Vector Commitments

and practical applications

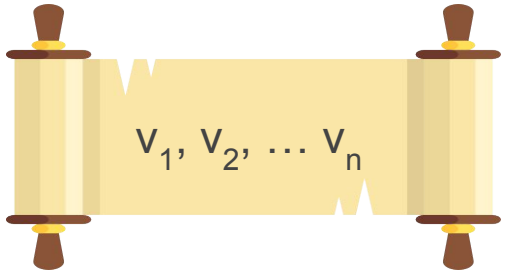
ia.cr/2022/705

with Matteo Campanelli, Carla Ràfols,
Alexandros Zacharakis, Arantxa Zapico

Anca Nitulescu

Protocol Labs

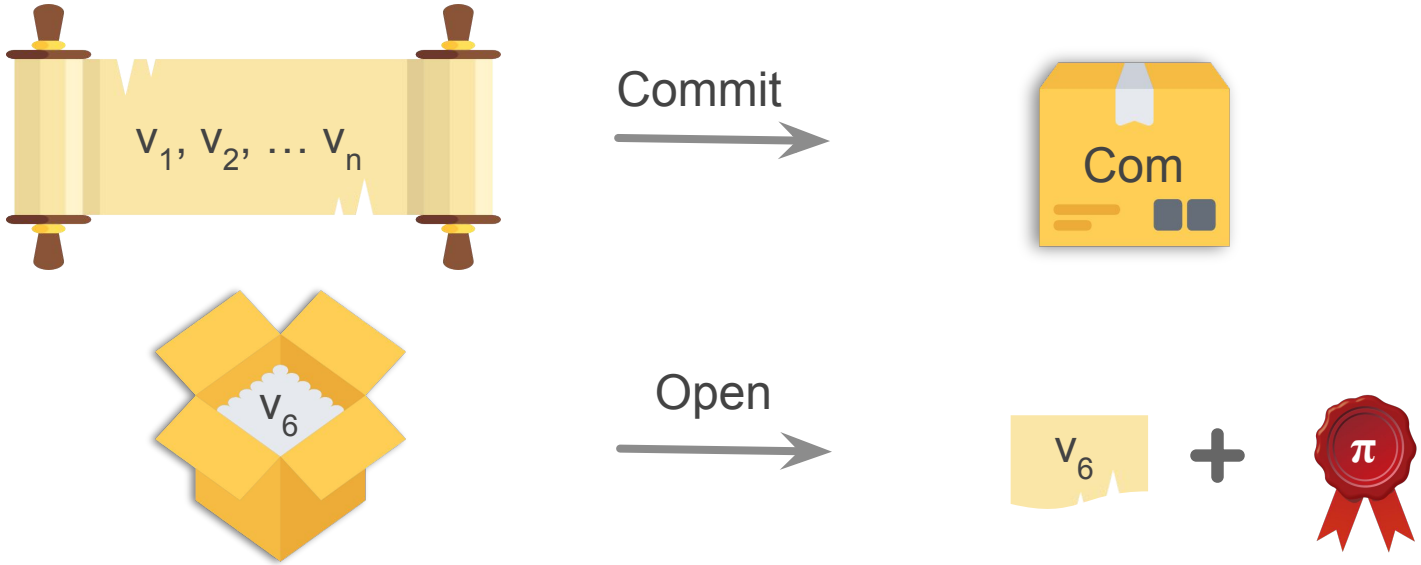
Vector Commitments



Commit

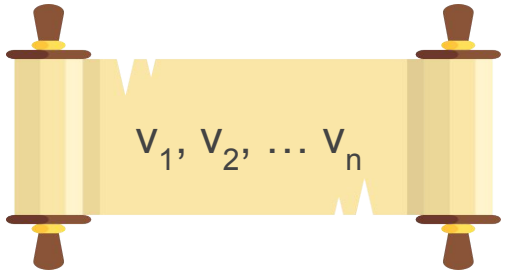
Vector Commitments and their Applications, by Dario Catalano and Dario Fiore

