Aggregatable Subvector Commitments for Stateless Cryptocurrencies

Alin Tomescu[1]
@alinush407

Ittai Abraham[1]
@ittaia

Vitalik Buterin[2]
@VitalikButerin

Justin Drake[2]
@drakefjustin

Dankrad Feist[2]
@dankrad

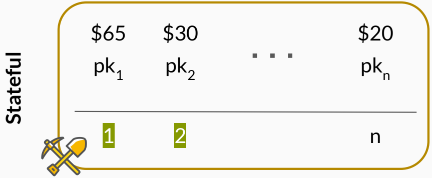
Dmitry Khovratovich[2]
@Khovr

[1]VMware Research, [2]Ethereum Foundation

September 14th, 2020

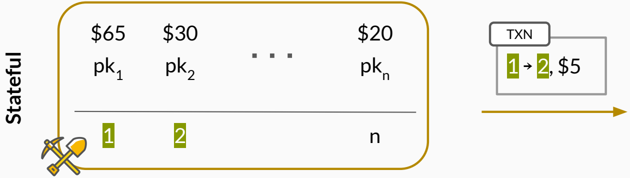
Motivation

Miners rely on state to validate transactions and blocks.



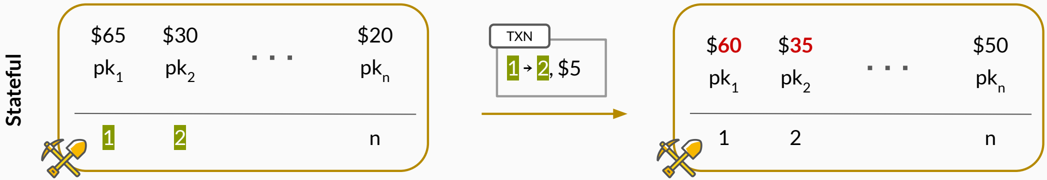
Motivation

Miners rely on state to validate transactions and blocks.



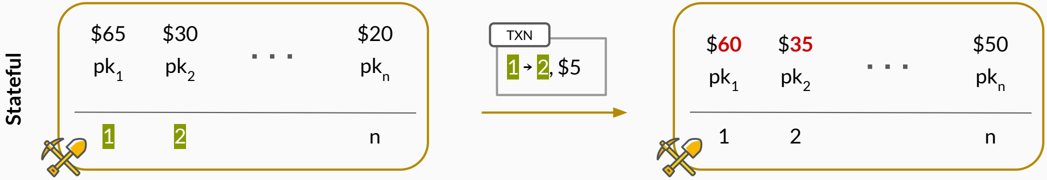
Motivation

Miners rely on state to validate transactions and blocks.



Motivation

Miners rely on state to validate transactions and blocks.



Validation state can be **very large**:

Motivation

Miners rely on state to validate transactions and blocks.

Validation state can be **very large**:

- Hundreds of GBs in Ethereum

Motivation

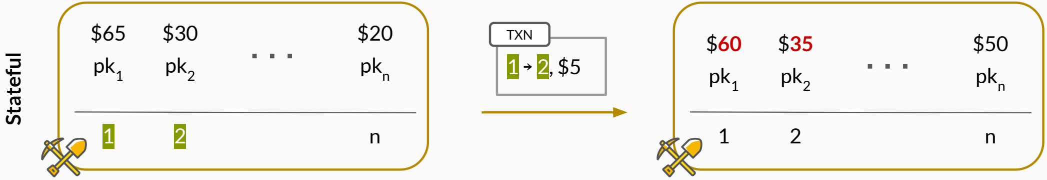
Miners rely on state to validate transactions and blocks.

Validation state can be **very large**:

- Hundreds of GBs in Ethereum
- GBs in Bitcoin

Motivation

Miners rely on state to validate transactions and blocks.



Validation state can be **very large**:

- Hundreds of GBs in Ethereum
- GBs in Bitcoin

This poses scalability challenges:

Motivation

Miners rely on state to validate transactions and blocks.

Validation state can be **very large**:

- Hundreds of GBs in Ethereum
- GBs in Bitcoin

This poses scalability challenges:

- Consensus via sharding

Motivation

Miners rely on state to validate transactions and blocks.

Validation state can be **very large**:

- Hundreds of GBs in Ethereum
- GBs in Bitcoin

This poses scalability challenges:

- Consensus via sharding
- Barrier to entry for P2P nodes

Motivation

Miners rely on state to validate transactions and blocks.

Validation state can be **very large**:

- Hundreds of GBs in Ethereum
- GBs in Bitcoin

This poses scalability challenges:

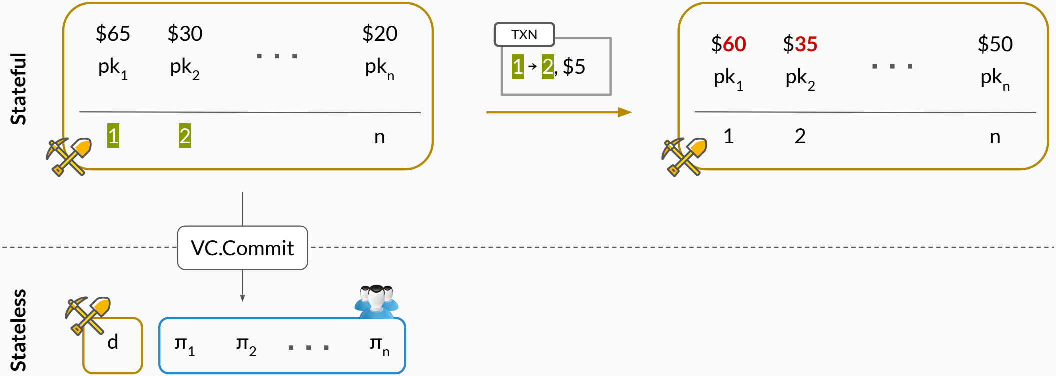
- Consensus via sharding
- Barrier to entry for P2P nodes
- DoS attacks

Previous Work Outsources State via *Vector Commitments (VCs)*

[CPZ18] was first to proposed stateless validation using VCs!

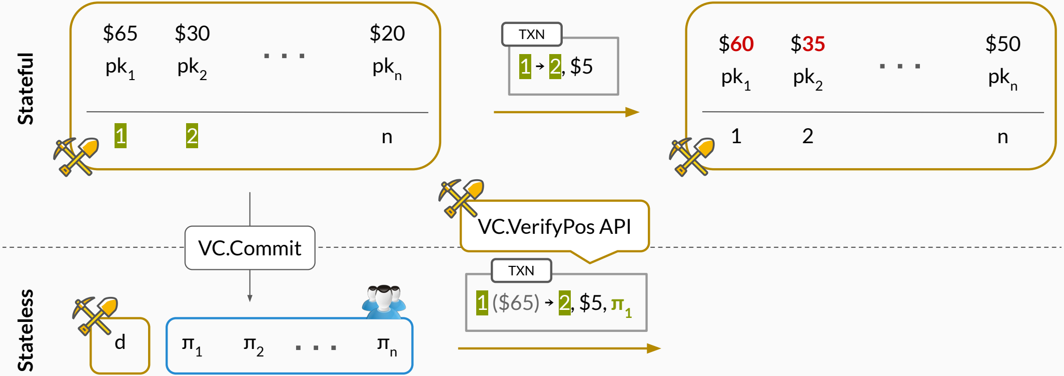
Previous Work Outsources State via *Vector Commitments (VCs)*

[CPZ18] was first to proposed stateless validation using VCs!



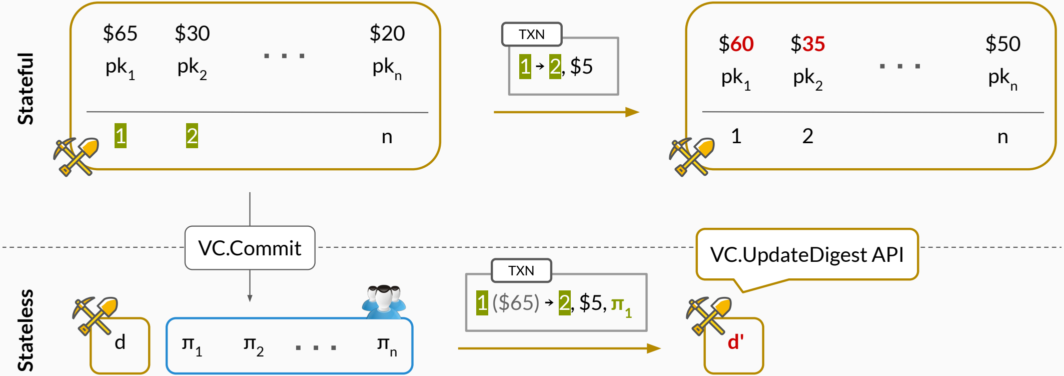
Previous Work Outsources State via *Vector Commitments (VCs)*

[CPZ18] was first to proposed stateless validation using VCs!



