of the final polynomial is $\sum_{j=1}^{k} c_j$, which is precisely the result of the election!

In Summary

1. Every voter creates and publishes an x_i.

In Summary

1. Every voter creates and publishes an x_i.
2. Every voter creates a random polynomial of degree $k-1$ where k is the number of voters and embeds their ballot in the constant term of the polynomial.

In Summary

1. Every voter creates and publishes an x_i.
2. Every voter creates a random polynomial of degree $k-1$ where k is the number of voters and embeds their ballot in the constant term of the polynomial.
3. Every voter evaluates their polynomial with each of the inputs (x_i's) and sends each respective voter his/her result.

In Summary

1. Every voter creates and publishes an x_i.
2. Every voter creates a random polynomial of degree $k-1$ where k is the number of voters and embeds their ballot in the constant term of the polynomial.
3. Every voter evaluates their polynomial with each of the inputs (x_i's) and sends each respective voter his/her result.
4. Every voter sums the polynomial outputs they've received.

In Summary

1. Every voter creates and publishes an x_i.
2. Every voter creates a random polynomial of degree $k-1$ where k is the number of voters and embeds their ballot in the constant term of the polynomial.
3. Every voter evaluates their polynomial with each of the inputs (x_i's) and sends each respective voter his/her result.
4. Every voter sums the polynomial outputs they've received.
5. All k sums the voters compute are used to find the constant term of a polynomial whose constant term is precisely the sum of all ballot.

Demo

There's a problem...

There's a problem...

What's to stop someone from putting an invalid ballot in the constant term of their polynomial that sways the election in their favor?

There's a problem...

What's to stop someone from putting an invalid ballot in the constant term of their polynomial that sways the election in their favor?

For example, in a "yes/no" election, someone could put 2 for their ballot and have the result of 2 votes.

The Remedy

Where I got all of the following material:

The Remedy

Where I got all of the following material:
`https://vitalik.ca/general/2017/11/09/starks_part_1.html`

The Remedy

Where I got all of the following material:
`https://vitalik.ca/general/2017/11/09/starks_part_1.html`
The following material was very rushed.

- We want to prove that a given P_i constant term is valid *without* revealing what it is. This is equivalent to checking whether $P_i(0)$ is valid. This is the essence of a Zero-Knowledge Proof.

- We want to prove that a given P_i constant term is valid *without* revealing what it is. This is equivalent to checking whether $P_i(0)$ is valid. This is the essence of a Zero-Knowledge Proof.
- Now, let $C(x)$ be a *constraint checking polynomial* that is zero if x is a valid constant and nonzero otherwise. For example, if we assume a valid constant is one that is either a zero or one, we can construct $C(x)$ very simply:

$$C(x) = (x - 0)(x - 1) = x^2 - x$$

- We want to prove that a given P_i constant term is valid *without* revealing what it is. This is equivalent to checking whether $P_i(0)$ is valid. This is the essence of a Zero-Knowledge Proof.
- Now, let $C(x)$ be a *constraint checking polynomial* that is zero if x is a valid constant and nonzero otherwise. For example, if we assume a valid constant is one that is either a zero or one, we can construct $C(x)$ very simply:

$$C(x) = (x - 0)(x - 1) = x^2 - x$$

- Now, we can restate the problem as: we need to prove that $C(P(x)) = 0$ when $x = 0$.

- Let $Z(x) = x$

[2] see commitment schemes

- Let $Z(x) = x$
- It's a known mathematical fact that any polynomial that is zero at $x = 0$ must be a multiple of Z. Therefore, there exists some $D(x)$ such that