Vector Commitments

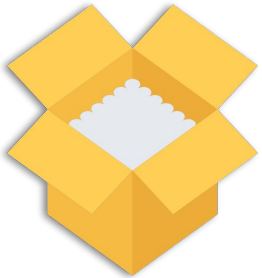


Vector Commitments and their Applications, by Dario Catalano and Dario Fiore

SubVector Commitment

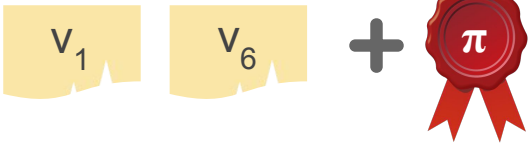


Commit

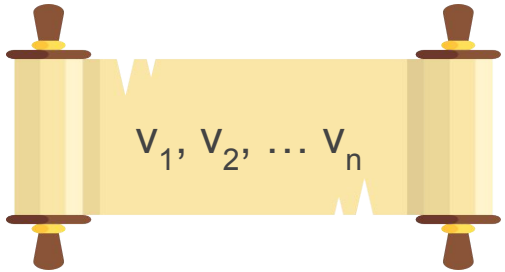


Open

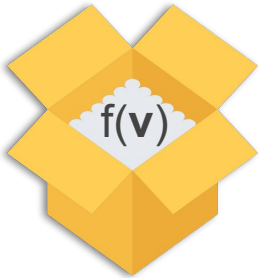
subvector



Functional VC



Commit

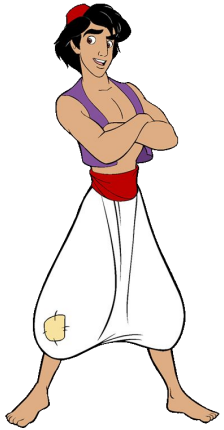


Open

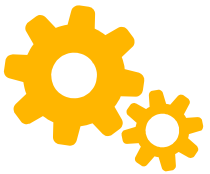
Motivation

Storage Delegation

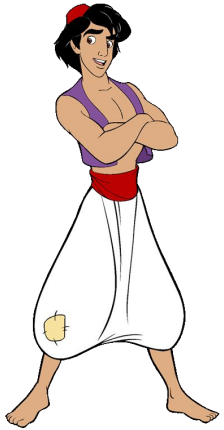
V_1 V_2 V_3 V_4 V_5



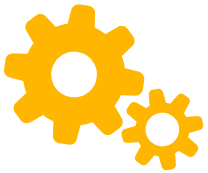
Storage Delegation



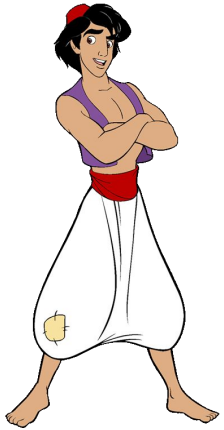
V_1 V_2 V_3 V_4 V_5



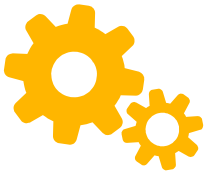
Storage Delegation



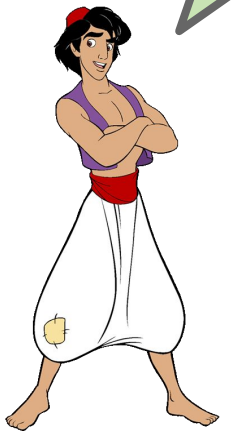
V_1 V_2 V_3 V_4 V_5



Storage Delegation



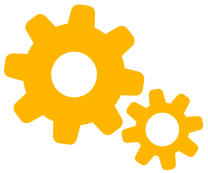
Query 3

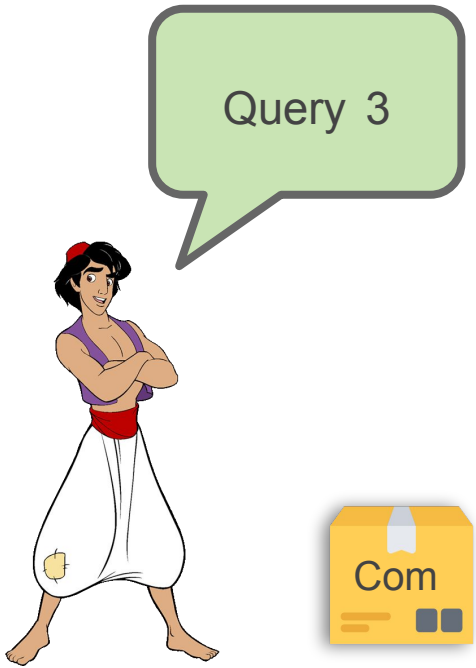


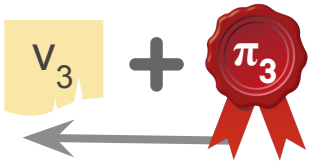
Com

V_1 V_2 V_3 V_4 V_5

Storage Delegation

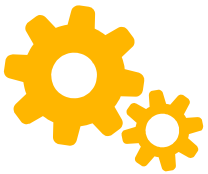


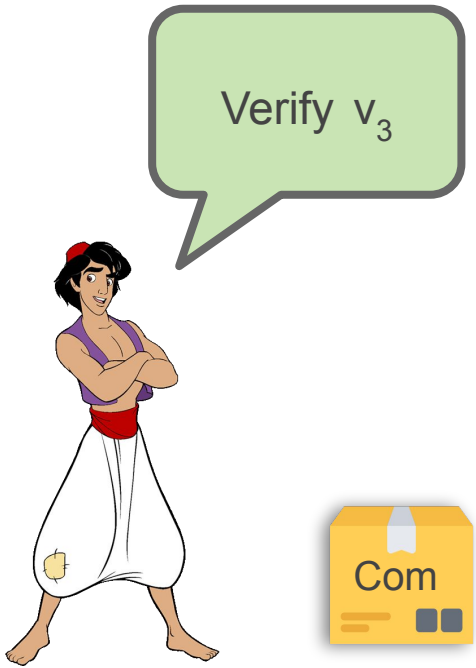


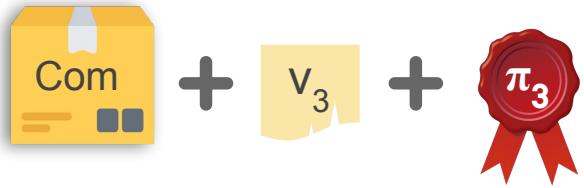


v_1 v_2 v_3 v_4 v_5

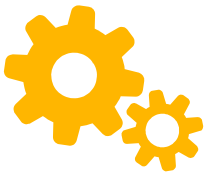
Storage Delegation

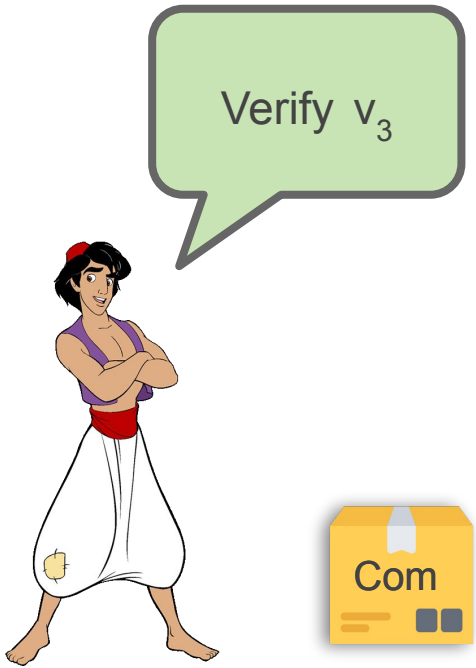




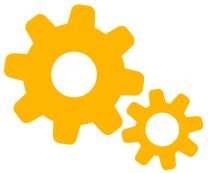


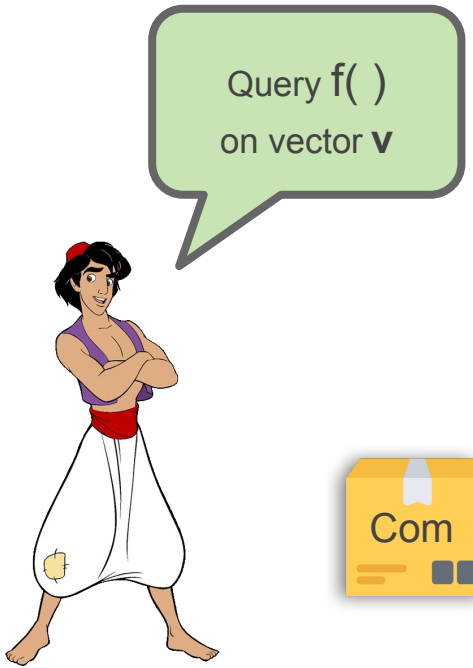
Storage Delegation



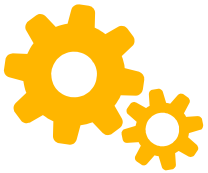


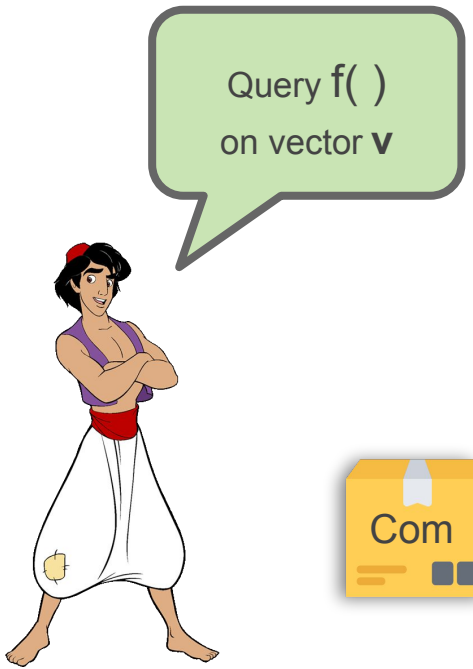
Storage Delegation

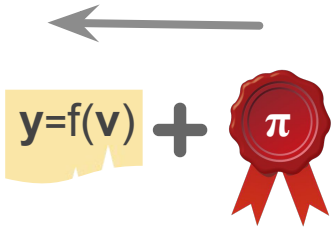




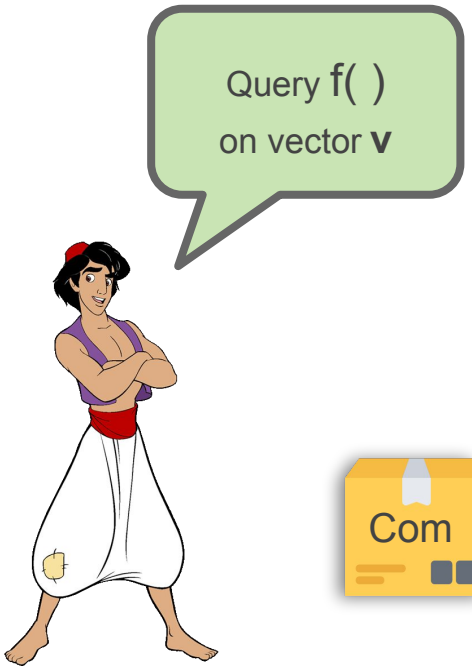
Storage Delegation

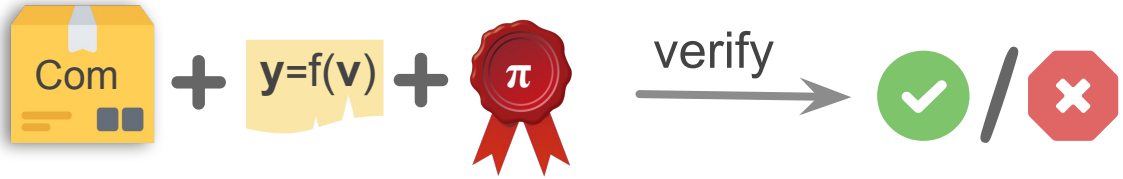






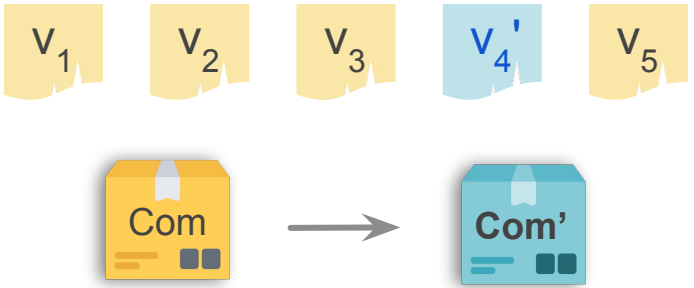
Storage Delegation





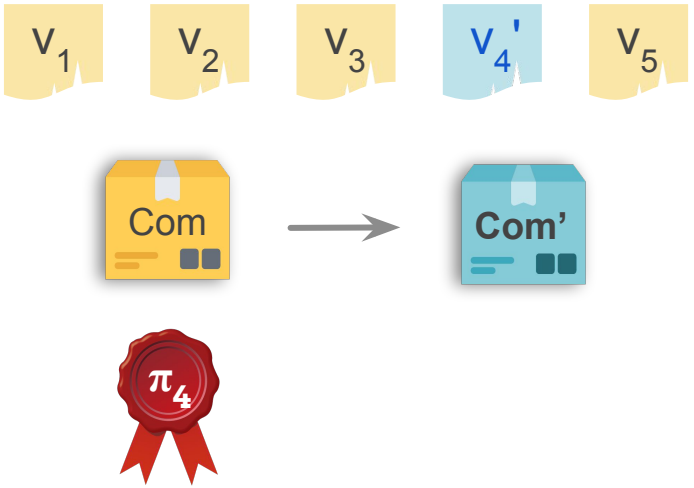
More Properties

Updates



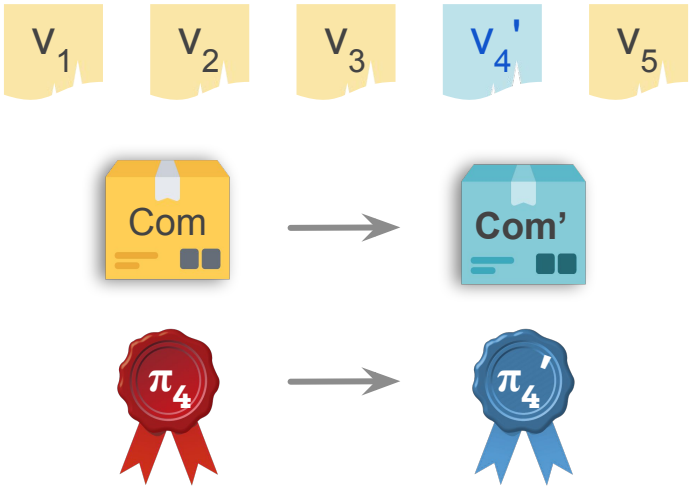