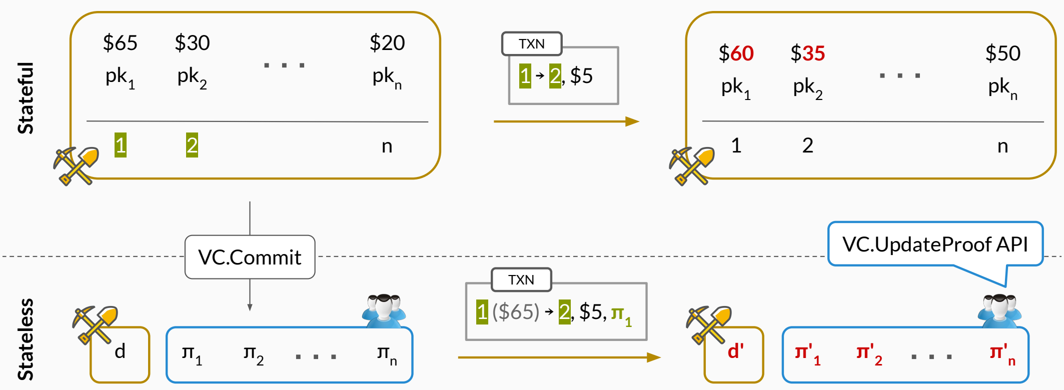
Previous Work Outsources State via *Vector Commitments (VCs)*

[CPZ18] was first to proposed stateless validation using VCs!



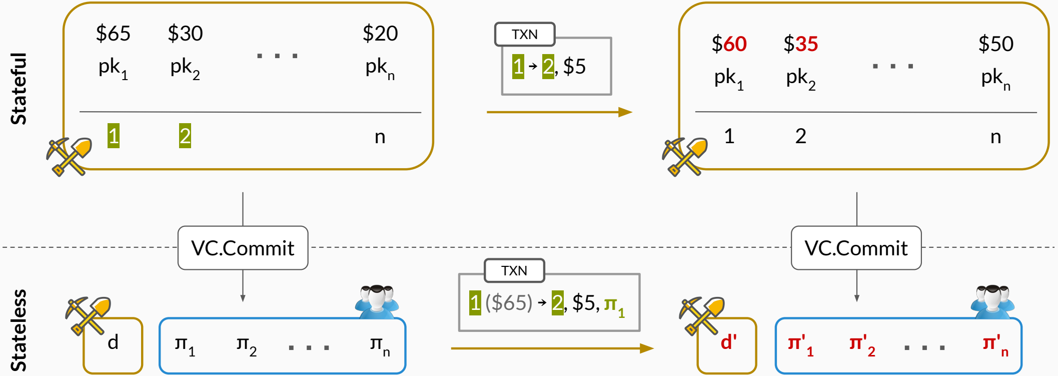
Previous Work Outsources State via *Vector Commitments (VCs)*

[CPZ18] was first to proposed stateless validation using VCs!



Previous Work Outsources State via *Vector Commitments (VCs)*

[CPZ18] was first to proposed stateless validation using VCs!



Our contributions: Efficient VC from KZG polynomial commitments [KZG10]

Our contributions: Efficient VC from KZG polynomial commitments [KZG10]

- Constant-sized, aggregatable and updatable proofs

Our contributions: Efficient VC from KZG polynomial commitments [KZG10]

- Constant-sized, aggregatable and updatable proofs
- Constant-sized update keys

Our contributions: Efficient VC from KZG polynomial commitments [KZG10]

- Constant-sized, aggregatable and updatable proofs
- Constant-sized update keys
- Quasilinear-time proof pre-computation, via Feist-Khovratovich (FK) technique [FK20]

Our contributions: Efficient VC from KZG polynomial commitments [KZG10]

- Constant-sized, aggregatable and updatable proofs
- Constant-sized update keys
- Quasilinear-time proof pre-computation, via Feist-Khovratovich (FK) technique [FK20]
- **Previous work** is either: non-aggregatable

Our contributions: Efficient VC from KZG polynomial commitments [KZG10]

- Constant-sized, aggregatable and updatable proofs
- Constant-sized update keys
- Quasilinear-time proof pre-computation, via Feist-Khovratovich (FK) technique [FK20]
- **Previous work** is either: non-aggregatable, or has linear-sized update keys

Our contributions: Efficient VC from KZG polynomial commitments [KZG10]

- Constant-sized, aggregatable and updatable proofs
- Constant-sized update keys
- Quasilinear-time proof pre-computation, via Feist-Khovratovich (FK) technique [FK20]
- **Previous work** is either: non-aggregatable, or has linear-sized update keys, or is based on less efficient, hidden-order groups

Our contributions: Efficient VC from KZG polynomial commitments [KZG10]

- Constant-sized, aggregatable and updatable proofs
- Constant-sized update keys
- Quasilinear-time proof pre-computation, via Feist-Khovratovich (FK) technique [FK20]
- **Previous work** is either: non-aggregatable, or has linear-sized update keys, or is based on less efficient, hidden-order groups

Additionally,

Our contributions: Efficient VC from KZG polynomial commitments [KZG10]

- Constant-sized, aggregatable and updatable proofs
- Constant-sized update keys
- Quasilinear-time proof pre-computation, via Feist-Khovratovich (FK) technique [FK20]
- **Previous work** is either: non-aggregatable, or has linear-sized update keys, or is based on less efficient, hidden-order groups

Additionally,

- New security definition for KZG batch proofs, proven secure under n-SBDH

Our contributions: Efficient VC from KZG polynomial commitments [KZG10]

- Constant-sized, aggregatable and updatable proofs
- Constant-sized update keys
- Quasilinear-time proof pre-computation, via Feist-Khovratovich (FK) technique [FK20]
- **Previous work** is either: non-aggregatable, or has linear-sized update keys, or is based on less efficient, hidden-order groups

Additionally,

- New security definition for KZG batch proofs, proven secure under n-SBDH
- *Publicly-derivable* parameters from "powers-of-tau" parameters

Our contributions: Efficient VC from KZG polynomial commitments [KZG10]

- Constant-sized, aggregatable and updatable proofs
- Constant-sized update keys
- Quasilinear-time proof pre-computation, via Feist-Khovratovich (FK) technique [FK20]
- **Previous work** is either: non-aggregatable, or has linear-sized update keys, or is based on less efficient, hidden-order groups

Additionally,

- New security definition for KZG batch proofs, proven secure under n-SBDH
- *Publicly-derivable* parameters from “powers-of-tau“ parameters
    - Keeps trusted setup simple

Our contributions: Efficient VC from KZG polynomial commitments [KZG10]

- Constant-sized, aggregatable and updatable proofs
- Constant-sized update keys
- Quasilinear-time proof pre-computation, via Feist-Khovratovich (FK) technique [FK20]
- **Previous work** is either: non-aggregatable, or has linear-sized update keys, or is based on less efficient, hidden-order groups

Additionally,

- New security definition for KZG batch proofs, proven secure under n-SBDH
- *Publicly-derivable* parameters from “powers-of-tau“ parameters
  - Keeps trusted setup simple
  - Keeps parameters *updatable*

Our contributions: Efficient VC from KZG polynomial commitments [KZG10]

- Constant-sized, aggregatable and updatable proofs
- Constant-sized update keys
- Quasilinear-time proof pre-computation, via Feist-Khovratovich (FK) technique [FK20]
- **Previous work** is either: non-aggregatable, or has linear-sized update keys, or is based on less efficient, hidden-order groups

Additionally,

- New security definition for KZG batch proofs, proven secure under n-SBDH
- *Publicly-derivable* parameters from “powers-of-tau“ parameters
  - Keeps trusted setup simple
  - Keeps parameters *updatable*
- Subtleties of VC-based stateless cryptocurrencies

