Vector Commitments:

From Theory to Practice, and Back Again

Dario Fiore IMDEA Software Institute, Spain

RECSI 2022 — Santander, October 21, 2022

European Research Council
Established by the European Commission

Let's go discover the planets of vector commitments!

Commitments

Commit

Commitments

Opening

*"It's **m** inside the envelope"*

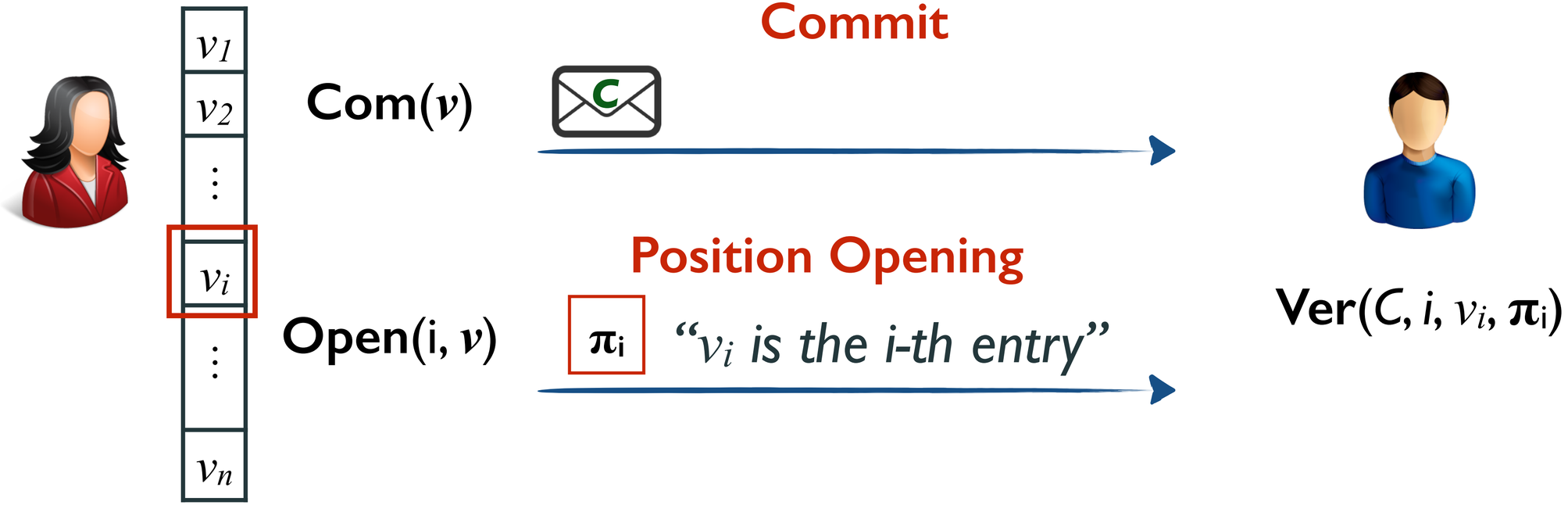
Security properties

Hiding: Bob does not learn anything about ***m*** before Alice opens 

Binding: Alice cannot change her mind about ***m***

Vector Commitments

[Libert-Yung, TCC'10]
[Catalano-Fiore, PKC'13]



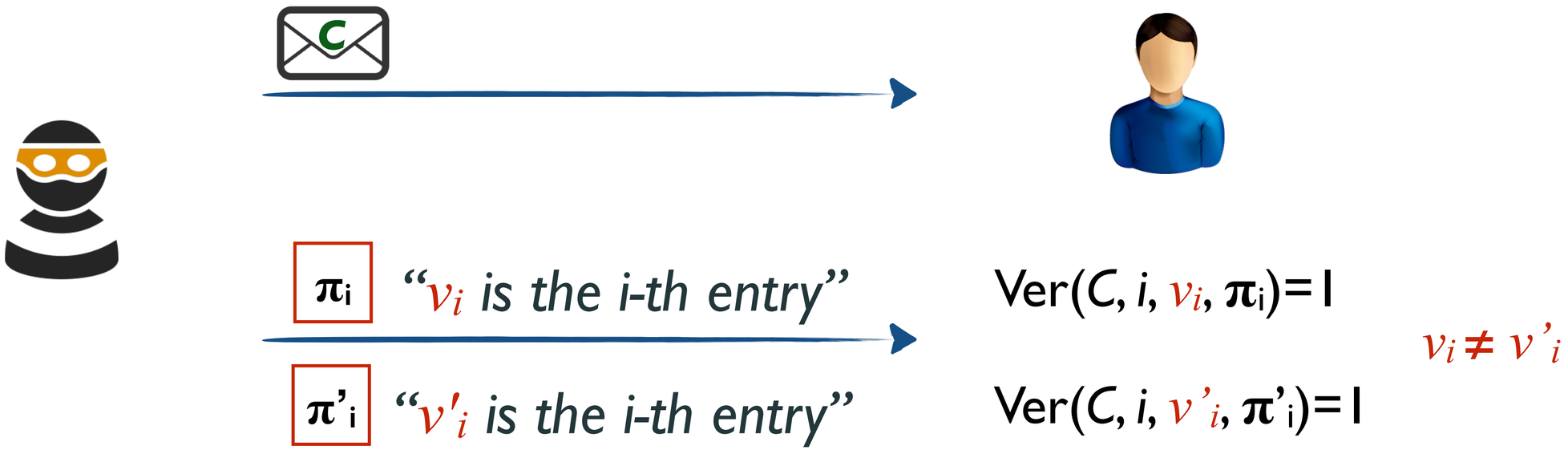
Basic idea: commit to a vector and selectively open single entries