Aggregation

Updates



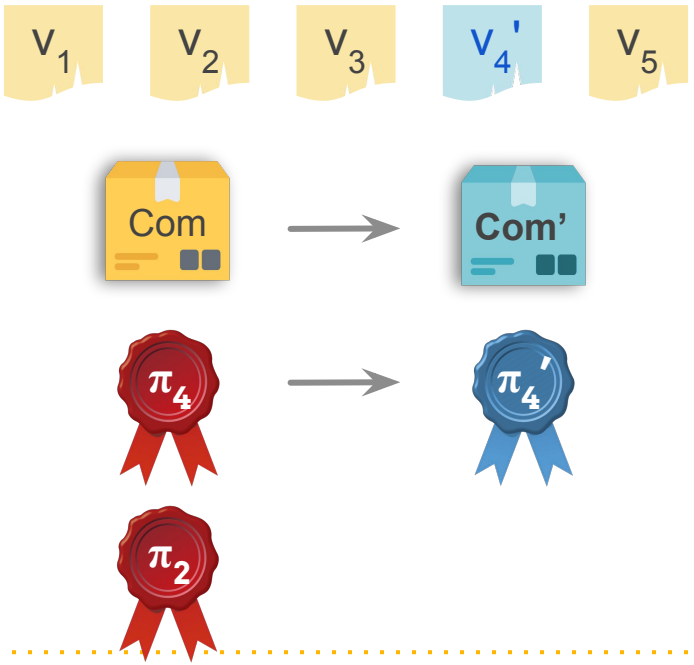
Aggregation

Updates



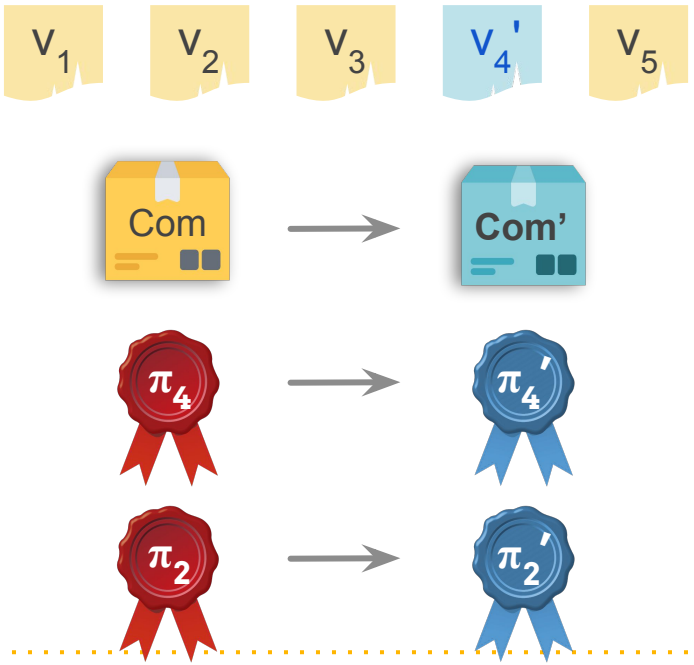
Aggregation

Updates



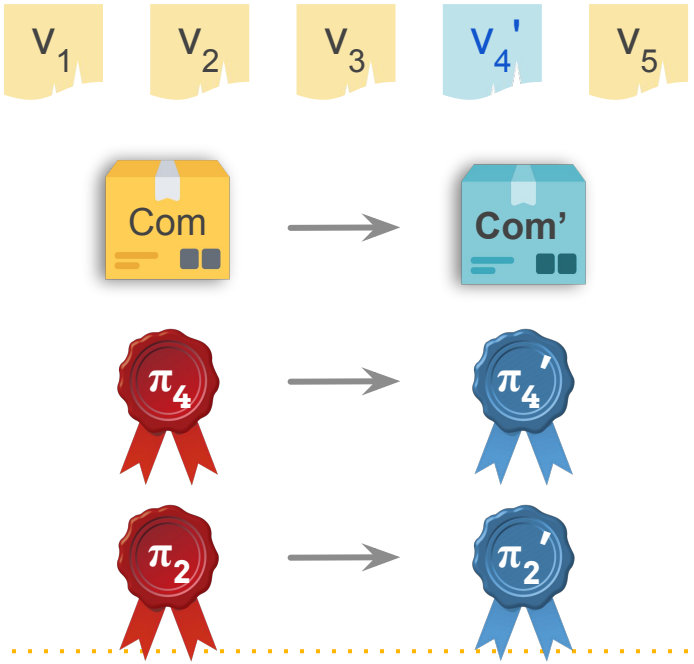
Aggregation

Updates

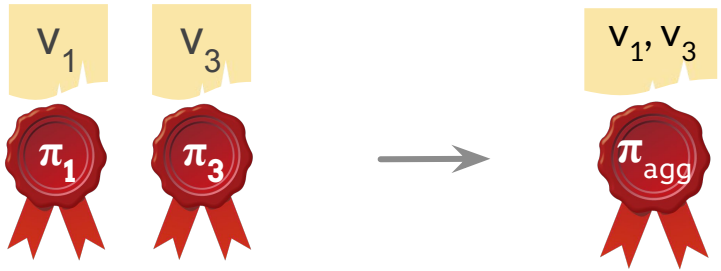


Aggregation

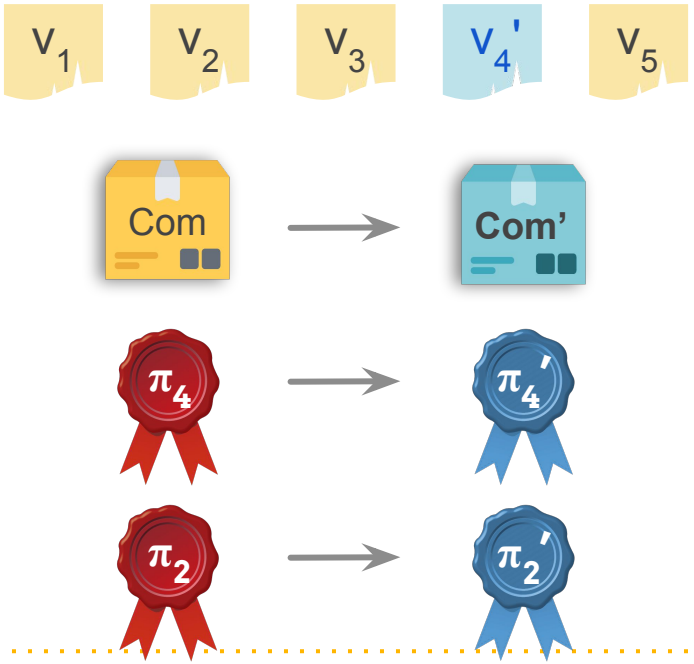
Updates



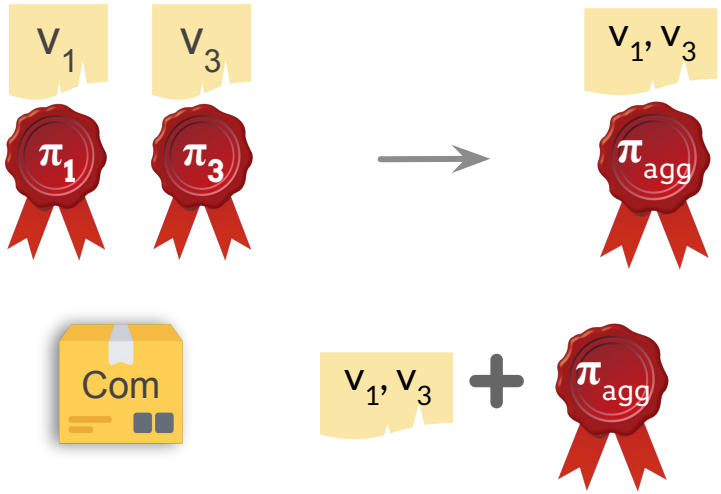
Aggregation



Updates

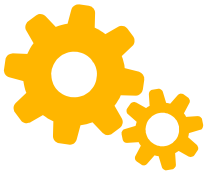


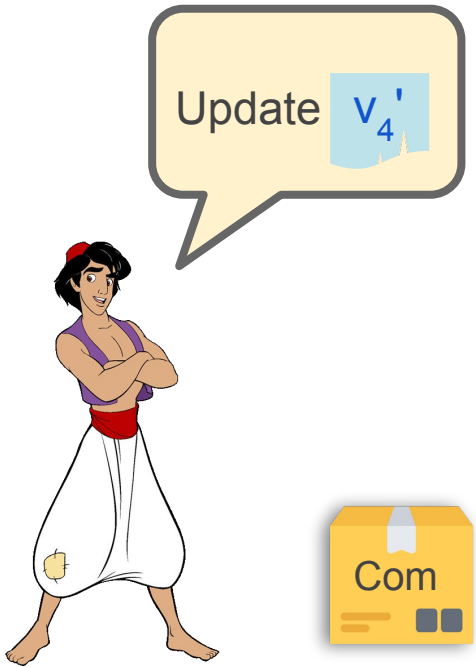
Aggregation



Does not need the entire vector **v**

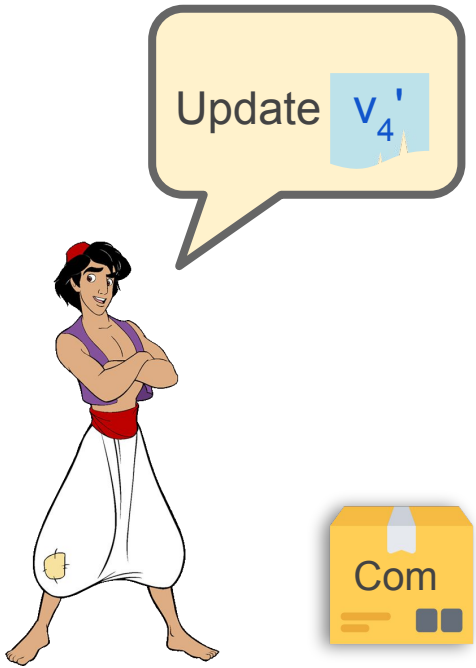
Updates for the vector

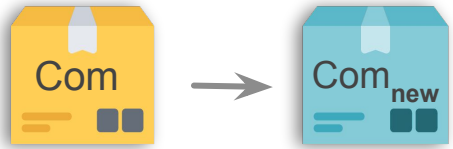




v_1 v_2 v_3 v_4 v_5

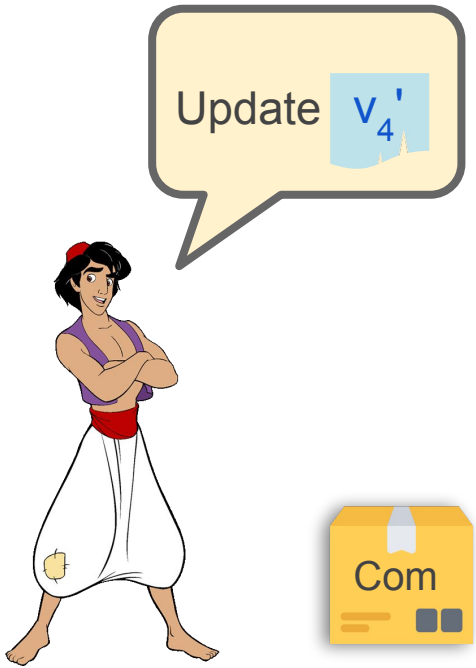
Updates for the vector

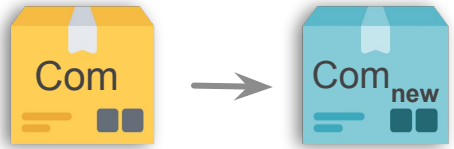


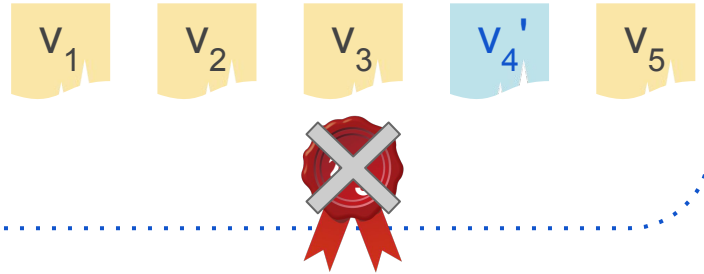


v_1 v_2 v_3 v_4 v_5

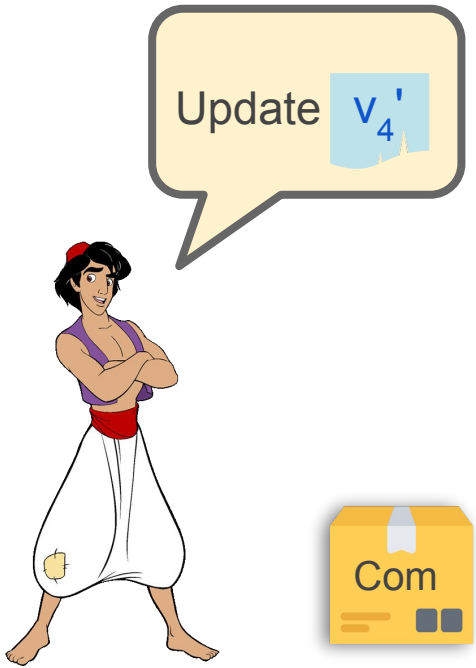
Updates for the vector

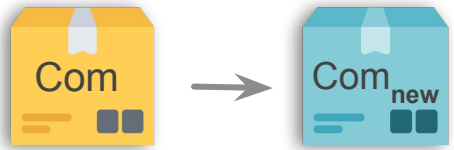






Updates for the vector

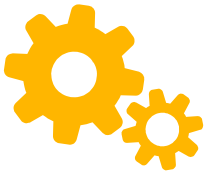


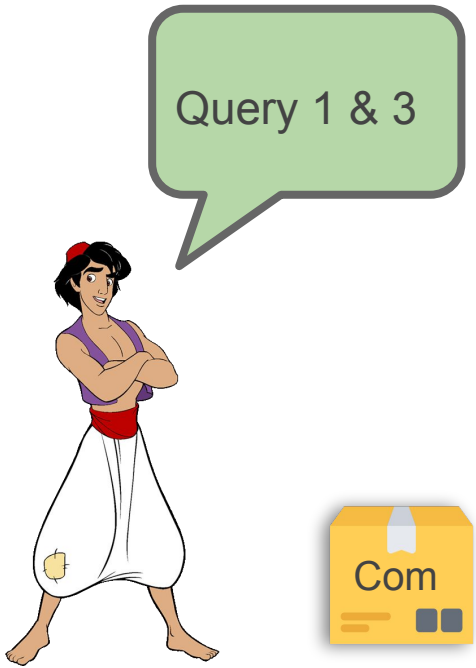


v_1 v_2 v_3 v_4' v_5

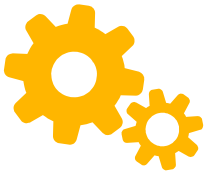
π_3'

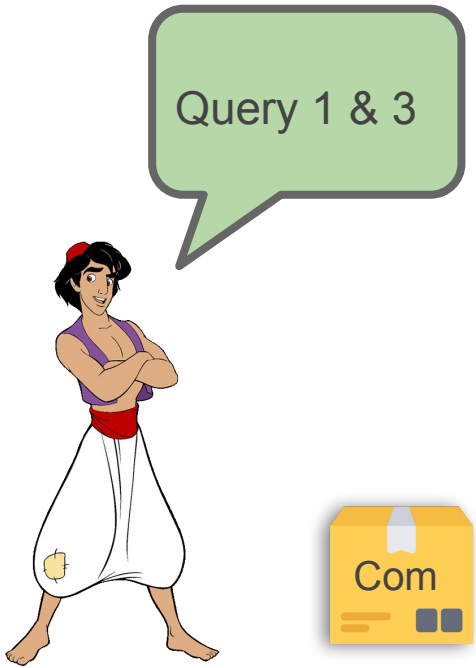
Proof Aggregation

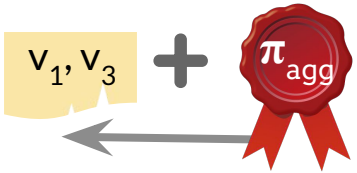


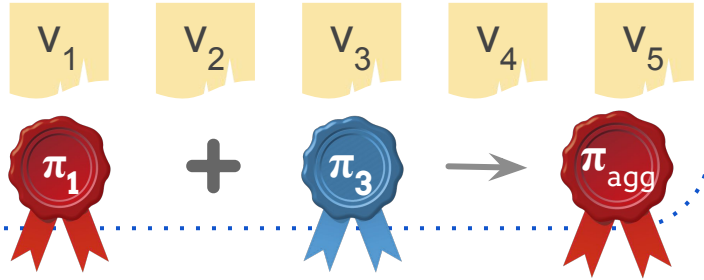


Proof Aggregation







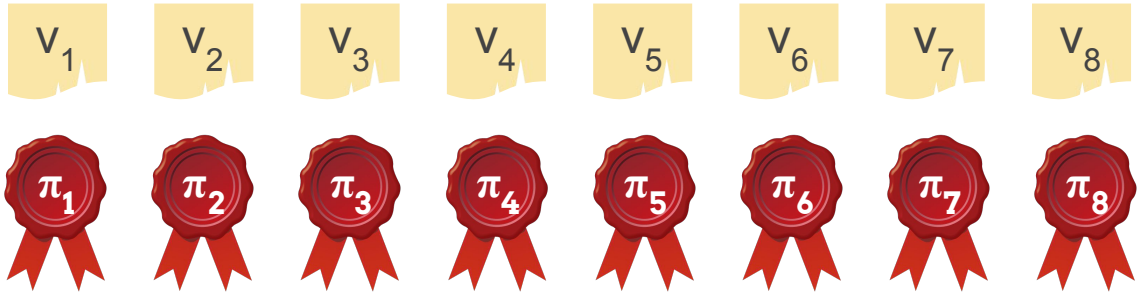


Drawbacks

Properties:

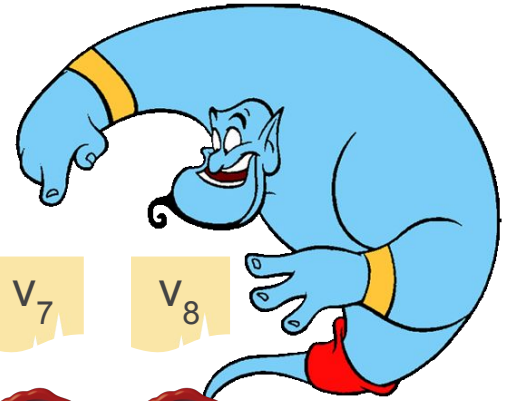
- Short proofs
- Fast verification
- Updates & Aggregation
- **Slow Prover**

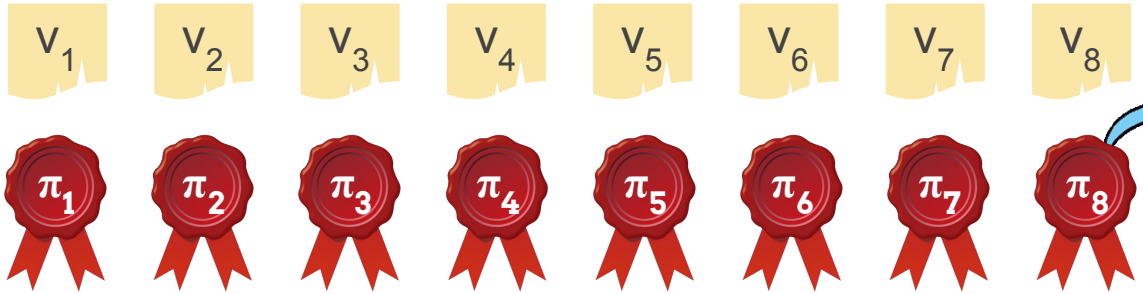
Trade Space for Time



Offline: Compute all proofs

Trade Space

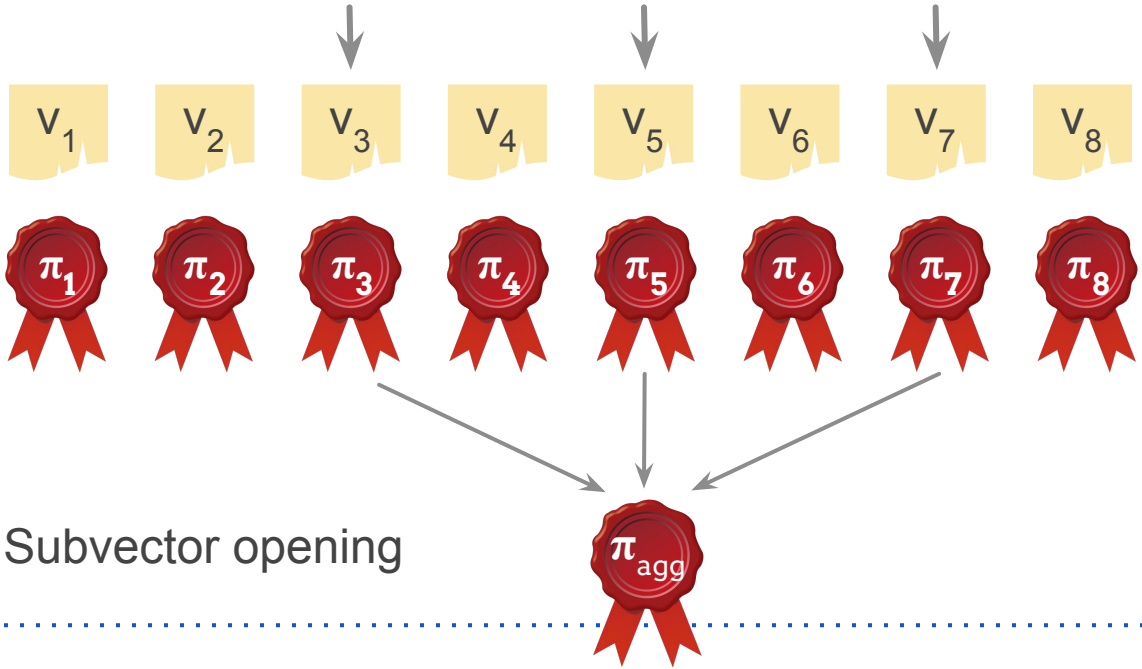




Offline: Compute all proofs

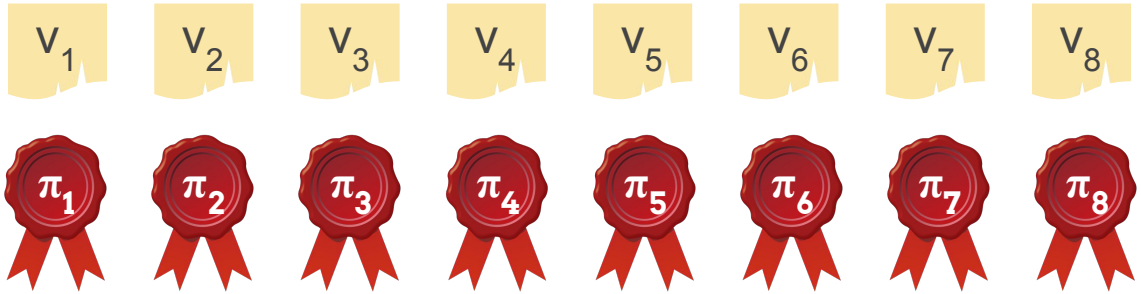
Online: Open by sending the stored proof