Our contributions: Efficient VC from KZG polynomial commitments [KZG10]

- Constant-sized, aggregatable and updatable proofs
- Constant-sized update keys
- Quasilinear-time proof pre-computation, via Feist-Khovratovich (FK) technique [FK20]
- **Previous work** is either: non-aggregatable, or has linear-sized update keys, or is based on less efficient, hidden-order groups

Additionally,

- New security definition for KZG batch proofs, proven secure under n-SBDH
- *Publicly-derivable* parameters from "powers-of-tau" parameters
  - Keeps trusted setup simple
  - Keeps parameters *updatable*
- Subtleties of VC-based stateless cryptocurrencies
  - Keeping track of transaction counters

Our contributions: Efficient VC from KZG polynomial commitments [KZG10]

- Constant-sized, aggregatable and updatable proofs
- Constant-sized update keys
- Quasilinear-time proof pre-computation, via Feist-Khovratovich (FK) technique [FK20]
- **Previous work** is either: non-aggregatable, or has linear-sized update keys, or is based on less efficient, hidden-order groups

Additionally,

- New security definition for KZG batch proofs, proven secure under n-SBDH
- *Publicly-derivable* parameters from “powers-of-tau“ parameters
    - Keeps trusted setup simple
    - Keeps parameters *updatable*
- Subtleties of VC-based stateless cryptocurrencies
    - Keeping track of transaction counters
    - Verifiable update keys

Our contributions: Efficient VC from KZG polynomial commitments [KZG10]

- Constant-sized, aggregatable and updatable proofs
- Constant-sized update keys
- Quasilinear-time proof pre-computation, via Feist-Khovratovich (FK) technique [FK20]
- **Previous work** is either: non-aggregatable, or has linear-sized update keys, or is based on less efficient, hidden-order groups

Additionally,

- New security definition for KZG batch proofs, proven secure under n-SBDH
- *Publicly-derivable* parameters from “powers-of-tau“ parameters
  - Keeps trusted setup simple
  - Keeps parameters *updatable*
- Subtleties of VC-based stateless cryptocurrencies
  - Keeping track of transaction counters
  - Verifiable update keys
  - DoS attacks on new user registration

Thank you!

Paper is too long? **Read our blogpost!**

https://alinush.github.io/2020/05/06/aggregatable-subvector-commitments-for-stateless-cryptocurrencies.html

Appendix

Outline

Aggregatable Subvector Commitments (aSVCs) versus Previous Work

Aggregatable Subvector Commitments (aSVCs) versus Previous Work

Table 1: Asymptotic comparison to previous (aS)VCs. n is the size of the vector $\vec{v}$.

(aS)VC scheme	Public parameters	Proof size	Update key size	Digest update	Aggr. b proofs	Prove all

Aggregatable Subvector Commitments (aSVCs) versus Previous Work

Table 1: Asymptotic comparison to previous (aS)VCs. n is the size of the vector $\vec{v}$.

(aS)VC scheme	Public parameters	Proof size	Update key size	Digest update	Aggr. b proofs	Prove all
Merkle trees [Mer88]	1	$\log n$	×	×	×	n

Aggregatable Subvector Commitments (aSVCs) versus Previous Work

Table 1: Asymptotic comparison to previous (aS)VCs. n is the size of the vector $\vec{v}$.

(aS)VC scheme	Public parameters	Proof size	Update key size	Digest update	Aggr. b proofs	Prove all
Merkle trees [Mer88]	1	$\log n$	✗	✗	✗	n
CDHK [CDHK15]	n	1	n	✔	✗	n^2
CPZ [CPZ18]	n	$\log n$	$\log n$	✔	✗	$n \log n$
TCZ [TCZ+20, Tom20]	n	$\log n$	$\log n$	✔	✗	$n \log n$
Pointproofs [GRWZ20]	n	1	n	✔	$b_{\mathbb{G}}$	n^2

Aggregatable Subvector Commitments (aSVCs) versus Previous Work

Table 1: Asymptotic comparison to previous (aS)VCs. n is the size of the vector $\vec{v}$.

(aS)VC scheme	Public parameters	Proof size	Update key size	Digest update	Aggr. b proofs	Prove all
Merkle trees [Mer88]	1	$\log n$	✗	✗	✗	n
CDHK [CDHK15]	n	1	n	✔	✗	n^2
CPZ [CPZ18]	n	$\log n$	$\log n$	✔	✗	$n \log n$
TCZ [TCZ⁺20, Tom20]	n	$\log n$	$\log n$	✔	✗	$n \log n$
Pointproofs [GRWZ20]	n	1	n	✔	$b_{\mathbb{G}}$	n^2
BBF [BBF19]	1	$1_{\mathbb{G}_?}$	✗	✗	$b \log n_{\mathbb{G}_?}$	$n \log n_{\mathbb{G}_?}$
CFG_1 [CFG⁺20]	1	$1_{\mathbb{G}_?}$	✗	✗	$b \log b \log n_{\mathbb{G}_?}$	$n \log^2 n_{\mathbb{G}_?}$

Aggregatable Subvector Commitments (aSVCs) versus Previous Work

Table 1: Asymptotic comparison to previous (aS)VCs. n is the size of the vector $\vec{v}$.

(aS)VC scheme	Public parameters	Proof size	Update key size	Digest update	Aggr. b proofs	Prove all
Merkle trees [Mer88]	1	$\log n$	✗	✗	✗	n
CDHK [CDHK15]	n	1	n	✔	✗	n^2
CPZ [CPZ18]	n	$\log n$	$\log n$	✔	✗	$n \log n$
TCZ [TCZ+20, Tom20]	n	$\log n$	$\log n$	✔	✗	$n \log n$
Pointproofs [GRWZ20]	n	1	n	✔	$b_{\mathbb{G}}$	n^2
BBF [BBF19]	1	$1_{\mathbb{G}_?}$	✗	✗	$b \log n_{\mathbb{G}_?}$	$n \log n_{\mathbb{G}_?}$
CFG_1 [CFG+20]	1	$1_{\mathbb{G}_?}$	✗	✗	$b \log b \log n_{\mathbb{G}_?}$	$n \log^2 n_{\mathbb{G}_?}$
CFG_2 [CF13, LM19, CFG+20]	1	$1_{\mathbb{G}_?}$	$1_{\mathbb{G}_?}$	✔	$b \log^2 b_{\mathbb{G}_?}$	$n \log^2 n_{\mathbb{G}_?}$

Aggregatable Subvector Commitments (aSVCs) versus Previous Work

Table 1: Asymptotic comparison to previous (aS)VCs. n is the size of the vector $\vec{v}$.

(aS)VC scheme	Public parameters	Proof size	Update key size	Digest update	Aggr. b proofs	Prove all
Merkle trees [Mer88]	1	$\log n$	✗	✗	✗	n
CDHK [CDHK15]	n	1	n	✔	✗	n^2
CPZ [CPZ18]	n	$\log n$	$\log n$	✔	✗	$n \log n$
TCZ [TCZ+20, Tom20]	n	$\log n$	$\log n$	✔	✗	$n \log n$
Pointproofs [GRWZ20]	n	1	n	✔	$b_{\mathbb{G}}$	n^2
BBF [BBF19]	1	$1_{\mathbb{G}_?}$	✗	✗	$b \log n_{\mathbb{G}_?}$	$n \log n_{\mathbb{G}_?}$
CFG_1 [CFG+20]	1	$1_{\mathbb{G}_?}$	✗	✗	$b \log b \log n_{\mathbb{G}_?}$	$n \log^2 n_{\mathbb{G}_?}$
CFG_2 [CF13, LM19, CFG+20]	1	$1_{\mathbb{G}_?}$	$1_{\mathbb{G}_?}$	✔	$b \log^2 b_{\mathbb{G}_?}$	$n \log^2 n_{\mathbb{G}_?}$
Our aSVC	n	1	1	✔	$b \lg^2 b_{\mathbb{F}} + b_{\mathbb{G}}$	$n \log n$

Some limitations

Unlike schemes based on hidden-order groups [CF13, LM19, BBF19, CFG⁺20], we have:

Some limitations

Unlike schemes based on hidden-order groups [CF13, LM19, BBF19, CFG+20], we have:

- Trusted setup

Some limitations

Unlike schemes based on hidden-order groups [CF13, LM19, BBF19, CFG⁺20], we have:

- Trusted setup
- No *incremental* aggregation & no dis-aggregation [CFG⁺20]

Some limitations

Unlike schemes based on hidden-order groups [CF13, LM19, BBF19, CFG+20], we have:

- Trusted setup
- No *incremental* aggregation & no dis-aggregation [CFG+20]
- No space-time trade-off for proof pre-computation [BBF19, CFG+20]

Outline

KZG Constant-sized Polynomial Commitments [KZG10]

KZG Constant-sized Polynomial Commitments [KZG10]

Fix n-SDH public parameters $\left(g^{\tau^i}\right)_{0 \le i \le n}$ such that trapdoor $\tau \in \mathbb{F}_p$ is unknown.