$$C(P(x)) = Z(x) \cdot D(x)$$

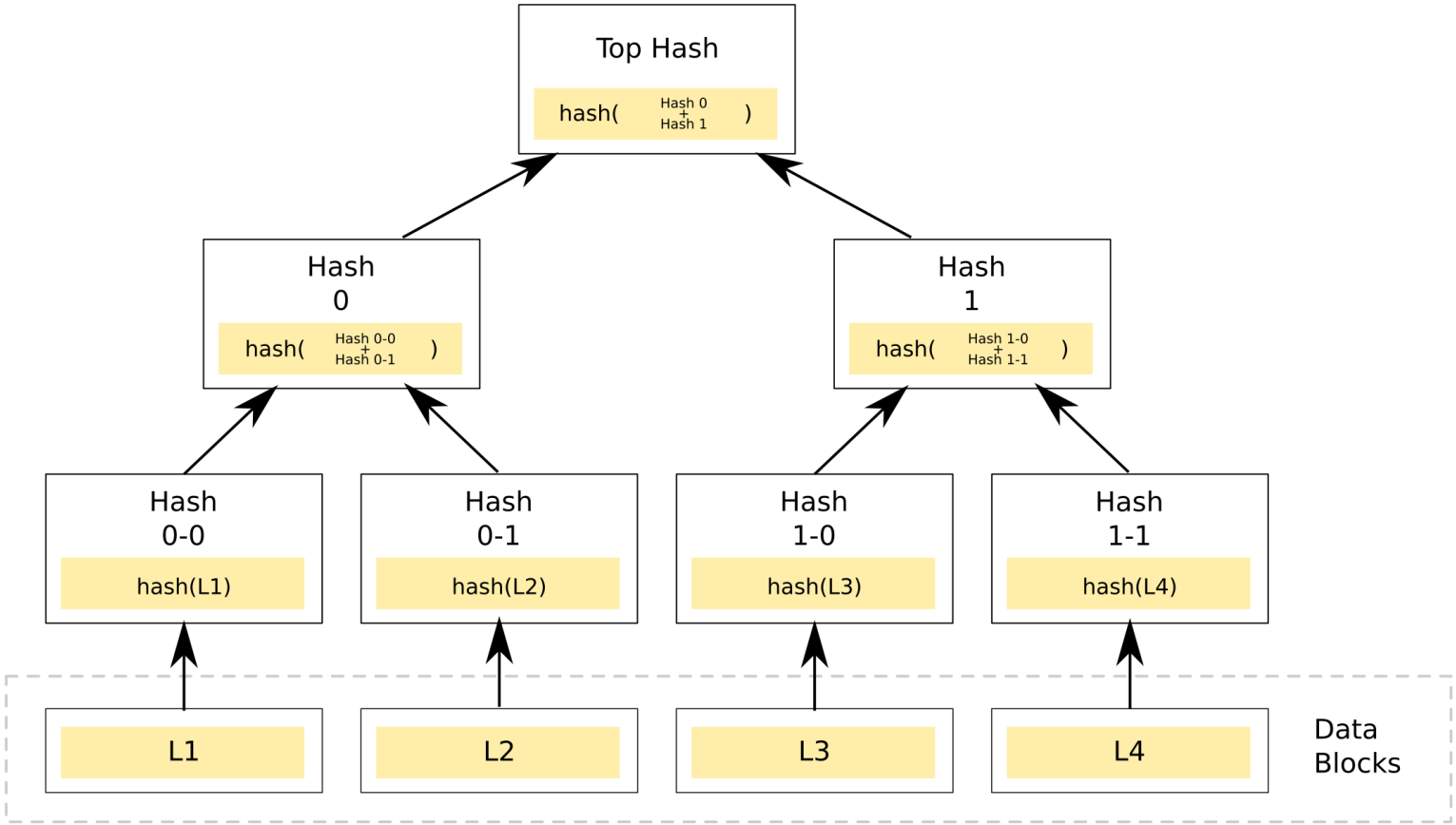
[2] see commitment schemes

- Let $Z(x) = x$
- It's a known mathematical fact that any polynomial that is zero at $x = 0$ must be a multiple of Z. Therefore, there exists some $D(x)$ such that

$$C(P(x)) = Z(x) \cdot D(x)$$

- Before anything, everyone "commits" [2] to their polynomial by creating a merkle tree of the outputs of $P(x)$ and $D(x)$ values and sending the root of the tree to everyone.

[2] see commitment schemes


Top Hash
hash(Hash 0 + Hash 1)
Hash 0
hash(Hash 0-0 + Hash 0-1)
Hash 1
hash(Hash 1-0 + Hash 1-1)
Hash 0-0
hash(L1)
Hash 0-1
hash(L2)
Hash 1-0
hash(L3)
Hash 1-1
hash(L4)
L1
L2
L3
L4
Data Blocks