Trade Space

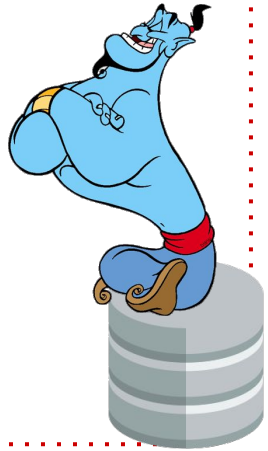


Subvector opening

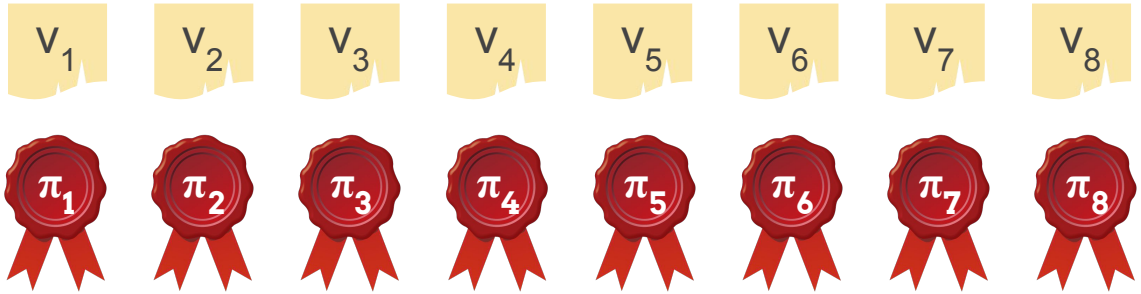
Space = expensive

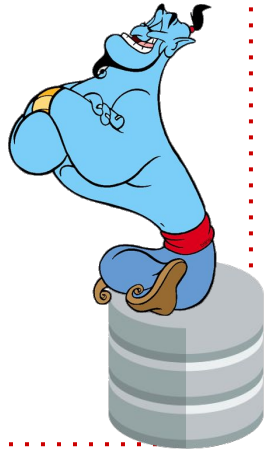


Computation: preprocessing with *small* amortized cost



Space = expensive

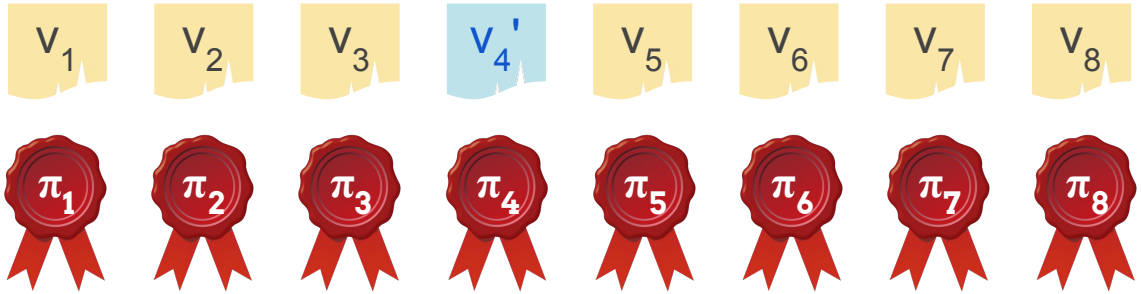


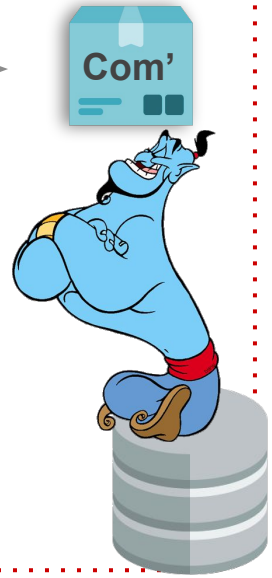


Computation: preprocessing with *small* amortized cost

Storage: doubles **Updates:** recompute all

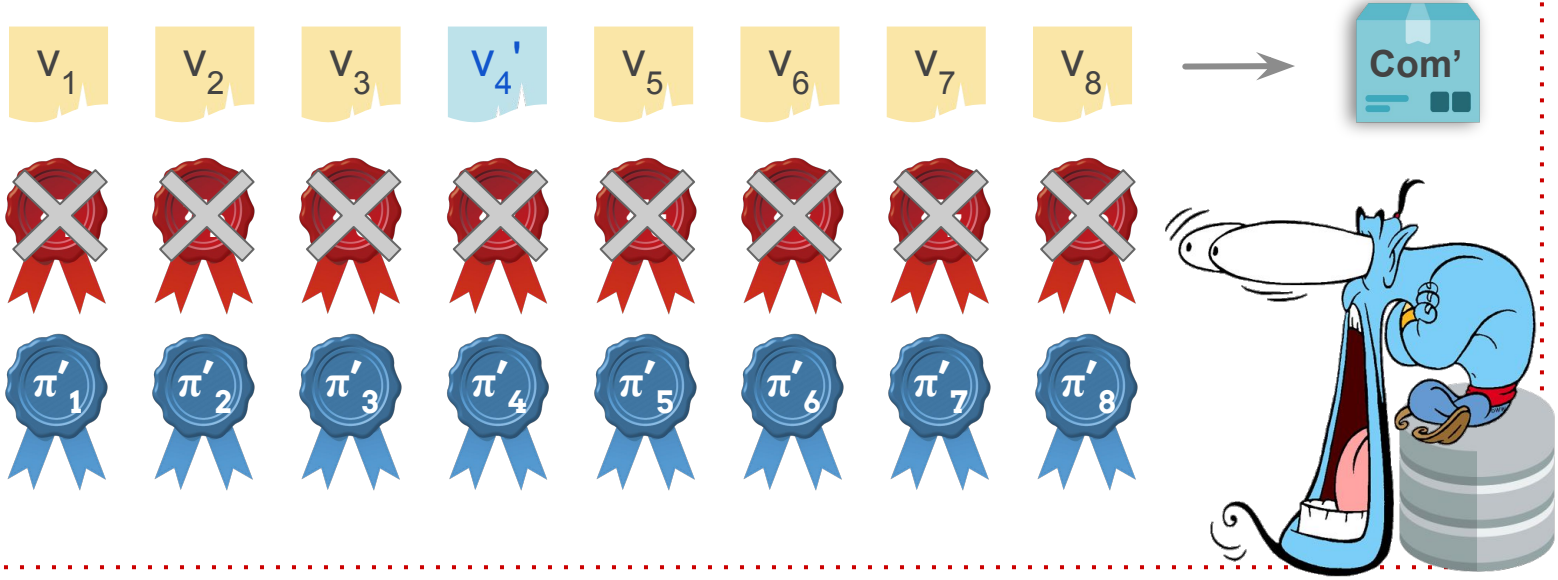
Updates = expensive





Updates = expensive

Updates = expensive



Maintainability

Update all: cost significantly smaller than recomputing all proofs

Space vs. Time

Space

- Sublinear, fast updates
- Ideally: flexibility - tune it

Time

- Sublinear(size of vector)
- Ideally: Sublinear(opening size)

Different solutions

Pairing-Based schemes

[KZG10], [CF13], [LM19],[GRWZ20], [TAB+20]

Advantages:

- constant-size openings
- constant-time verification
- **aggregatable**

Downsides:

- linear prover: no tradeoffs
- trusted setup
- linear parameters = bound on vector size
- updates: hard to maintain proofs

Tree-based schemes

[Merkle Trees, Hyperproofs, Verkle Trees]

Downsides:

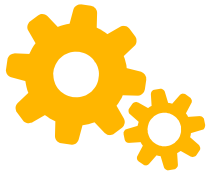
- log-size openings
- log-time verification
- non-aggregatable

Advantages:

- linear prover: **tradeoffs**
- transparent setup
- independent parameters
- maintainability for proofs

Our Constructions

Contributions



Linear-Map Vector Commitments

- New simple constructions for Inner-Product Vector Commitments
- Unbounded Aggregation & Updatability with static keys
- Special Subset Openings and Subset opening with Efficient Verifier

Maintainable Constructions

- Maintainable Tree-based construction from any LVC
- Trade-offs for the prover with flexible Space and Time savings
- All this with reusable Setup “powers of tau” from SNARKs

LVC Algorithms

KeyGen(λ, n) → (ck, vk)

Commit(ck, **v**) → (C, aux)

Open(ck, aux, f, **y**) → π $\mathcal{R}$: f(**v**) = **y**

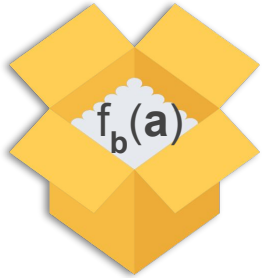
Verify(vk, C, f, **y**, π) → 0/1

Inner-Product VC

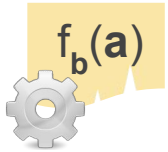
Inner Product of a committed vector with a public vector

$f_b(a) = \langle a, b \rangle$

$a, b \in \mathbb{F}^n$



Open



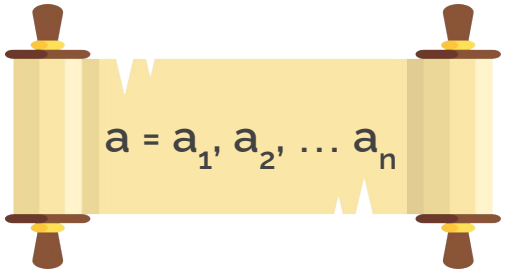
$f : \mathbb{F}^n \to \mathbb{F}$

Inner-Product VC

Encode the committed vector **a** as a polynomial

$f_b(a) = \langle a, b \rangle$

$a, b \in \mathbb{F}^n$



$A(X) = a_1 + a_2X + a_3X^2 + \ldots a_nX^{n-1}$

Monomial basis

$$M_i(X) = X^i$$

$a = a_1, a_2, \ldots a_n$

$A(X) = a_1 + a_2X + a_3X^2 + \ldots a_nX^{n-1}$

Lagrange basis

$$L_i(X) = \prod_{j \neq i} \frac{X - j}{i - j}$$

$a = a_1, a_2, \ldots a_n$

$A(X) = a_1L_1(X) + a_2L_2(X) + \ldots a_nL_n(X)$

$A(1) = a_1,\ A(2) = a_2,\ \ldots\ A(n) = a_n$

Background

Bilinear Groups

$$[a]_1 \in \mathbf{G}_1, [b]_2 \in \mathbf{G}_2$$

$$e : \mathbf{G}_1 \times \mathbf{G}_2 \rightarrow \mathbf{G}_T$$

$$e([a]_1, [b]_2) = [ab]_T$$

Inner Product VC

$f_b(\mathbf{a}) = \langle \mathbf{a}, \mathbf{b} \rangle$

KeyGen(λ, n) → ck: $[1]_1, [\tau]_1, [\tau^2]_1, \ldots [\tau^{n-1}]_1$

Inner Product VC

$f_b(\mathbf{a}) = \langle \mathbf{a}, \mathbf{b} \rangle$

KeyGen(λ, n) → ck: $[1]_1, [\tau]_1, [\tau^2]_1, \ldots [\tau^{n-1}]_1$

Commit(ck, **a**) → $C = [A(\tau)]_1$

$A(X) = a_1 + a_2X + a_3X^2 + \ldots a_nX^{n-1}$

Inner Product VC

$f_b(\mathbf{a}) = \langle \mathbf{a}, \mathbf{b} \rangle$

KeyGen(λ, n) $\rightarrow$ ck: $[1]_1, [\tau]_1, [\tau^2]_1, \ldots [\tau^{n-1}]_1$

Commit(ck, **a**) $\rightarrow$ $C = [A(\tau)]_1$

Open(ck, **a**, **b**, y=<**a**, **b**>) $\rightarrow$ π : $[H(\tau)]_1, [R(\tau)]_1, [\check{R}(\tau)]_1$

Inner Product VC

$f_b(\mathbf{a}) = \langle \mathbf{a}, \mathbf{b} \rangle$

KeyGen(λ, n) $\rightarrow$ ck: $[1]_1, [\tau]_1, [\tau^2]_1, \ldots [\tau^{n-1}]_1$

Commit(ck, **a**) $\rightarrow$ $C = [A(\tau)]_1$

Open(ck, **a**, **b**, y=<**a**, **b**>) $\rightarrow$ π: $[H(\tau)]_1, [R(\tau)]_1, [\check{R}(\tau)]_1$

$B(X) = b_n + b_{n-1}X + b_{n-2}X^2 + \ldots b_1X^{n-1}$

$A(X) = a_1 + a_2X + a_3X^2 + \ldots a_nX^{n-1}$

Inner Product VC

$f_b(\mathbf{a}) = \langle \mathbf{a}, \mathbf{b} \rangle$

KeyGen(λ, n) $\rightarrow$ ck: $[1]_1, [\tau]_1, [\tau^2]_1, \ldots [\tau^{n-1}]_1$

Commit(ck, **a**) $\rightarrow$ $C = [A(\tau)]_1$

Open(ck, **a**, **b**, y=<**a**, **b**>) $\rightarrow$ π: $[H(\tau)]_1, [R(\tau)]_1, [\check{R}(\tau)]_1$

$B(X) = b_n + b_{n-1}X + b_{n-2}X^2 + \ldots b_1X^{n-1}$

$A(X) = a_1 + a_2X + a_3X^2 + \ldots a_nX^{n-1}$

$A(X)\,B(X) =$

Inner Product VC

$f_b(\mathbf{a}) = \langle \mathbf{a}, \mathbf{b} \rangle$

KeyGen(λ, n) $\rightarrow$ ck: $[1]_1, [\tau]_1, [\tau^2]_1, \ldots [\tau^{n-1}]_1$

Commit(ck, **a**) $\rightarrow$ $C = [A(\tau)]_1$

Open(ck, **a**, **b**, y=<**a**, **b**>) $\rightarrow$ π : $[H(\tau)]_1, [R(\tau)]_1, [\check{R}(\tau)]_1$

$B(X) = b_n + b_{n-1}X + b_{n-2}X^2 + \ldots b_1X^{n-1}$

$A(X) = a_1 + a_2X + a_3X^2 + \ldots a_nX^{n-1}$

$A(X)\,B(X) = \langle \mathbf{a}, \mathbf{b} \rangle X^{n-1}$

Inner Product VC

$f_b(\mathbf{a}) = \langle \mathbf{a}, \mathbf{b} \rangle$

KeyGen(λ, n) $\rightarrow$ ck: $[1]_1, [\tau]_1, [\tau^2]_1, \ldots [\tau^{n-1}]_1$

Commit(ck, **a**) $\rightarrow$ $C = [A(\tau)]_1$

Open(ck, **a**, **b**, y=<**a**, **b**>) $\rightarrow$ π: $[H(\tau)]_1, [R(\tau)]_1, [\check{R}(\tau)]_1$

$B(X) = b_n + b_{n-1}X + b_{n-2}X^2 + \ldots b_1X^{n-1}$

$A(X) = a_1 + a_2X + a_3X^2 + \ldots a_nX^{n-1}$

$A(X)\,B(X) = \langle \mathbf{a}, \mathbf{b} \rangle X^{n-1} + H(X)\,X^n$

Inner Product VC

$f_b(\mathbf{a}) = \langle \mathbf{a}, \mathbf{b} \rangle$

KeyGen(λ, n) $\rightarrow$ ck: $[1]_1, [\tau]_1, [\tau^2]_1, \ldots [\tau^{n-1}]_1$

Commit(ck, **a**) $\rightarrow$ $C = [A(\tau)]_1$

Open(ck, **a**, **b**, y=$\langle \mathbf{a}, \mathbf{b} \rangle$) $\rightarrow$ π: $[H(\tau)]_1, [R(\tau)]_1, [\check{R}(\tau)]_1$

$B(X) = b_n + b_{n-1}X + b_{n-2}X^2 + \ldots b_1 X^{n-1}$

$A(X) = a_1 + a_2 X + a_3 X^2 + \ldots a_n X^{n-1}$

$A(X)\,B(X) = \langle \mathbf{a}, \mathbf{b} \rangle X^{n-1} + H(X)\,X^n + R(X)$

Inner Product VC

$f_b(\mathbf{a}) = \langle \mathbf{a}, \mathbf{b} \rangle$

KeyGen(λ, n) $\rightarrow$ ck: $[1]_1, [\tau]_1, [\tau^2]_1, \ldots [\tau^{n-1}]_1$

Commit(ck, **a**) $\rightarrow$ $C = [A(\tau)]_1$

Open(ck, **a**, **b**, y=<**a**, **b**>) $\rightarrow$ π: $[H(\tau)]_1, [R(\tau)]_1, [\check{R}(\tau)]_1$

$B(X) = b_n + b_{n-1}X + b_{n-2}X^2 + \ldots b_1X^{n-1}$

$A(X) = a_1 + a_2X + a_3X^2 + \ldots a_nX^{n-1}$

$A(X)\,B(X) = \langle \mathbf{a}, \mathbf{b} \rangle X^{n-1} + H(X)\,X^n + R(X)$

$\check{R}(X) = XR(X)$

Inner Product VC

$f_b(\mathbf{a}) = \langle \mathbf{a}, \mathbf{b} \rangle$

KeyGen(λ, n) $\rightarrow$ ck: $[1]_1, [\tau]_1, [\tau^2]_1, \ldots [\tau^{n-1}]_1$

Commit(ck, **a**) $\rightarrow$ $C = [A(\tau)]_1$

Open(ck, **a**, **b**, y=<**a**, **b**>) $\rightarrow$ π : $[H(\tau)]_1, [R(\tau)]_1, [\check{R}(\tau)]_1$

Verify(vk, C, **b**, y, π): $A(X)\,B(X) = y\,X^{n-1} + H(X)\,X^n + R(X)$ $\quad \check{R}(X) = XR(X)$

Inner Product VC

$f_b(\mathbf{a}) = \langle \mathbf{a}, \mathbf{b} \rangle$

KeyGen(λ, n) $\rightarrow$ ck: $[1]_1, [\tau]_1, [\tau^2]_1, \ldots [\tau^{n-1}]_1$

Commit(ck, **a**) $\rightarrow$ $C = [A(\tau)]_1$

Open(ck, **a**, **b**, y=<**a**, **b**>) $\rightarrow$ π : $[H(\tau)]_1, [R(\tau)]_1, [\check{R}(\tau)]_1$