KZG Constant-sized Polynomial Commitments [KZG10]

Fix n-SDH public parameters $\left(g^{\tau^i}\right)_{0 \le i \le n}$ such that trapdoor $\tau \in \mathbb{F}_p$ is unknown.

Committing: Given $\phi \in \mathbb{F}_p[X]$

KZG Constant-sized Polynomial Commitments [KZG10]

Fix n-SDH public parameters $\left(g^{\tau^i}\right)_{0 \le i \le n}$ such that trapdoor $\tau \in \mathbb{F}_p$ is unknown.

Committing: Given $\phi \in \mathbb{F}_p[X]$, where $\phi(X) = \sum_{i=0}^{n} \phi_i X^i = \langle \phi_0, \phi_1, \dots, \phi_n \rangle$

KZG Constant-sized Polynomial Commitments [KZG10]

Fix n-SDH public parameters $\left(g^{\tau^i}\right)_{0 \le i \le n}$ such that trapdoor $\tau \in \mathbb{F}_p$ is unknown.

Committing: Given $\phi \in \mathbb{F}_p[X]$, where $\phi(X) = \sum_{i=0}^{n} \phi_i X^i = \langle \phi_0, \phi_1, \dots, \phi_n \rangle$ of degree $\le n$:

KZG Constant-sized Polynomial Commitments [KZG10]

Fix n-SDH public parameters $\left(g^{\tau^i}\right)_{0 \le i \le n}$ such that trapdoor $\tau \in \mathbb{F}_p$ is unknown.

Committing: Given $\phi \in \mathbb{F}_p[X]$, where $\phi(X) = \sum_{i=0}^{n} \phi_i X^i = \langle \phi_0, \phi_1, \dots, \phi_n \rangle$ of degree $\le n$:

$$c(\phi) = g^{\phi(\tau)} \tag{1}$$

$$\tag{2}$$

KZG Constant-sized Polynomial Commitments [KZG10]

Fix n-SDH public parameters $\left(g^{\tau^i}\right)_{0 \le i \le n}$ such that trapdoor $\tau \in \mathbb{F}_p$ is unknown.

Committing: Given $\phi \in \mathbb{F}_p[X]$, where $\phi(X) = \sum_{i=0}^{n} \phi_i X^i = \langle \phi_0, \phi_1, \dots, \phi_n \rangle$ of degree $\le n$:

$$
\begin{aligned}
c(\phi) &= g^{\phi(\tau)} && (1)\\
&= \left(g^{\tau^n}\right)^{\phi_n} \left(g^{\tau^{n-1}}\right)^{\phi_{n-1}} \dots (g^{\tau})^{\phi_1} (g)^{\phi_0} && (2)
\end{aligned}
$$

KZG Constant-sized Polynomial Commitments [KZG10]

Fix n-SDH public parameters $\left(g^{\tau^i}\right)_{0 \le i \le n}$ such that trapdoor $\tau \in \mathbb{F}_p$ is unknown.

Committing: Given $\phi \in \mathbb{F}_p[X]$, where $\phi(X) = \sum_{i=0}^{n} \phi_i X^i = \langle \phi_0, \phi_1, \dots, \phi_n \rangle$ of degree $\le n$:

$$c(\phi) = g^{\phi(\tau)} \quad (1)$$

$$= \left(g^{\tau^n}\right)^{\phi_n} \left(g^{\tau^{n-1}}\right)^{\phi_{n-1}} \dots (g^{\tau})^{\phi_1} (g)^{\phi_0} \quad (2)$$

Homomorphism: For all field elements a, b and polynomials $\phi(X), \psi(X)$, we have:

KZG Constant-sized Polynomial Commitments [KZG10]

Fix n-SDH public parameters $\left(g^{\tau^i}\right)_{0 \le i \le n}$ such that trapdoor $\tau \in \mathbb{F}_p$ is unknown.

Committing: Given $\phi \in \mathbb{F}_p[X]$, where $\phi(X) = \sum_{i=0}^{n} \phi_i X^i = \langle \phi_0, \phi_1, \dots, \phi_n \rangle$ of degree $\le n$:

$$c(\phi) = g^{\phi(\tau)} \tag{1}$$

$$= \left(g^{\tau^n}\right)^{\phi_n} \left(g^{\tau^{n-1}}\right)^{\phi_{n-1}} \dots (g^{\tau})^{\phi_1} (g)^{\phi_0} \tag{2}$$

Homomorphism: For all field elements a, b and polynomials $\phi(X), \psi(X)$, we have:

$$c(a \cdot \phi + b \cdot \psi) = g^{a \cdot \phi(\tau) + b \cdot \psi(\tau)} \tag{3}$$

(4)

(5)

KZG Constant-sized Polynomial Commitments [KZG10]

Fix n-SDH public parameters $\left(g^{\tau^i}\right)_{0 \le i \le n}$ such that trapdoor $\tau \in \mathbb{F}_p$ is unknown.

Committing: Given $\phi \in \mathbb{F}_p[X]$, where $\phi(X) = \sum_{i=0}^{n} \phi_i X^i = \langle \phi_0, \phi_1, \ldots, \phi_n \rangle$ of degree $\le n$:

$$c(\phi) = g^{\phi(\tau)} \tag{1}$$

$$= \left(g^{\tau^n}\right)^{\phi_n} \left(g^{\tau^{n-1}}\right)^{\phi_{n-1}} \ldots (g^{\tau})^{\phi_1} (g)^{\phi_0} \tag{2}$$

Homomorphism: For all field elements a, b and polynomials $\phi(X), \psi(X)$, we have:

$$c(a \cdot \phi + b \cdot \psi) = g^{a \cdot \phi(\tau) + b \cdot \psi(\tau)} \tag{3}$$

$$= \left(g^{\phi(\tau)}\right)^a \left(g^{\psi(\tau)}\right)^b \tag{4}$$

$$\tag{5}$$

KZG Constant-sized Polynomial Commitments [KZG10]

Fix n-SDH public parameters $\left(g^{\tau^i}\right)_{0 \le i \le n}$ such that trapdoor $\tau \in \mathbb{F}_p$ is unknown.

Committing: Given $\phi \in \mathbb{F}_p[X]$, where $\phi(X) = \sum_{i=0}^{n} \phi_i X^i = \langle \phi_0, \phi_1, \dots, \phi_n \rangle$ of degree $\le n$:

$$
\begin{aligned}
c(\phi) &= g^{\phi(\tau)} && (1)\\
&= \left(g^{\tau^n}\right)^{\phi_n} \left(g^{\tau^{n-1}}\right)^{\phi_{n-1}} \dots (g^{\tau})^{\phi_1} (g)^{\phi_0} && (2)
\end{aligned}
$$

Homomorphism: For all field elements a, b and polynomials $\phi(X), \psi(X)$, we have:

$$
\begin{aligned}
c(a \cdot \phi + b \cdot \psi) &= g^{a \cdot \phi(\tau) + b \cdot \psi(\tau)} && (3)\\
&= \left(g^{\phi(\tau)}\right)^{a} \left(g^{\psi(\tau)}\right)^{b} && (4)\\
&= c(\phi)^{a} c(\psi)^{b} && (5)
\end{aligned}
$$

VCs from Lagrange Basis Polynomials [CDHK15]

VCs from Lagrange Basis Polynomials [CDHK15]

Represent vector $\vec{v}$ with a polynomial ϕ s.t. $\phi(i) = v_i$:

VCs from Lagrange Basis Polynomials [CDHK15]

Represent vector $\vec{v}$ with a polynomial ϕ s.t. $\phi(i) = v_i$:

$$\phi(X) = \sum_{i=0}^{n-1} v_i \cdot L_i(X) \tag{6}$$

$$\tag{7}$$

VCs from Lagrange Basis Polynomials [CDHK15]

Represent vector $\vec{v}$ with a polynomial ϕ s.t. $\phi(i) = v_i$:

$$\phi(X) = \sum_{i=0}^{n-1} v_i \cdot L_i(X) \tag{6}$$

$$L_i(X) = \prod_{\substack{j \in [0,n) \\ j \neq i}} \frac{X - j}{i - j} \tag{7}$$

VCs from Lagrange Basis Polynomials [CDHK15]

Represent vector $\vec{v}$ with a polynomial ϕ s.t. $\phi(i) = v_i$:

$$\phi(X) = \sum_{i=0}^{n-1} v_i \cdot L_i(X) \tag{6}$$

$$L_i(X) = \prod_{\substack{j \in [0,n) \\ j \neq i}} \frac{X - j}{i - j} \tag{7}$$

Applying the KZG homomorphism to Equation (6):

VCs from Lagrange Basis Polynomials [CDHK15]

Represent vector $\vec{v}$ with a polynomial ϕ s.t. $\phi(i) = v_i$:

$$\phi(X) = \sum_{i=0}^{n-1} v_i \cdot L_i(X) \tag{6}$$

$$L_i(X) = \prod_{\substack{j \in [0,n) \\ j \neq i}} \frac{X - j}{i - j} \tag{7}$$

Applying the KZG homomorphism to Equation (6):

$$c(\phi) = \prod_{i=0}^{n-1} c\left(L_i\right)^{v_i} \tag{8}$$

VCs from Lagrange Basis Polynomials [CDHK15]

Represent vector $\vec{v}$ with a polynomial ϕ s.t. $\phi(i) = v_i$:

$$\phi(X) = \sum_{i=0}^{n-1} v_i \cdot L_i(X) \tag{6}$$

$$L_i(X) = \prod_{\substack{j \in [0,n) \\ j \neq i}} \frac{X - j}{i - j} \tag{7}$$

Applying the KZG homomorphism to Equation (6):

$$c(\phi) = \prod_{i=0}^{n-1} c(L_i)^{v_i} \tag{8}$$

Note: Public parameters include commitments $c(L_i)$.

VCs from Lagrange Basis Polynomials [CDHK15]

Represent vector $\vec{v}$ with a polynomial ϕ s.t. $\phi(i) = v_i$:

$$\phi(X) = \sum_{i=0}^{n-1} v_i \cdot L_i(X) \tag{6}$$

$$L_i(X) = \prod_{\substack{j \in [0,n) \\ j \neq i}} \frac{X - j}{i - j} \tag{7}$$

Applying the KZG homomorphism to Equation (6):

$$c(\phi) = \prod_{i=0}^{n-1} c(L_i)^{v_i} \tag{8}$$

Note: Public parameters include commitments $c(L_i)$. *Can derive from* g^{τ^i}*'s.*

Updating the Digest = Updating Polynomial & Updating Commitment

Updating the Digest = Updating Polynomial & Updating Commitment

Assume v_i changed to $v_i + \delta_i$.

Updating the Digest = Updating Polynomial & Updating Commitment

Assume v_i changed to $v_i + \delta_i$. Old polynomial was:

$$\phi(X) = \sum_{i=0}^{n-1} v_i \cdot L_i(X) \tag{9}$$

Updating the Digest = Updating Polynomial & Updating Commitment

Assume v_i changed to $v_i + \delta_i$. Old polynomial was:

$$\phi(X) = \sum_{i=0}^{n-1} v_i \cdot L_i(X) \tag{9}$$

Updated polynomial will be:

$$\phi'(X) = \phi(X) + \delta_i L_i(X) \tag{10}$$

Updating the Digest = Updating Polynomial & Updating Commitment

Assume v_i changed to $v_i + \delta_i$. Old polynomial was:

$$\phi(X) = \sum_{i=0}^{n-1} v_i \cdot L_i(X) \tag{9}$$

Updated polynomial will be:

$$\phi'(X) = \phi(X) + \delta_i L_i(X) \tag{10}$$

Updated commitment will be:

$$c(\phi') = c(\phi) \cdot c\left(L_i\right)^{\delta_i} \tag{11}$$

Updating the Digest = Updating Polynomial & Updating Commitment

Assume v_i changed to $v_i + \delta_i$. Old polynomial was:

$$\phi(X) = \sum_{i=0}^{n-1} v_i \cdot L_i(X) \tag{9}$$

Updated polynomial will be:

$$\phi'(X) = \phi(X) + \delta_i L_i(X) \tag{10}$$

Updated commitment will be:

$$c(\phi') = c(\phi) \cdot c\left(L_i\right)^{\delta_i} \tag{11}$$

Thus, for our purposes, each upk_i will include $c\left(L_i\right)$.

Proof π_i for v_i Refresher

Proof π_i for v_i Refresher

A proof π_i for v_i must convince that $\phi(i) = v_i$

Proof π_i for v_i Refresher

A proof π_i for v_i must convince that $\phi(i) = v_i \Leftrightarrow \phi \bmod (X - i) = v_i$.

Proof π_i for v_i Refresher

A proof π_i for v_i must convince that $\phi(i) = v_i \Leftrightarrow \phi \bmod (X - i) = v_i$.

$$q_i(X) = \frac{\phi(X) - v_i}{X - i} \tag{12}$$

$$\tag{13}$$

Proof π_i for v_i Refresher

A proof π_i for v_i must convince that $\phi(i) = v_i \Leftrightarrow \phi \bmod (X - i) = v_i$.

$$q_i(X) = \frac{\phi(X) - v_i}{X - i} \tag{12}$$

$$\pi_i = c\left(q_i\right) = g^{q_i(\tau)} \tag{13}$$

Proof π_i for v_i Refresher

A proof π_i for v_i must convince that $\phi(i) = v_i \Leftrightarrow \phi \bmod (X - i) = v_i$.

$$q_i(X) = \frac{\phi(X) - v_i}{X - i} \tag{12}$$

$$\pi_i = c\left(q_i\right) = g^{q_i(\tau)} \tag{13}$$

To verify, use pairings:

Proof π_i for v_i Refresher

A proof π_i for v_i must convince that $\phi(i) = v_i \Leftrightarrow \phi \bmod (X - i) = v_i$.

$$q_i(X) = \frac{\phi(X) - v_i}{X - i} \tag{12}$$

$$\pi_i = c\left(q_i\right) = g^{q_i(\tau)} \tag{13}$$

To verify, use pairings:

$$e(c(\phi)/g^{v_i}, g) = e(\pi_i, g^{\tau}/g^{i}) \Leftrightarrow \tag{14}$$

$$\tag{15}$$

Proof π_i for v_i Refresher

A proof π_i for v_i must convince that $\phi(i) = v_i \Leftrightarrow \phi \bmod (X - i) = v_i$.

$$q_i(X) = \frac{\phi(X) - v_i}{X - i} \tag{12}$$

$$\pi_i = c\left(q_i\right) = g^{q_i(\tau)} \tag{13}$$

To verify, use pairings:

$$e(c(\phi)/g^{v_i}, g) = e(\pi_i, g^{\tau}/g^{i}) \Leftrightarrow \tag{14}$$

$$\phi(\tau) - v_i = q_i(\tau)(\tau - i) \tag{15}$$

Outline

New Technique: Updating Proof $\pi_i = c(q_i)$ After Change (i, δ_i)

New Technique: Updating Proof $\pi_i = c(q_i)$ After Change (i, δ_i)

We know $\pi_i' = c\left(q_i'\right)$, where:

New Technique: Updating Proof $\pi_i = c(q_i)$ After Change (i, δ_i)

We know $\pi_i' = c\left(q_i'\right)$, where:

$$q_i'(X) = \frac{\phi'(X) - (v_i + \delta_i)}{X - i} \tag{16}$$

(17)

(18)

(19)

New Technique: Updating Proof $\pi_i = c(q_i)$ After Change (i, δ_i)

We know $\pi_i' = c\left(q_i'\right)$, where:

$$
\begin{aligned}
q_i'(X) &= \frac{\phi'(X) - (v_i + \delta_i)}{X - i} && (16) \\
&= \frac{\big(\phi(X) + \delta_i L_i(X)\big) - v_i - \delta_i}{X - i} && (17) \\
& && (18) \\
& && (19)
\end{aligned}
$$

New Technique: Updating Proof $\pi_i = c(q_i)$ After Change (i, δ_i)