Verify(vk, C, **b**, y, π): $e([A(\tau)]_1, [B(\tau)]_2) = e([y\,\tau^{n-1}]_1, [1]_2)\, e([H(\tau)]_1, [\tau^n]_2)\, e([R(\tau)]_1, [1]_2)$

Inner Product VC

$f_b(\mathbf{a}) = \langle \mathbf{a}, \mathbf{b} \rangle$

KeyGen(λ, n) $\rightarrow$ ck: $[1]_1, [\tau]_1, [\tau^2]_1, \ldots [\tau^{n-1}]_1$ vk: $[1]_2, [\tau]_2, \ldots [\tau^n]_2$

Commit(ck, **a**) $\rightarrow$ $C = [A(\tau)]_1$

Open(ck, **a**, **b**, y=<**a**, **b**>) $\rightarrow$ π : $[H(\tau)]_1, [R(\tau)]_1, [\check{R}(\tau)]_1$

Verify(vk, C, **b**, y, π): $e([A(\tau)]_1, [B(\tau)]_2) = e([y\,\tau^{n-1}]_1, [1]_2)\, e([H(\tau)]_1, [\tau^n]_2)\, e([R(\tau)]_1, [1]_2)$

Inner Product VC

$f_b(\mathbf{a}) = \langle \mathbf{a}, \mathbf{b} \rangle$

KeyGen(λ, n) $\rightarrow$ ck: $[1]_1, [\tau]_1, [\tau^2]_1, \ldots [\tau^{n-1}]_1$ vk: $[1]_2, [\tau]_2, \ldots [\tau^n]_2$

Commit(ck, **a**) $\rightarrow$ $C = [A(\tau)]_1$

Open(ck, **a**, **b**, y=<**a**, **b**>) $\rightarrow$ π : $[H(\tau)]_1, [R(\tau)]_1, [\check{R}(\tau)]_1$

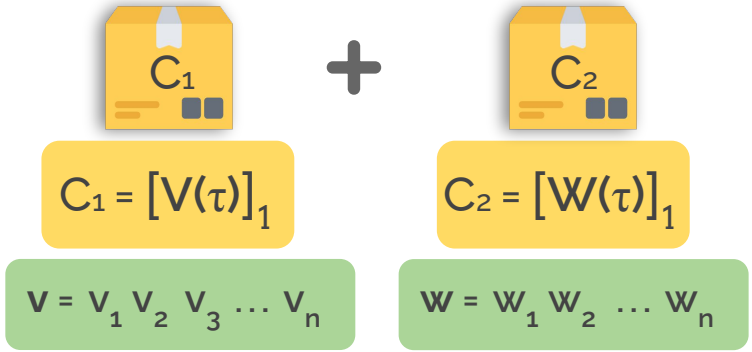
Verify(vk, C, **b**, y, π): $e([\check{R}(\tau)]_1, [1]_2) = e([R(\tau)]_1, [\tau]_2)$

$\check{R}(X) = XR(X)$

Properties of LVC

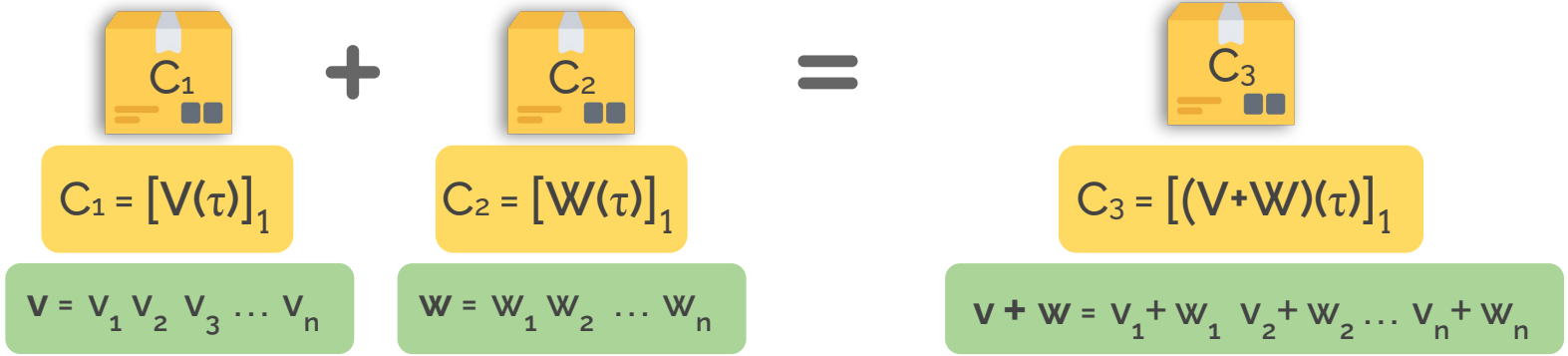
Homomorphism

Commitments



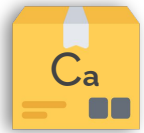
Homomorphism

Commitments



Homomorphism

Openings



+

=

$y_1 = \langle a, b_1 \rangle$ $y_2 = \langle a, b_2 \rangle$ $y_3 = \langle a, b_1 + b_2 \rangle$

Homomorphism

Openings

$y_1 = \langle a, b_1 \rangle$ $y_2 = \langle a, b_2 \rangle$ $y_3 = \langle a, b_1 + b_2 \rangle$

Proofs

$z_1 = \langle a_1, b \rangle$ $z_2 = \langle a_2, b \rangle$ $z_3 = \langle a_1 + a_2, b \rangle$

Updatability for Com

$a' = a_1\ a_2 \ldots a_i + \delta \ldots a_n$

upk: $[1]_1, [\tau]_1, [\tau^2]_1, \ldots [\tau^{n-1}]_1$

$C' = [A(\tau)]_1 + [\delta\tau^i]_1$

Updatability for Proof

$a' = a_1\ a_2 \dots a_i + \delta \dots a_n$

upk: $[1]_1, [\tau]_1, [\tau^2]_1, \dots [\tau^{n-1}]_1$

$C' = [A(\tau)]_1 + [\delta\tau^i]_1$

$y' = \langle \mathbf{a} + \delta \mathbf{e}_i, \mathbf{b} \rangle = y_1 = \langle \mathbf{a}, \mathbf{b} \rangle + \delta \langle \mathbf{e}_i, \mathbf{b} \rangle$

Updatability for Proof

$\mathbf{a'} = a_1\ a_2 \ldots a_i + \delta \ldots a_n$

upk: $[1]_1, [\tau]_1, [\tau^2]_1, \ldots [\tau^{n-1}]_1$

$C' = [A(\tau)]_1 + [\delta\tau^i]_1$

$A(X)\,B(X) = \langle \mathbf{a}, \mathbf{b} \rangle X^{n-1} + H(X)\,X^n + R(X)$

$=$

$+$

$y' = \langle \mathbf{a} + \delta \mathbf{e}_i, \mathbf{b} \rangle = y_1 = \langle \mathbf{a}, \mathbf{b} \rangle + \delta \langle \mathbf{e}_i, \mathbf{b} \rangle$

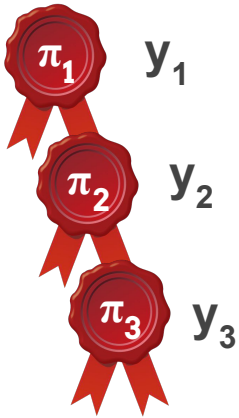
$\pi_{ij}: \langle \mathbf{e}_i, \mathbf{e}_j \rangle = 0$

$H(X)/R(X) = X^{i+n-j}$

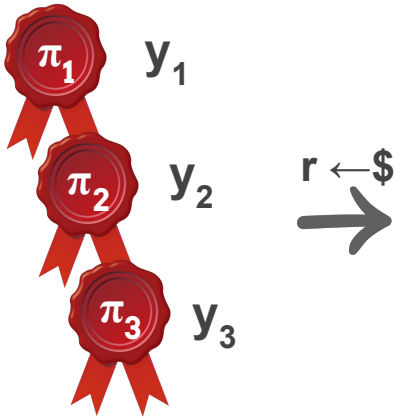
$\pi_{ii}: \langle \mathbf{e}_i, \mathbf{e}_i \rangle = 1$

$H(X)/R(X) = 0$

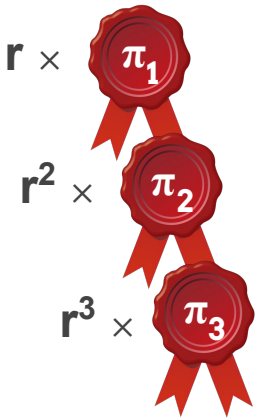
Aggregation

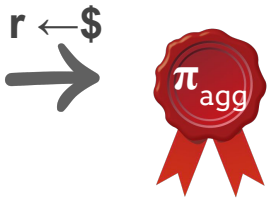


Aggregation



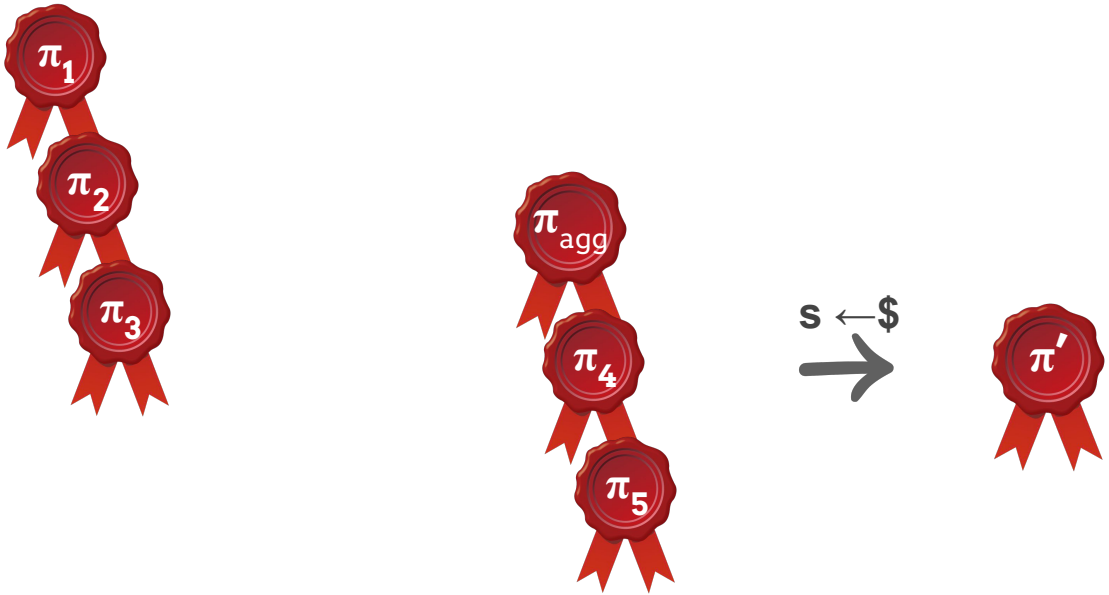
Aggregation



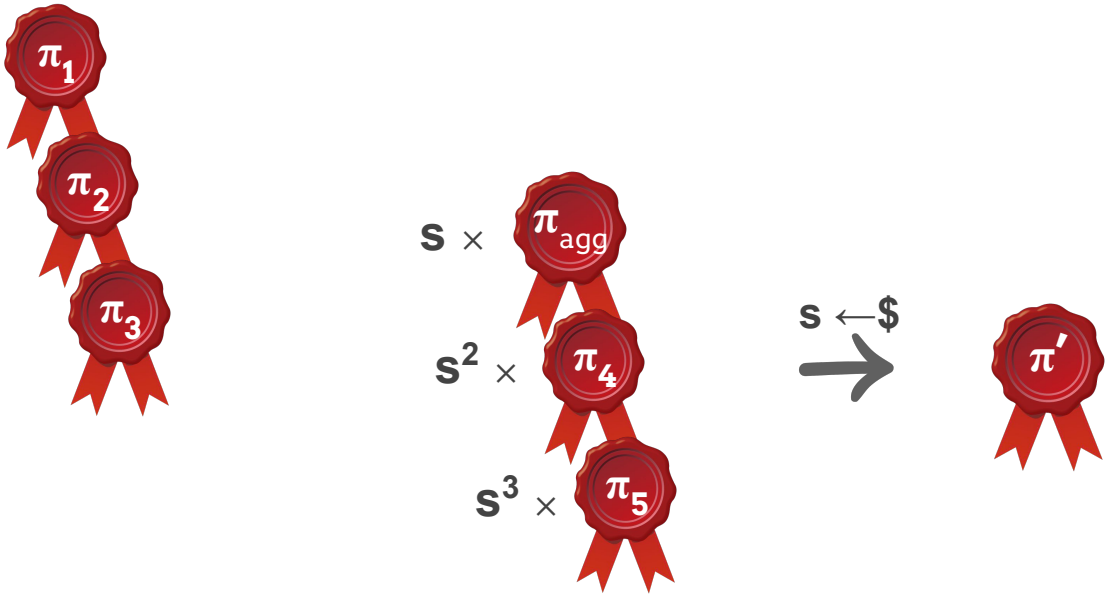


$$\pi_{agg} = \sum r^i \pi_i$$

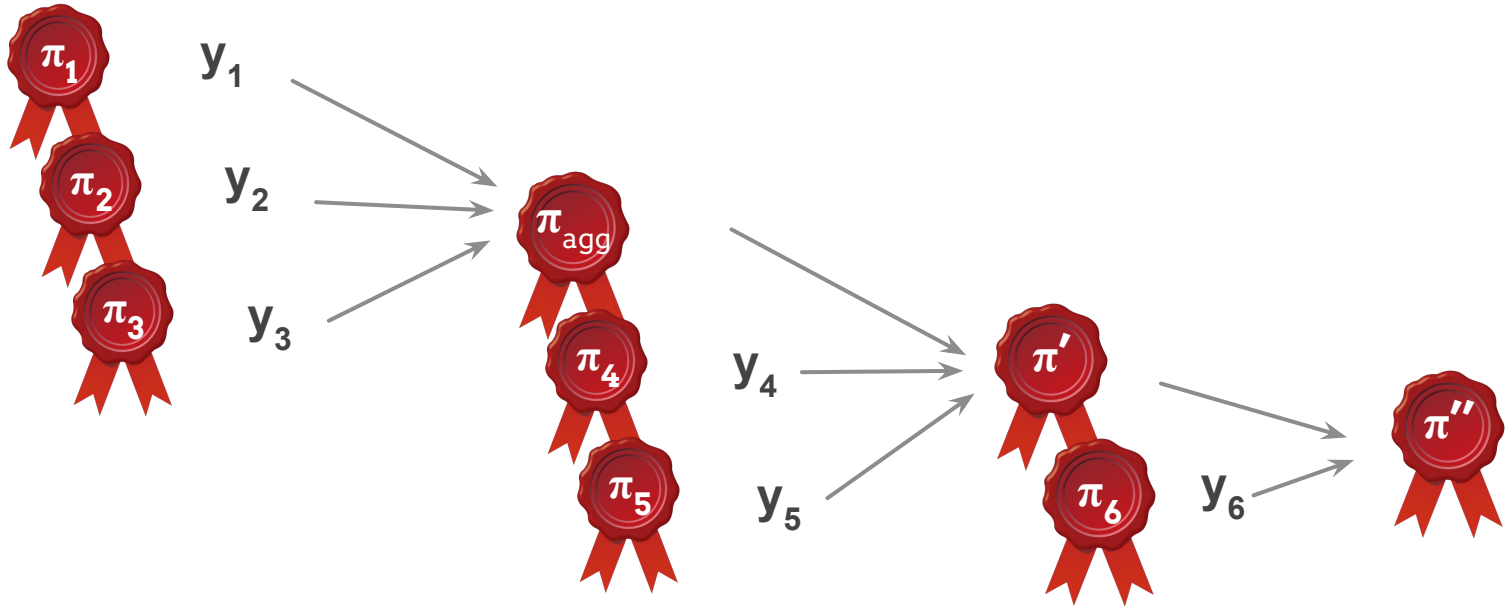
Aggregation



Aggregation



Aggregation



Linear-Map VC from IP

$f : \mathbb{F}^n \to \mathbb{F}^\ell$

KeyGen(λ, n) ⟶ ck: $[1]_1, [\tau]_1, [\tau^2]_1, \ldots [\tau^{n-1}]_1$ vk: $[1]_2, [\tau]_2, \ldots [\tau^n]_2$