We know $\pi_i' = c\left(q_i'\right)$, where:

$$
\begin{aligned}
q_i'(X) &= \frac{\phi'(X) - (v_i + \delta_i)}{X - i} && (16) \\
&= \frac{\left(\phi(X) + \delta_i L_i(X)\right) - v_i - \delta_i}{X - i} && (17) \\
&= \frac{\phi(X) - v_i}{X - i} - \frac{\delta_i(L_i(X) - 1)}{X - i} && (18) \\
& && (19)
\end{aligned}
$$

New Technique: Updating Proof $\pi_i = c(q_i)$ After Change (i, δ_i)

We know $\pi_i' = c\left(q_i'\right)$, where:

$$q_i'(X) = \frac{\phi'(X) - (v_i + \delta_i)}{X - i} \tag{16}$$

$$= \frac{\left(\phi(X) + \delta_i L_i(X)\right) - v_i - \delta_i}{X - i} \tag{17}$$

$$= \frac{\phi(X) - v_i}{X - i} - \frac{\delta_i(L_i(X) - 1)}{X - i} \tag{18}$$

$$q_i'(X) = q_i(X) + \delta_i\left(\frac{L_i(X) - 1}{X - i}\right) \tag{19}$$

New Technique: Updating Proof $\pi_i = c(q_i)$ After Change (i, δ_i)

We know $\pi_i' = c\left(q_i'\right)$, where:

$$q_i'(X) = \frac{\phi'(X) - (v_i + \delta_i)}{X - i} \tag{16}$$

$$= \frac{\left(\phi(X) + \delta_i L_i(X)\right) - v_i - \delta_i}{X - i} \tag{17}$$

$$= \frac{\phi(X) - v_i}{X - i} - \frac{\delta_i(L_i(X) - 1)}{X - i} \tag{18}$$

$$q_i'(X) = q_i(X) + \delta_i \left(\frac{L_i(X) - 1}{X - i}\right) \tag{19}$$

Applying KZG homomorphism, it follows that:

New Technique: Updating Proof $\pi_i = c(q_i)$ After Change (i, δ_i)

We know $\pi_i' = c\left(q_i'\right)$, where:

$$q_i'(X) = \frac{\phi'(X) - (v_i + \delta_i)}{X - i} \tag{16}$$

$$= \frac{\left(\phi(X) + \delta_i L_i(X)\right) - v_i - \delta_i}{X - i} \tag{17}$$

$$= \frac{\phi(X) - v_i}{X - i} - \frac{\delta_i (L_i(X) - 1)}{X - i} \tag{18}$$

$$q_i'(X) = q_i(X) + \delta_i \left(\frac{L_i(X) - 1}{X - i}\right) \tag{19}$$

Applying KZG homomorphism, it follows that:

$$\pi_i' = c\left(q_i'\right) = c\left(q_i\right) \cdot c\left(\frac{L_i(X) - 1}{X - i}\right)^{\delta_i} \tag{20}$$

New Technique: Updating Proof $\pi_i = c(q_i)$ After Change (i, δ_i)

We know $\pi_i' = c\left(q_i'\right)$, where:

$$q_i'(X) = \frac{\phi'(X) - (v_i + \delta_i)}{X - i} \tag{16}$$

$$= \frac{\left(\phi(X) + \delta_i L_i(X)\right) - v_i - \delta_i}{X - i} \tag{17}$$

$$= \frac{\phi(X) - v_i}{X - i} - \frac{\delta_i (L_i(X) - 1)}{X - i} \tag{18}$$

$$q_i'(X) = q_i(X) + \delta_i \left(\frac{L_i(X) - 1}{X - i}\right) \tag{19}$$

Applying KZG homomorphism, it follows that:

$$\pi_i' = c\left(q_i'\right) = c\left(q_i\right) \cdot c\left(\frac{L_i(X) - 1}{X - i}\right)^{\delta_i} \tag{20}$$

Thus, each upk_i must include $c\left(\frac{L_i(X)-1}{X-i}\right)$.

New Technique: Updating Proof $\pi_i = c(q_i)$ After Change (i, δ_i)

We know $\pi_i' = c\left(q_i'\right)$, where:

$$q_i'(X) = \frac{\phi'(X) - (v_i + \delta_i)}{X - i} \tag{16}$$

$$= \frac{\left(\phi(X) + \delta_i L_i(X)\right) - v_i - \delta_i}{X - i} \tag{17}$$

$$= \frac{\phi(X) - v_i}{X - i} - \frac{\delta_i (L_i(X) - 1)}{X - i} \tag{18}$$

$$q_i'(X) = q_i(X) + \delta_i \left(\frac{L_i(X) - 1}{X - i}\right) \tag{19}$$

Applying KZG homomorphism, it follows that:

$$\pi_i' = c\left(q_i'\right) = c\left(q_i\right) \cdot c\left(\frac{L_i(X) - 1}{X - i}\right)^{\delta_i} \tag{20}$$

Thus, each upk_i must include $c\left(\frac{L_i(X)-1}{X-i}\right)$. *Can derive these from g^{τ^i}'s!*

New Technique: Updating Proof $\pi_i = c(q_i)$ After Change (j, δ_j), $j \neq i$

New Technique: Updating Proof $\pi_i = c(q_i)$ After Change (j, δ_j), $j \neq i$

Once again, we know $\pi_i' = c\left(q_i'\right)$, where:

New Technique: Updating Proof $\pi_i = c(q_i)$ After Change (j, δ_j), $j \neq i$

Once again, we know $\pi_i' = c\left(q_i'\right)$, where:

$$q_i'(X) = \frac{\phi'(X) - v_i}{X - i} \tag{21}$$

(22)

(23)

(24)

New Technique: Updating Proof $\pi_i = c(q_i)$ After Change (j, δ_j), $j \neq i$

Once again, we know $\pi_i' = c\left(q_i'\right)$, where:

$$q_i'(X) = \frac{\phi'(X) - v_i}{X - i} \tag{21}$$

$$= \frac{\left(\phi(X) + \delta_j L_j(X)\right) - v_i}{X - i} \tag{22}$$

$$\tag{23}$$

$$\tag{24}$$

New Technique: Updating Proof $\pi_i = c(q_i)$ After Change (j, δ_j), $j \neq i$

Once again, we know $\pi_i' = c\left(q_i'\right)$, where:

$$
\begin{align}
q_i'(X) &= \frac{\phi'(X) - v_i}{X - i} \tag{21}\\
&= \frac{\left(\phi(X) + \delta_j L_j(X)\right) - v_i}{X - i} \tag{22}\\
&= \frac{\phi(X) - v_i}{X - i} - \frac{\delta_j L_j(X)}{X - i} \tag{23}\\
& \tag{24}
\end{align}
$$

New Technique: Updating Proof $\pi_i = c(q_i)$ After Change (j, δ_j), $j \neq i$

Once again, we know $\pi_i' = c\left(q_i'\right)$, where:

$$
\begin{aligned}
q_i'(X) &= \frac{\phi'(X) - v_i}{X - i} && (21)\\
&= \frac{\left(\phi(X) + \delta_j L_j(X)\right) - v_i}{X - i} && (22)\\
&= \frac{\phi(X) - v_i}{X - i} - \frac{\delta_j L_j(X)}{X - i} && (23)\\
q_i'(X) &= q_i(X) + \delta_j \left(\frac{L_j(X)}{X - i}\right) && (24)
\end{aligned}
$$

New Technique: Updating Proof $\pi_i = c(q_i)$ After Change (j, δ_j), $j \neq i$

Once again, we know $\pi_i' = c\left(q_i'\right)$, where:

$$q_i'(X) = \frac{\phi'(X) - v_i}{X - i} \tag{21}$$

$$= \frac{\left(\phi(X) + \delta_j L_j(X)\right) - v_i}{X - i} \tag{22}$$

$$= \frac{\phi(X) - v_i}{X - i} - \frac{\delta_j L_j(X)}{X - i} \tag{23}$$

$$q_i'(X) = q_i(X) + \delta_j \left(\frac{L_j(X)}{X - i}\right) \tag{24}$$

Applying KZG homomorphism, it follows that:

New Technique: Updating Proof $\pi_i = c(q_i)$ After Change (j, δ_j), $j \neq i$

Once again, we know $\pi_i' = c\left(q_i'\right)$, where:

$$q_i'(X) = \frac{\phi'(X) - v_i}{X - i} \tag{21}$$

$$= \frac{\left(\phi(X) + \delta_j L_j(X)\right) - v_i}{X - i} \tag{22}$$

$$= \frac{\phi(X) - v_i}{X - i} - \frac{\delta_j L_j(X)}{X - i} \tag{23}$$

$$q_i'(X) = q_i(X) + \delta_j \left(\frac{L_j(X)}{X - i}\right) \tag{24}$$

Applying KZG homomorphism, it follows that:

$$\pi_i' = c\left(q_i'\right) = c\left(q_i\right) \cdot c\left(\frac{L_j(X)}{X - i}\right)^{\delta_j} \tag{25}$$

New Technique: Updating Proof $\pi_i = c(q_i)$ After Change (j, δ_j), $j \neq i$

Once again, we know $\pi_i' = c\left(q_i'\right)$, where:

$$q_i'(X) = \frac{\phi'(X) - v_i}{X - i} \tag{21}$$

$$= \frac{\left(\phi(X) + \delta_j L_j(X)\right) - v_i}{X - i} \tag{22}$$

$$= \frac{\phi(X) - v_i}{X - i} - \frac{\delta_j L_j(X)}{X - i} \tag{23}$$

$$q_i'(X) = q_i(X) + \delta_j \left(\frac{L_j(X)}{X - i}\right) \tag{24}$$

Applying KZG homomorphism, it follows that:

$$\pi_i' = c\left(q_i'\right) = c\left(q_i\right) \cdot c\left(\frac{L_j(X)}{X - i}\right)^{\delta_j} \tag{25}$$

Problem: To update any π_i after a change to j, need $c\left(\frac{L_j(X)}{X-i}\right), \forall i \neq j$

New Technique: Updating Proof $\pi_i = c(q_i)$ After Change (j, δ_j), $j \neq i$

Once again, we know $\pi_i' = c\left(q_i'\right)$, where:

$$q_i'(X) = \frac{\phi'(X) - v_i}{X - i} \tag{21}$$

$$= \frac{\left(\phi(X) + \delta_j L_j(X)\right) - v_i}{X - i} \tag{22}$$

$$= \frac{\phi(X) - v_i}{X - i} - \frac{\delta_j L_j(X)}{X - i} \tag{23}$$

$$q_i'(X) = q_i(X) + \delta_j \left(\frac{L_j(X)}{X - i}\right) \tag{24}$$

Applying KZG homomorphism, it follows that:

$$\pi_i' = c\left(q_i'\right) = c\left(q_i\right) \cdot c\left(\frac{L_j(X)}{X - i}\right)^{\delta_j} \tag{25}$$

Problem: To update any π_i after a change to j, need $c\left(\frac{L_j(X)}{X-i}\right), \forall i \neq j \Rightarrow O(n)$-sized upk_j.

New Technique: Updating Proof $\pi_i = c(q_i)$ After Change (j, δ_j), $j \neq i$

Once again, we know $\pi_i' = c\left(q_i'\right)$, where:

$$q_i'(X) = \frac{\phi'(X) - v_i}{X - i} \tag{21}$$

$$= \frac{\left(\phi(X) + \delta_j L_j(X)\right) - v_i}{X - i} \tag{22}$$

$$= \frac{\phi(X) - v_i}{X - i} - \frac{\delta_j L_j(X)}{X - i} \tag{23}$$

$$q_i'(X) = q_i(X) + \delta_j \left(\frac{L_j(X)}{X - i}\right) \tag{24}$$

Applying KZG homomorphism, it follows that:

$$\pi_i' = c\left(q_i'\right) = c\left(q_i\right) \cdot c\left(\frac{L_j(X)}{X - i}\right)^{\delta_j} \tag{25}$$

Problem: To update any π_i after a change to j, need $c\left(\frac{L_j(X)}{X-i}\right), \forall i \neq j \Rightarrow O(n)$-sized upk_j.

Solution: Compute $c\left(\frac{L_j(X)}{X-i}\right)$ in $O(1)$ time from information in upk_i and upk_j.

New Technique: Compute $c\left(\frac{L_j(X)}{X-i}\right)$ in $O(1)$ Time

New Technique: Compute $c\left(\frac{L_j(X)}{X-i}\right)$ in $O(1)$ Time

Let $A(X) = \prod_{i\in[0,n)}(X - i)$.

New Technique: Compute $c\left(\frac{L_j(X)}{X-i}\right)$ in $O(1)$ Time

Let $A(X) = \prod_{i\in[0,n)}(X - i)$. Then:

$$\frac{L_j(X)}{X - i} = \frac{1}{A'(j)} \cdot \frac{A(X)}{(X - j)(X - i)} \tag{26}$$

Next, use **partial fraction decomposition** to rewrite:

New Technique: Compute $c\left(\frac{L_j(X)}{X-i}\right)$ in $O(1)$ Time

Let $A(X) = \prod_{i \in [0,n)}(X - i)$. Then:

$$\frac{L_j(X)}{X - i} = \frac{1}{A'(j)} \cdot \frac{A(X)}{(X - j)(X - i)} \tag{26}$$

Next, use **partial fraction decomposition** to rewrite:

$$\frac{A(X)}{(X - i)(X - j)} = \frac{1}{i - j} \cdot \frac{A(X)}{X - i} + \frac{1}{j - i} \cdot \frac{A(X)}{X - j} \tag{27}$$

New Technique: Compute $c\left(\frac{L_j(X)}{X-i}\right)$ in $O(1)$ Time

Let $A(X) = \prod_{i \in [0,n)} (X - i)$. Then:

$$\frac{L_j(X)}{X - i} = \frac{1}{A'(j)} \cdot \frac{A(X)}{(X - j)(X - i)} \tag{26}$$

Next, use **partial fraction decomposition** to rewrite:

$$\frac{A(X)}{(X - i)(X - j)} = \frac{1}{i - j} \cdot \frac{A(X)}{X - i} + \frac{1}{j - i} \cdot \frac{A(X)}{X - j} \tag{27}$$

Now, replacing Equation (27) into Equation (26):

$$\frac{L_j(X)}{X - i} = \frac{1}{A'(j)} \cdot \left(\frac{1}{i - j} \cdot \frac{A(X)}{X - i} + \frac{1}{j - i} \cdot \frac{A(X)}{X - j}\right) \tag{28}$$

New Technique: Compute $c\left(\frac{L_j(X)}{X-i}\right)$ in $O(1)$ Time

Let $A(X) = \prod_{i \in [0,n)}(X - i)$. Then:

$$\frac{L_j(X)}{X - i} = \frac{1}{A'(j)} \cdot \frac{A(X)}{(X - j)(X - i)} \tag{26}$$

Next, use **partial fraction decomposition** to rewrite:

$$\frac{A(X)}{(X - i)(X - j)} = \frac{1}{i - j} \cdot \frac{A(X)}{X - i} + \frac{1}{j - i} \cdot \frac{A(X)}{X - j} \tag{27}$$

Now, replacing Equation (27) into Equation (26):

$$\frac{L_j(X)}{X - i} = \frac{1}{A'(j)} \cdot \left(\frac{1}{i - j} \cdot \frac{A(X)}{X - i} + \frac{1}{j - i} \cdot \frac{A(X)}{X - j}\right) \tag{28}$$

As a result:

New Technique: Compute $c\left(\frac{L_j(X)}{X-i}\right)$ in $O(1)$ Time

Let $A(X) = \prod_{i\in[0,n)}(X - i)$. Then:

$$\frac{L_j(X)}{X - i} = \frac{1}{A'(j)} \cdot \frac{A(X)}{(X - j)(X - i)} \tag{26}$$

Next, use **partial fraction decomposition** to rewrite:

$$\frac{A(X)}{(X - i)(X - j)} = \frac{1}{i - j} \cdot \frac{A(X)}{X - i} + \frac{1}{j - i} \cdot \frac{A(X)}{X - j} \tag{27}$$

Now, replacing Equation (27) into Equation (26):

$$\frac{L_j(X)}{X - i} = \frac{1}{A'(j)} \cdot \left(\frac{1}{i - j} \cdot \frac{A(X)}{X - i} + \frac{1}{j - i} \cdot \frac{A(X)}{X - j}\right) \tag{28}$$

As a result:

$$c\left(\frac{L_j(X)}{X - i}\right) = \left(c\left(\frac{A(X)}{X - i}\right)^{\frac{1}{i-j}} \cdot c\left(\frac{A(X)}{X - j}\right)^{\frac{1}{j-i}}\right)^{\frac{1}{A'(j)}} \tag{29}$$