Commit(ck, **a**) ⟶ $C = [A(\tau)]_1$

Open(ck, **a**, f: **B**=($\mathbf{b}_i$), **y**) ⟶ $\pi_1, \pi_2, \ldots \pi_\ell$ ⟶ π_{agg}

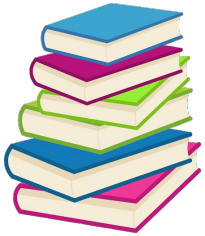
Verify(vk, C, f , **y**, π_{agg})

LVC comparison

$f : \mathbb{F}^n \to \mathbb{F}^\ell$

Schemes		|ck|	|vk|	|upk|	Updates	Aggregation	SVC	LVC
[LRY16]		n	n	—	✗	✗	✗	✓
[LM19]	SVC	n^2	n	n	key	✗	✓	✗
	LVC	$n\ell$	n	n	key	✗	✓	✓
Pointproofs		n	n	n	key	cross & one-hop	✓	✗
[CFG+20]		1	1	1	hint	same & incremental	✓	✗
Our Monomial		**n**	**n**	**n**	**key**	**cross & unbounded**	✓	✓
Our Lagrange		**n**	**ℓ**	**n**	**key**	**cross & unbounded**	✓	✓

References



[CF13] **Vector Commitments and their Applications**, by D. Catalano, D. Fiore, in PKC'13, https://eprint.iacr.org/2011/495

[CFG+20] **Vector Commitment Techniques and Applications to Verifiable Decentralized Storage**, by M. Campanelli, D. Fiore, N. Greco, D. Kolonelos, L. Nizzardo, in ASIACRYPT'20, https://eprint.iacr.org/2020/149

[GRWZ20] **Pointproofs: Aggregating Proofs for Multiple Vector Commitments**, by S. Gorbunov, L. Reyzin, H. Wee, Z. Zhang, in CCS'20, https://eprint.iacr.org/2020/419

[KZG10] **Constant-Size Commitments to Polynomials and Their Applications**, by A. Kate, G. Zaverucha, I. Goldberg, in ASIACRYPT'10,

[LRY10] **Functional commitment schemes: From polynomial commitments to pairing-based accumulators from simple assumptions**, by B. Libert, S. Ramanna and M. Yung, in ICALP'16

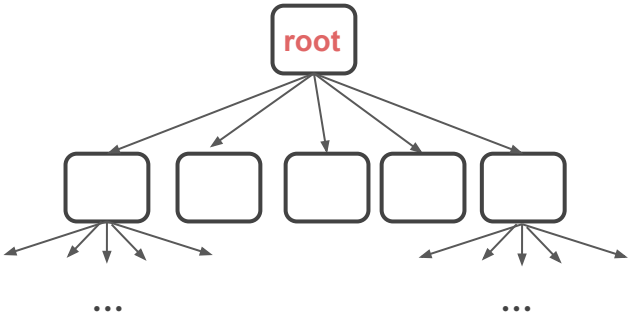
[LM19] **Subvector Commitments with Application to Succinct Arguments**, by R. W. F. Lai, G. Malavolta, in CRYPTO'19, https://eprint.iacr.org/2018/705

Maintainable Trees

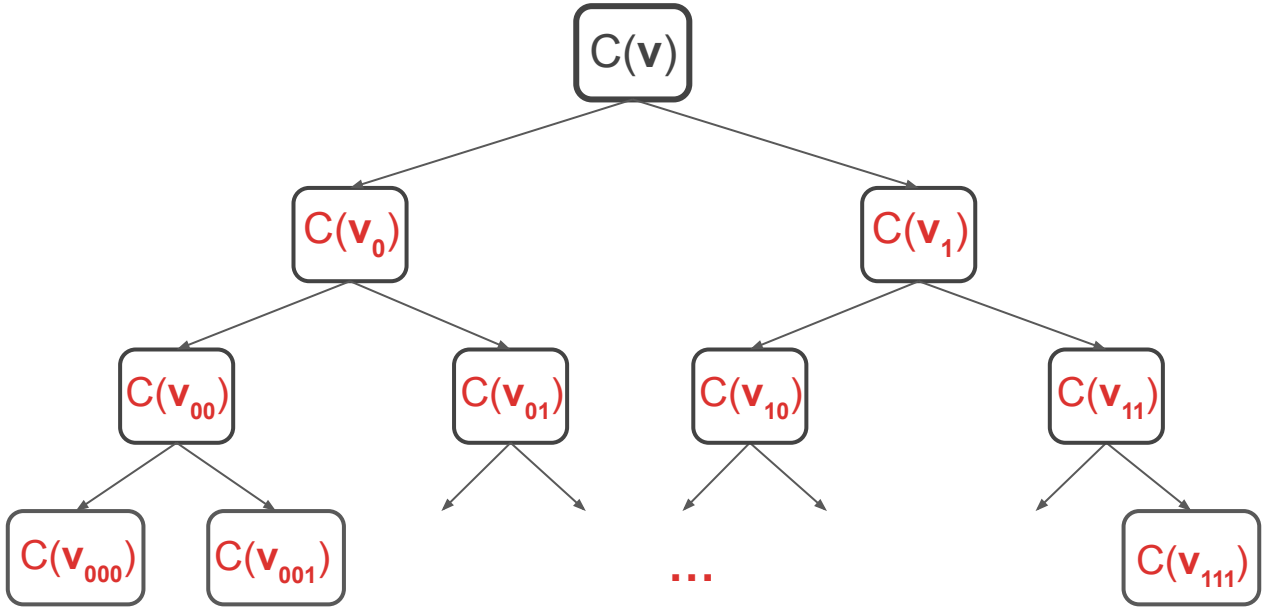
Tree-based Techniques

Ideas for tree-based VC:

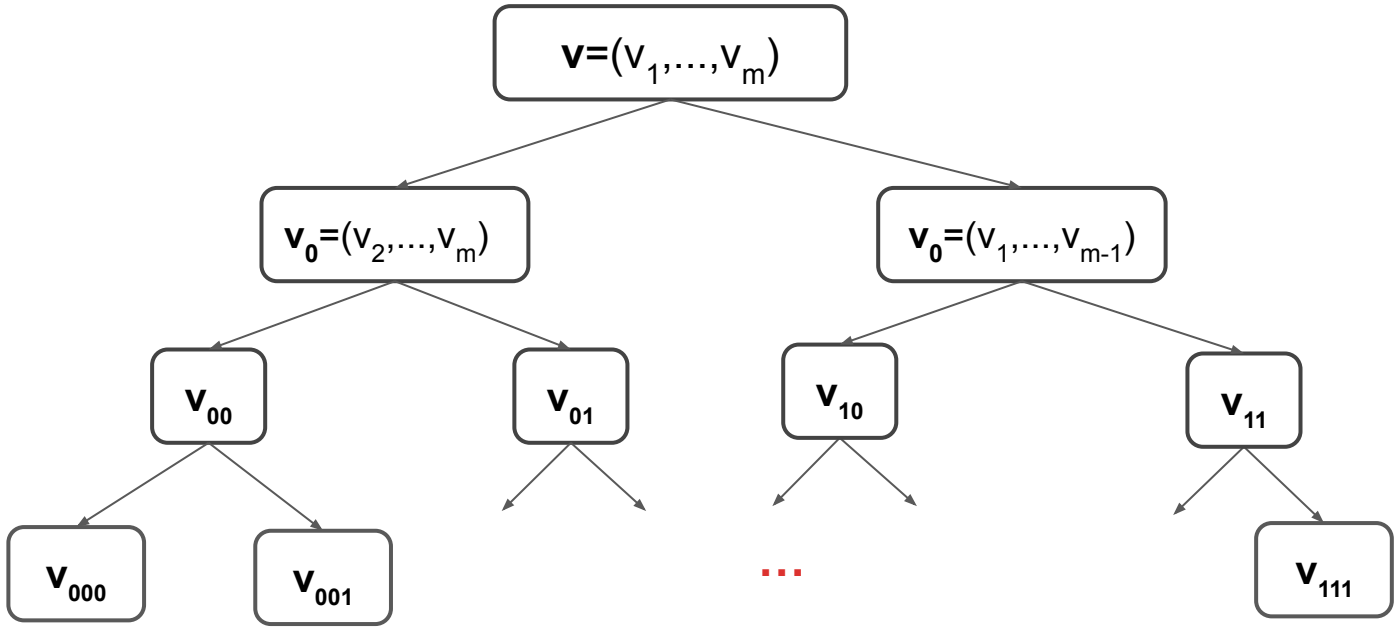
- maintainability for proofs
- generic for any homomorphic VC scheme
- any arity **k** vs. binary
- tune the tradeoff
- from multivariate polynomial to univariate
- **practical**: reusable trusted setup (Groth16)