New Technique: Compute $c\left(\frac{L_j(X)}{X-i}\right)$ in $O(1)$ Time

Let $A(X) = \prod_{i\in[0,n)}(X - i)$. Then:

$$\frac{L_j(X)}{X - i} = \frac{1}{A'(j)} \cdot \frac{A(X)}{(X - j)(X - i)} \tag{26}$$

Next, use **partial fraction decomposition** to rewrite:

$$\frac{A(X)}{(X - i)(X - j)} = \frac{1}{i - j} \cdot \frac{A(X)}{X - i} + \frac{1}{j - i} \cdot \frac{A(X)}{X - j} \tag{27}$$

Now, replacing Equation (27) into Equation (26):

$$\frac{L_j(X)}{X - i} = \frac{1}{A'(j)} \cdot \left(\frac{1}{i - j} \cdot \frac{A(X)}{X - i} + \frac{1}{j - i} \cdot \frac{A(X)}{X - j}\right) \tag{28}$$

As a result:

$$c\left(\frac{L_j(X)}{X - i}\right) = \left(c\left(\frac{A(X)}{X - i}\right)^{\frac{1}{i-j}} \cdot c\left(\frac{A(X)}{X - j}\right)^{\frac{1}{j-i}}\right)^{\frac{1}{A'(j)}} \tag{29}$$

Thus, each upk_i must include $c\left(\frac{A(X)}{X-i}\right)$ and $A'(i)$.

New Technique: Compute $c\left(\frac{L_j(X)}{X-i}\right)$ in $O(1)$ Time

Let $A(X) = \prod_{i\in[0,n)}(X - i)$. Then:

$$\frac{L_j(X)}{X - i} = \frac{1}{A'(j)} \cdot \frac{A(X)}{(X - j)(X - i)} \tag{26}$$

Next, use **partial fraction decomposition** to rewrite:

$$\frac{A(X)}{(X - i)(X - j)} = \frac{1}{i - j} \cdot \frac{A(X)}{X - i} + \frac{1}{j - i} \cdot \frac{A(X)}{X - j} \tag{27}$$

Now, replacing Equation (27) into Equation (26):

$$\frac{L_j(X)}{X - i} = \frac{1}{A'(j)} \cdot \left(\frac{1}{i - j} \cdot \frac{A(X)}{X - i} + \frac{1}{j - i} \cdot \frac{A(X)}{X - j}\right) \tag{28}$$

As a result:

$$c\left(\frac{L_j(X)}{X - i}\right) = \left(c\left(\frac{A(X)}{X - i}\right)^{\frac{1}{i-j}} \cdot c\left(\frac{A(X)}{X - j}\right)^{\frac{1}{j-i}}\right)^{\frac{1}{A'(j)}} \tag{29}$$

Thus, each upk_i must include $c\left(\frac{A(X)}{X-i}\right)$ and $A'(i)$. *Can derive from g^{τ^i}'s!*

Aggregating Proofs

Aggregating Proofs

Given many $(\pi_i)_{i \in I}$, can aggregate into succinct **subvector proof** π_I.

Aggregating Proofs

Given many $(\pi_i)_{i \in I}$, can aggregate into succinct **subvector proof** π_I.

High-level ideas, thanks to Drake and Buterin:

Aggregating Proofs

Given many $(\pi_i)_{i \in I}$, can aggregate into succinct **subvector proof** π_I.

High-level ideas, thanks to Drake and Buterin:

- Each π_i is a commitment to quotient q_i of division $\frac{\phi(X)}{X-i}$

Aggregating Proofs

Given many $(\pi_i)_{i \in I}$, can aggregate into succinct **subvector proof** π_I.

High-level ideas, thanks to Drake and Buterin:

- Each π_i is a commitment to quotient q_i of division $\frac{\phi(X)}{X-i}$
- π_I is a commitment to quotient q_I of division $\frac{\phi(X)}{\prod_{i \in I}(X-i)}$ (see [KZG10])

Aggregating Proofs

Given many $(\pi_i)_{i \in I}$, can aggregate into succinct **subvector proof** π_I.

High-level ideas, thanks to Drake and Buterin:

- Each π_i is a commitment to quotient q_i of division $\frac{\phi(X)}{X-i}$
- π_I is a commitment to quotient q_I of division $\frac{\phi(X)}{\prod_{i \in I}(X-i)}$ (see [KZG10])
- Compute c_i's such that $\frac{1}{\prod_{i \in I}(X-i)} = \sum_{i \in I} c_i \frac{1}{X-i}$

Aggregating Proofs

Given many $(\pi_i)_{i \in I}$, can aggregate into succinct **subvector proof** π_I.

High-level ideas, thanks to Drake and Buterin:

- Each π_i is a commitment to quotient q_i of division $\frac{\phi(X)}{X-i}$
- π_I is a commitment to quotient q_I of division $\frac{\phi(X)}{\prod_{i \in I}(X-i)}$ (see [KZG10])
- Compute c_i's such that $\frac{1}{\prod_{i \in I}(X-i)} = \sum_{i \in I} c_i \frac{1}{X-i}$
- Then, $q_I(X) = \sum_{i \in I} c_i \cdot q_i(X)$

Aggregating Proofs

Given many $(\pi_i)_{i \in I}$, can aggregate into succinct **subvector proof** π_I.

High-level ideas, thanks to Drake and Buterin:

- Each π_i is a commitment to quotient q_i of division $\frac{\phi(X)}{X-i}$
- π_I is a commitment to quotient q_I of division $\frac{\phi(X)}{\prod_{i \in I}(X-i)}$ (see [KZG10])
- Compute c_i's such that $\frac{1}{\prod_{i \in I}(X-i)} = \sum_{i \in I} c_i \frac{1}{X-i}$
- Then, $q_I(X) = \sum_{i \in I} c_i \cdot q_i(X)$
- Thus, $\pi_I = \prod_{i \in I} \pi_i^{c_i}$

Outline

References i

Dan Boneh, Benedikt Bünz, and Ben Fisch.
Batching Techniques for Accumulators with Applications to IOPs and Stateless Blockchains.
In *CRYPTO'19*, 2019.

Jan Camenisch, Maria Dubovitskaya, Kristiyan Haralambiev, and Markulf Kohlweiss.
Composable and Modular Anonymous Credentials: Definitions and Practical Constructions.
In *ASIACRYPT'15*, 2015.

Dario Catalano and Dario Fiore.
Vector Commitments and Their Applications.
In *PKC'13*, 2013.

References ii

- Matteo Campanelli, Dario Fiore, Nicola Greco, Dimitris Kolonelos, and Luca Nizzardo.
  Vector Commitment Techniques and Applications to Verifiable Decentralized Storage, 2020.
  https://eprint.iacr.org/2020/149.
- Alexander Chepurnoy, Charalampos Papamanthou, and Yupeng Zhang.
  Edrax: A Cryptocurrency with Stateless Transaction Validation, 2018.
  https://eprint.iacr.org/2018/968.
- Dankrad Feist and Dmitry Khovratovich.
  Fast amortized Kate proofs, 2020.
  https://github.com/khovratovich/Kate.

References iii

Sergey Gorbunov, Leonid Reyzin, Hoeteck Wee, and Zhenfei Zhang.
Pointproofs: Aggregating Proofs for Multiple Vector Commitments, 2020.
https://eprint.iacr.org/2020/419.

Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg.
Constant-Size Commitments to Polynomials and Their Applications.
In *ASIACRYPT'10*, 2010.

Russell W. F. Lai and Giulio Malavolta.
Subvector Commitments with Application to Succinct Arguments.
In *CRYPTO'19*, 2019.

- Ralph C. Merkle.
  A Digital Signature Based on a Conventional Encryption Function.
  In Carl Pomerance, editor, *CRYPTO '87*, pages 369–378, Berlin, Heidelberg, 1988. Springer Berlin Heidelberg.
- Alin Tomescu, Robert Chen, Yiming Zheng, Ittai Abraham, Benny Pinkas, Guy Golan Gueta, and Srinivas Devadas.
  Towards Scalable Threshold Cryptosystems.
  In *IEEE S&P'20*, May 2020.
- Alin Tomescu.
  How to Keep a Secret and Share a Public Key (Using Polynomial Commitments).
  PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 2020.