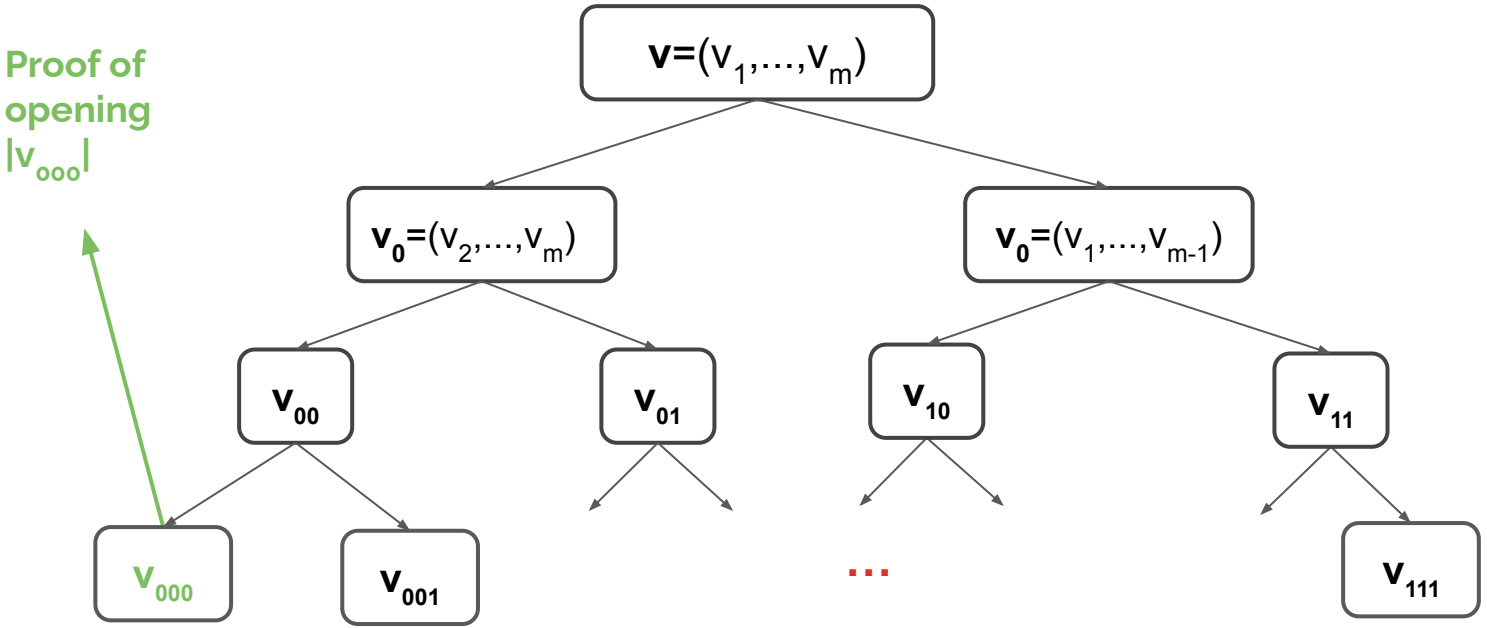
VC Tree



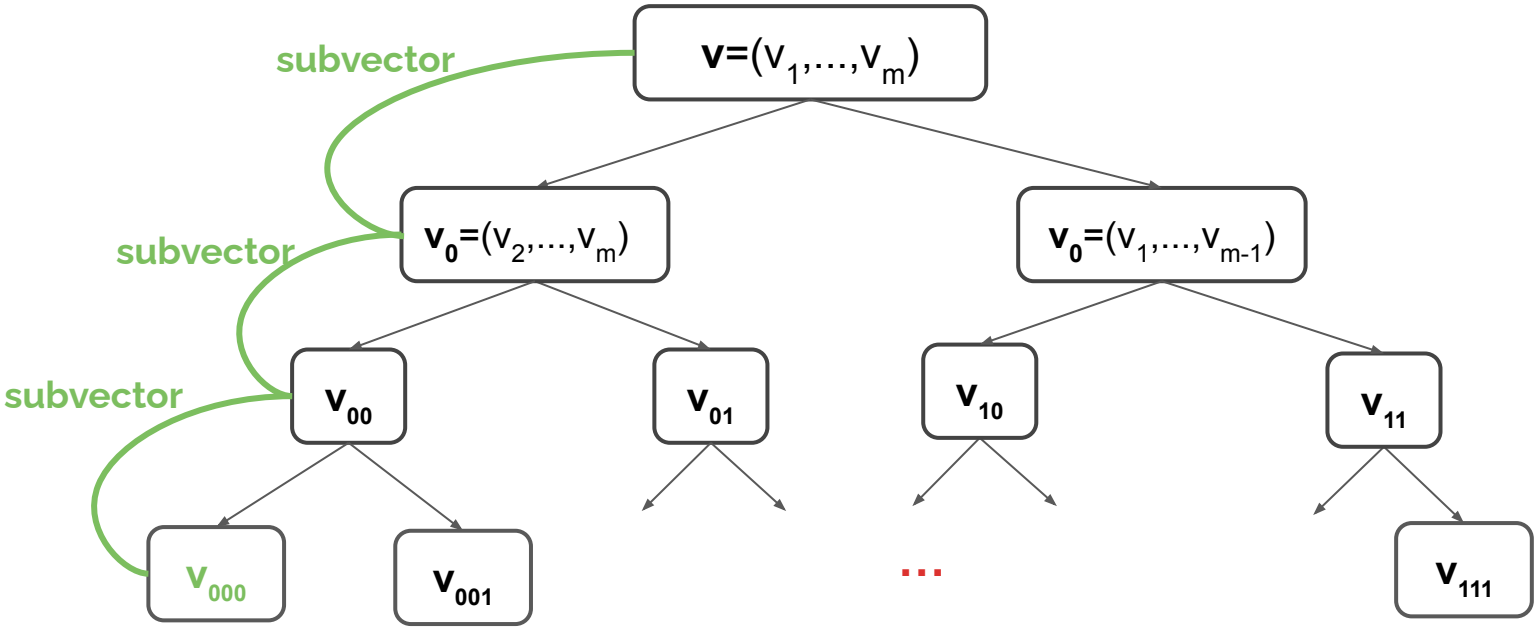
VC Tree



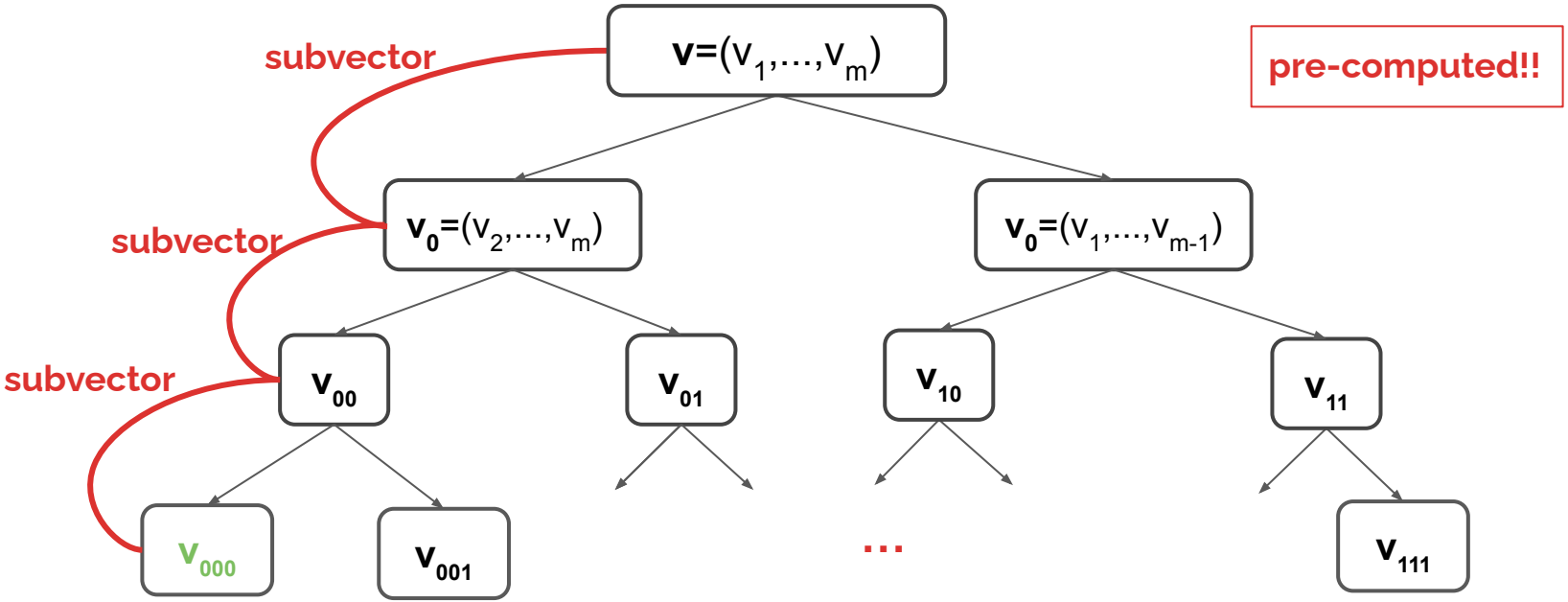
VC Tree - opening proof



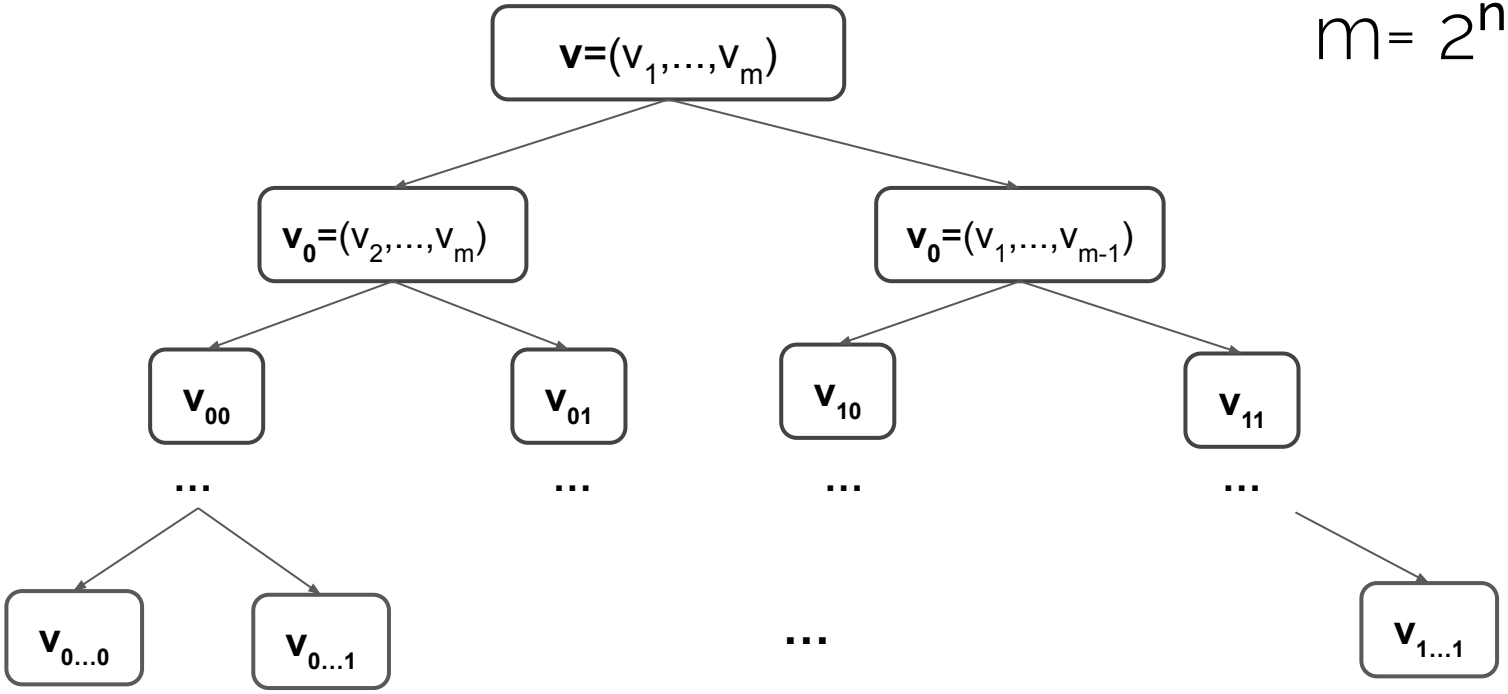
VC Tree - opening proof



VC Tree - opening proof

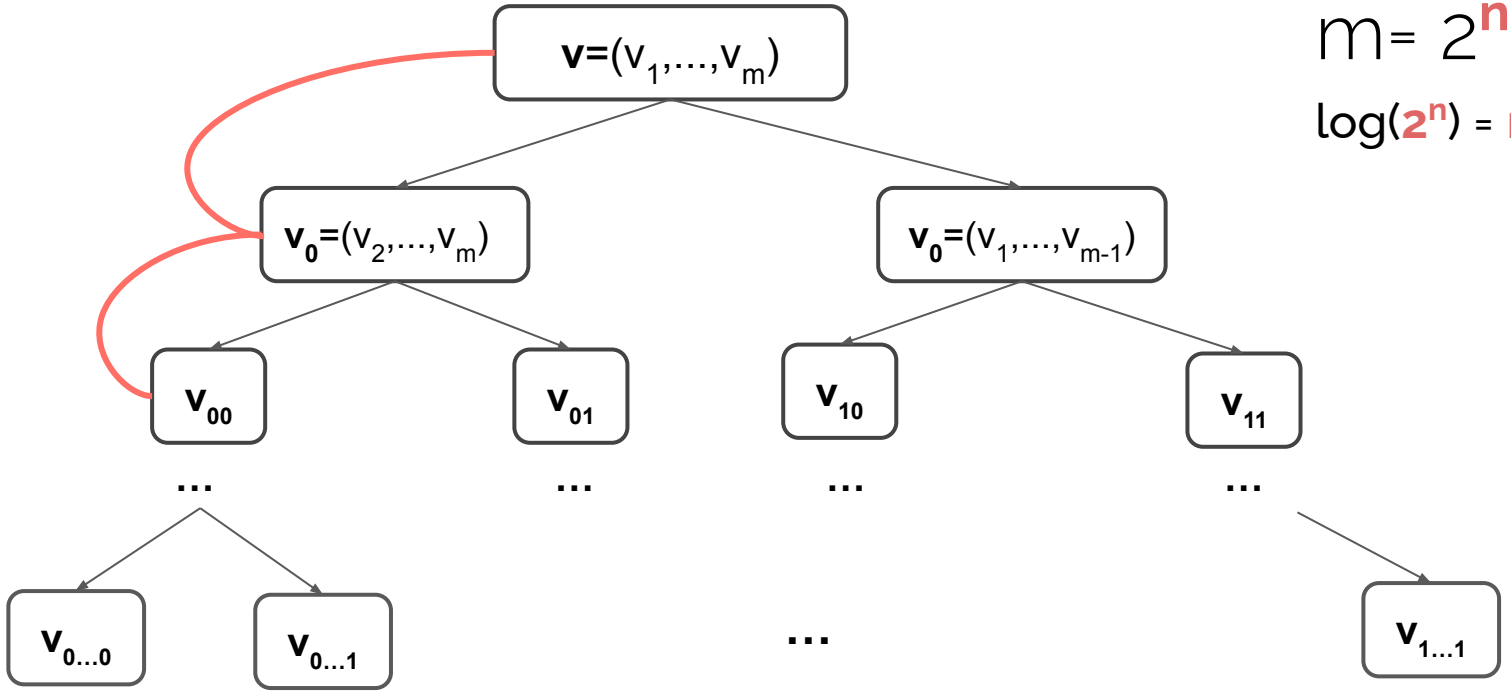


VC Tree



$m = 2^{n+k}$

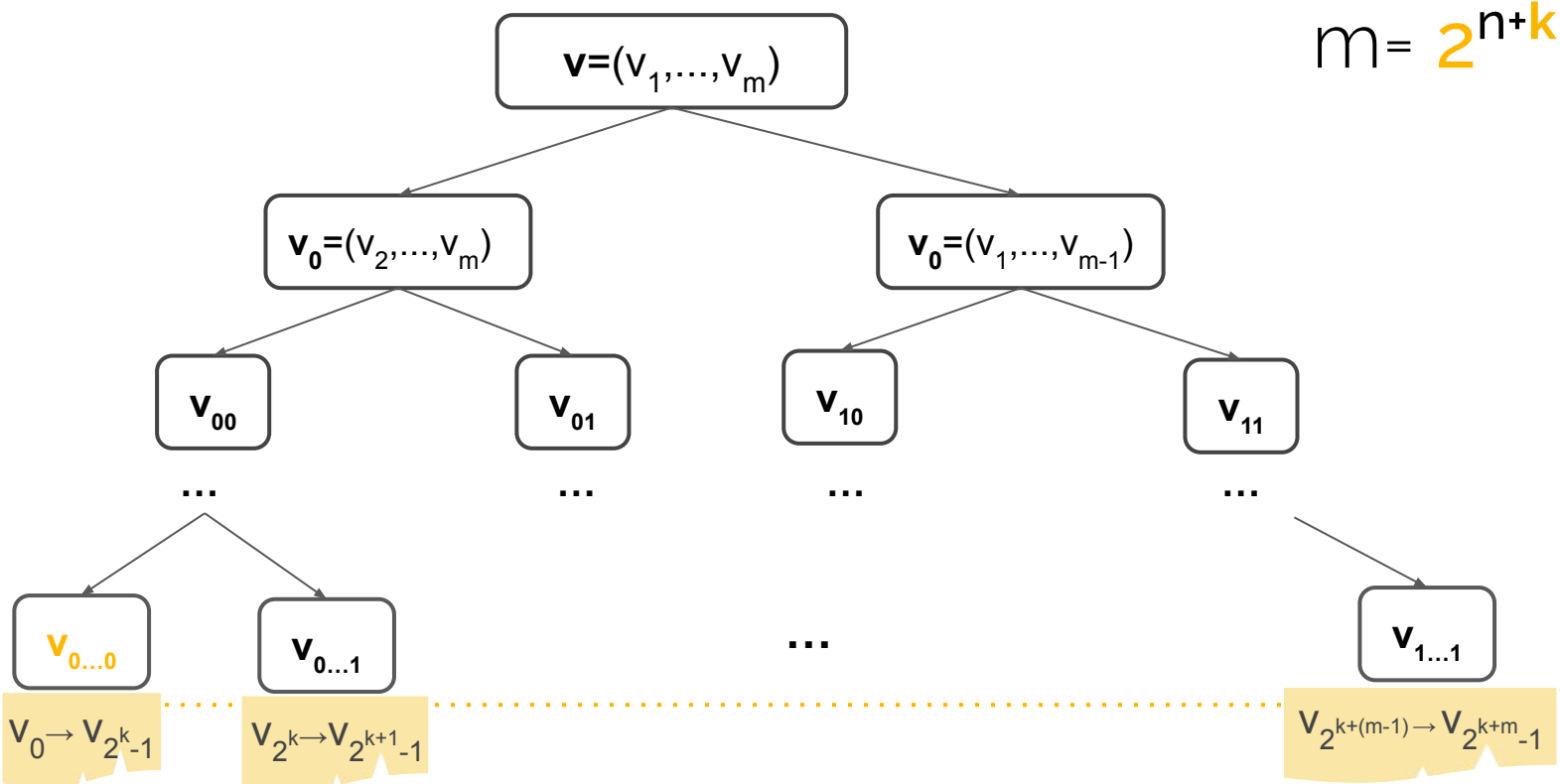
VC Tree



$m = 2^{n+k}$

$\log(2^n) = n$

VC Tree


v=(v1,…,vm)
v0=(v2,…,vm)
v0=(v1,…,vm-1)
v00
v01
v10
v11
…
…
…
…
v0…0
v0…1
…
v1…1
v0→ v2k-1
v2k→v2k+1-1
v2k+(m-1)→ v2k+m-1

$m = 2^{n+k}$

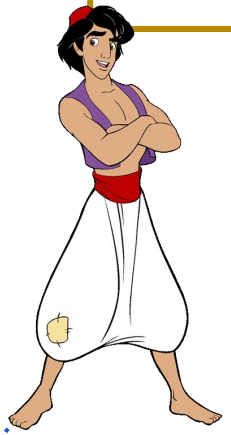
Stateless Cryptocurrency

application

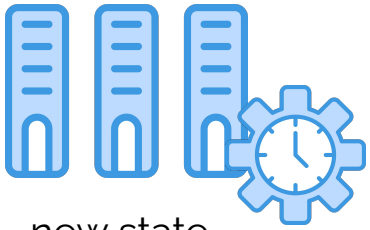
Distributed Ledger

New Block

A → B, **5 Coins**
C → D, **10 Coins**
A → C, **5 Coins**



previous state



new state

A: 20 Coins
B: 55 Coins
C: 30 Coins
D: 0 Coins
E: 90 Coins
F: 10 Coins
...

A: 10 Coins
B: 60 Coins
C: 25 Coins
D: 10 Coins
E: 90 Coins
F: 10 Coins
...

Grows over time

High storage & download cost

Distributed Ledger

New Block

A → B, **5 Co**
C → D, **10 C**
A → C, **5 Co**

Solution:

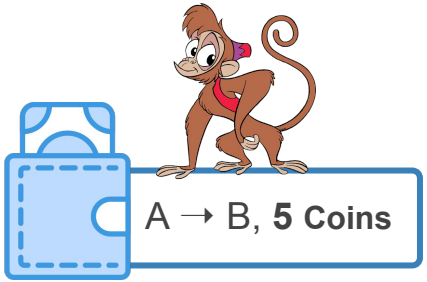
VC

Grows over time

High storage & download cost

Transaction validation

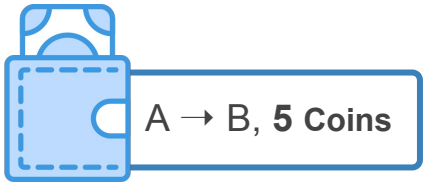
current state



A: 20 Coins
B: 55 Coins
C: 30 Coins
D: 0 Coins
E: 90 Coins
F: 10 Coins
...

Transaction validation

current state



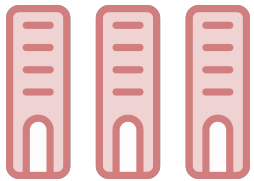
Validate Transaction

A: 20 Coins
B: 55 Coins
C: 30 Coins
D: 0 Coins
E: 90 Coins
F: 10 Coins
...

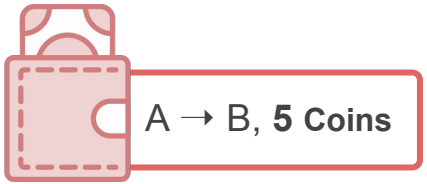
Check that

A has > **5** Coins

Stateful validation



current state



Validate Transaction

A: 20 Coins
B: 55 Coins
C: 30 Coins
D: 0 Coins
E: 90 Coins
F: 10 Coins
...

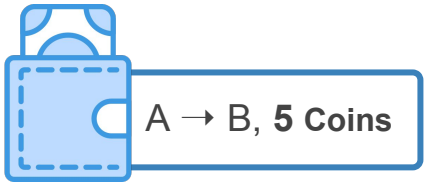
Stateless validation

State Commitment
(vector of balances)

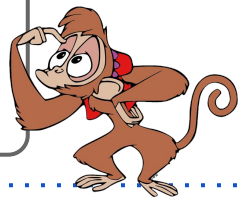
A: 20 Coins
B: 55 Coins
C: 30 Coins
D: 0 Coins
E: 90 Coins
F: 10 Coins
...

Stateless validation

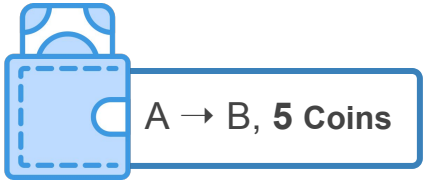
State Commitment
(vector of balances)



A: 20 Coins
B: 55 Coins
C: 30 Coins
D: 0 Coins
E: 90 Coins
F: 10 Coins
...



Stateless validation



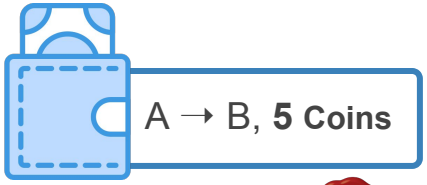
State Commitment
(vector of balances)

A: 20 Coins
B: 55 Coins
C: 30 Coins
D: 0 Coins
E: 90 Coins
F: 10 Coins
...

π_A
π_B
π_C
...

Stateless validation

State Commitment
(vector of balances)



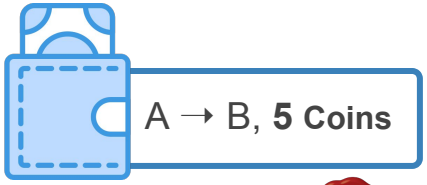
A: 20

A: 20 Coins
B: 55 Coins
C: 30 Coins
D: 0 Coins
E: 90 Coins
F: 10 Coins
...

π_A
π_B
π_C
...

Stateless validation

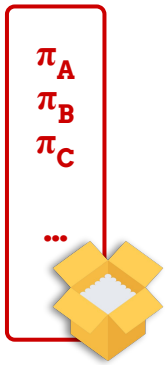
State Commitment
(vector of balances)

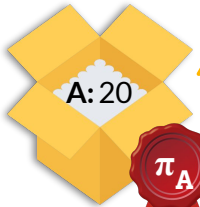


A: 20

A: 20 Coins
B: 55 Coins
C: 30 Coins
D: 0 Coins
E: 90 Coins
F: 10 Coins
...

π_A
π_B
π_C
...



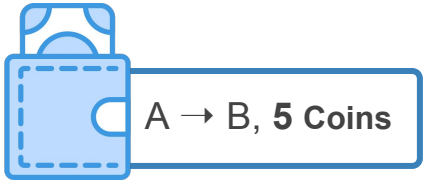


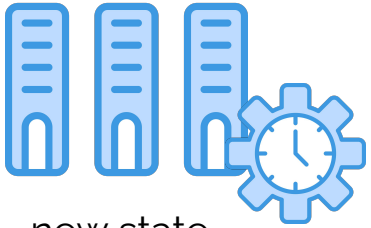
Check that

π_A is a valid opening for **Com**

More Requirements

State Commitment
(vector of balances)

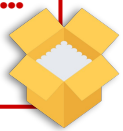




new state

A: 20 Coins
B: 55 Coins
C: 30 Coins
D: 0 Coins
E: 90 Coins
F: 10 Coins
...

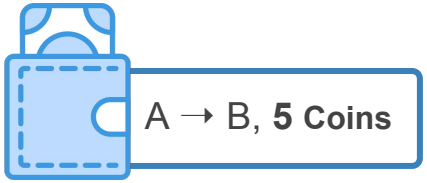
π_A
π_B
π_C
...



A: 15 Coins
B: 60 Coins
C: 30 Coins
D: 0 Coins
E: 90 Coins
F: 10 Coins
...

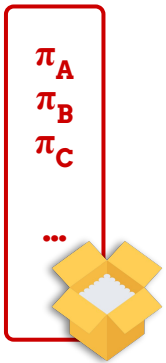
Updatability for Com

State Commitment
(vector of balances)



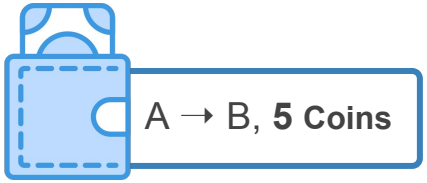
Update

A: 20 Coins
B: 55 Coins
C: 30 Coins
D: 0 Coins
E: 90 Coins
F: 10 Coins
...



A: 15 Coins
B: 60 Coins
C: 30 Coins
D: 0 Coins
E: 90 Coins
F: 10 Coins
...

Updatability for Proofs

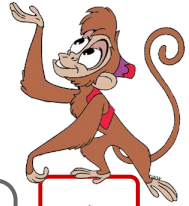


State Commitment
(vector of balances)

A: 20 Coins
B: 55 Coins
C: 30 Coins
D: 0 Coins
E: 90 Coins
F: 10 Coins
...

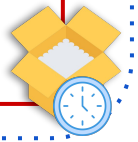
π_A
π_B
π_C
...

Update all



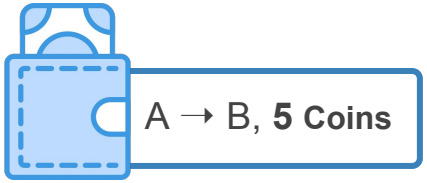
A: 15 Coins
B: 60 Coins
C: 30 Coins
D: 0 Coins
E: 90 Coins
F: 10 Coins
...

π'_A
π'_B
π'_C
...

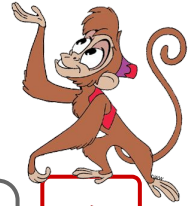


Updatability for Proofs

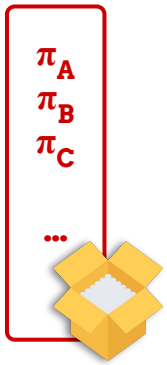
State Commitment
(vector of balances)



Update all



A: 20 Coins
B: 55 Coins
C: 30 Coins
D: 0 Coins
E: 90 Coins
F: 10 Coins
...

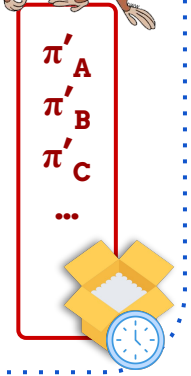


Updates types

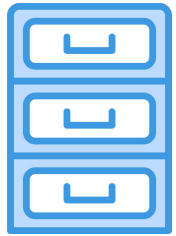
- **hint**: needs opening
- **key**: needs static key

\+ **maintainability**

A: 15 Coins
B: 60 Coins
C: 30 Coins
D: 0 Coins
E: 90 Coins
F: 10 Coins
...

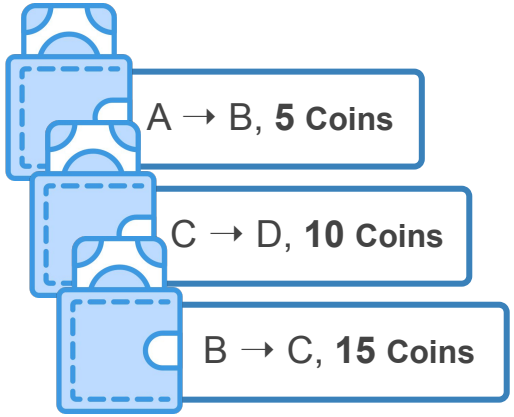


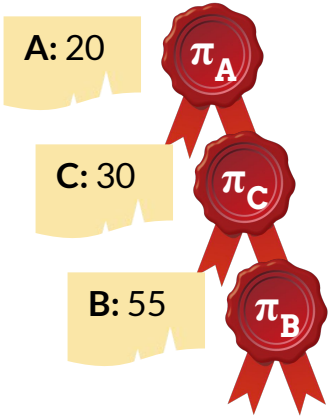
Aggregation



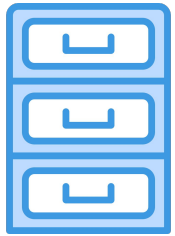
New Block

A → B, **5 Coins**
C → D, **10 Coins**
B → C, **15 Coins**



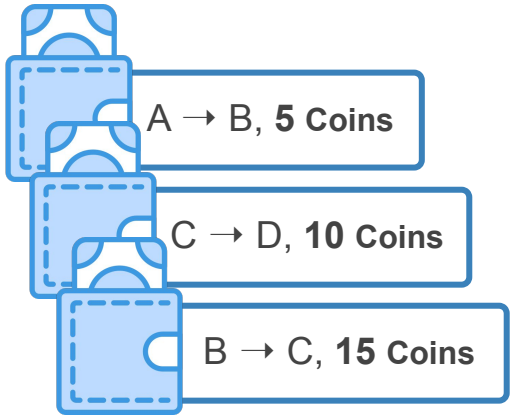


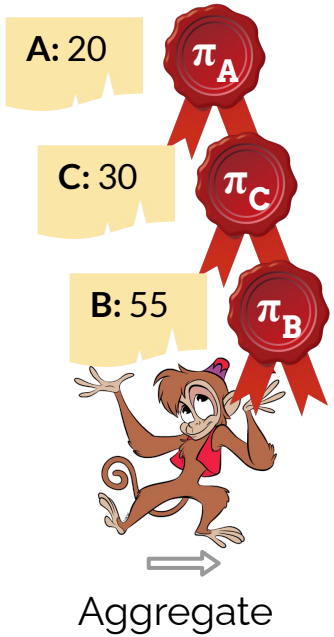
Aggregation

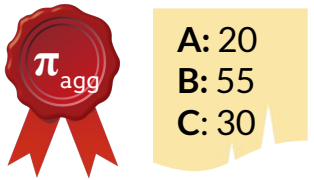


New Block

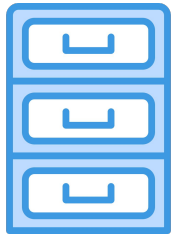
A → B, **5 Coins**
C → D, **10 Coins**
B → C, **15 Coins**





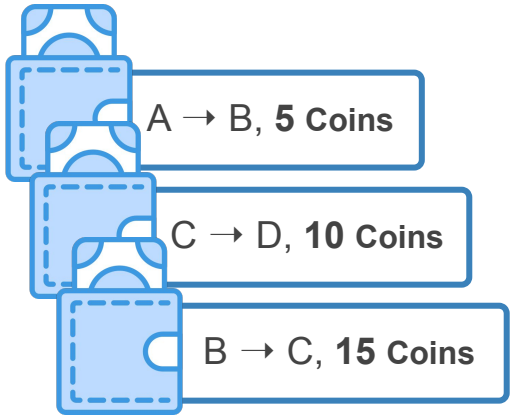


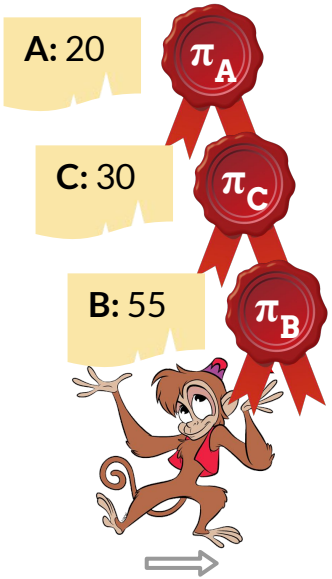
Aggregation

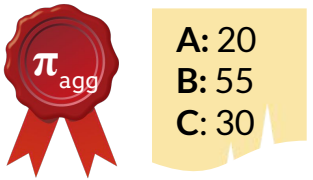


New Block

A → B, **5 Coins**
C → D, **10 Coins**
B → C, **15 Coins**







- same-commitment/**cross-commitment**
- one-hop/**unbounded**
- **incremental:** can disaggregate

Future Directions

What's next?

- Transparent Setup Vector Commitments
- Functional Vector Commitments for larger classes of functions
- Post-Quantum Vector Commitments with same properties
  - Code-based or lattice-based?
- More efficient Aggregation for Tree-based VC (relying on IPA)

Thanks!

ia.cr/2022/705

Credits

Special thanks to all those who made and released these resources for free:

- Presentation template by SlidesCarnival
- Clip arts by Iconfinder
- Illustrations by Disneyclips