Key properties:

position binding

conciseness

Position binding

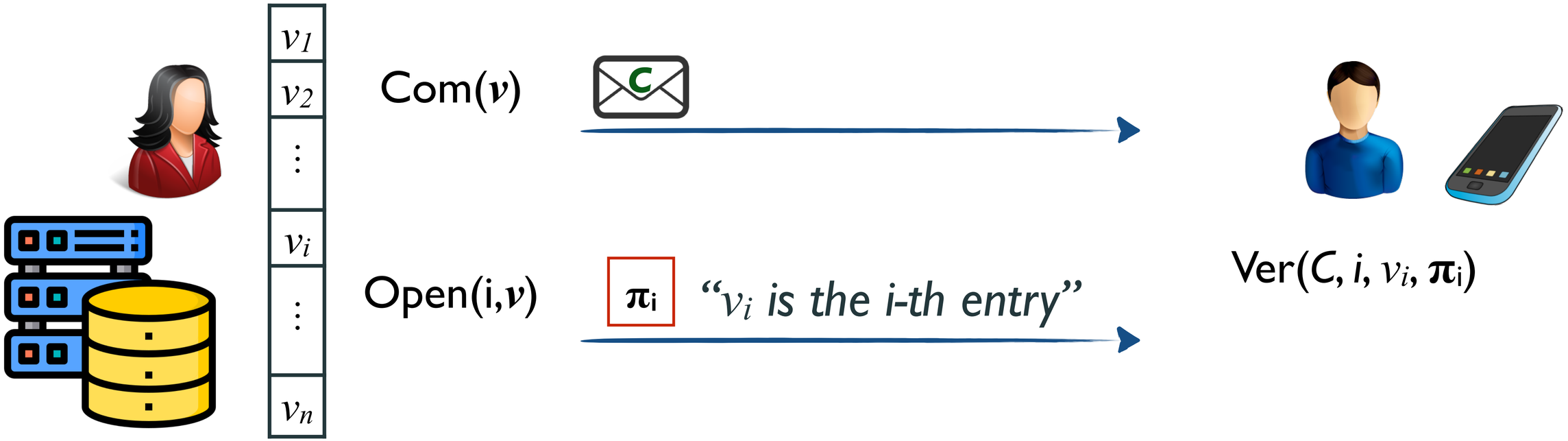


Cannot open the same position to two different values

i.e., computationally hard to find

$(C, i, v_i, \pi_i, v'_i, \pi'_i)$ s.t. $v_i \neq v'_i \wedge \text{Ver}(C, i, v_i, \pi_i)=1 \wedge \text{Ver}(C, i, v'_i, \pi'_i)=1$

Conciseness



Size of commitment and openings is independent of n

$$|C| = |\pi_i| = p(\lambda)$$

Main difference with **Merkle trees** where $|\pi_i| = O(\lambda \log n)$

Vector Commitments (VCs)

Setup(1^λ, n)$\rightarrow pp$

Com(pp, $\boldsymbol{v}$)$\rightarrow$ C

Open(pp, i, $\boldsymbol{v}$)$\rightarrow \boldsymbol{\pi}_i$

Ver(pp, C, i, $\boldsymbol{v}_i$, $\boldsymbol{\pi}_i$)$\rightarrow$ 0/1

Correctness. Ver(pp, Com(pp, $\boldsymbol{v}$), i, $\boldsymbol{v}_i$, Open(pp, i, $\boldsymbol{v}$)) = 1

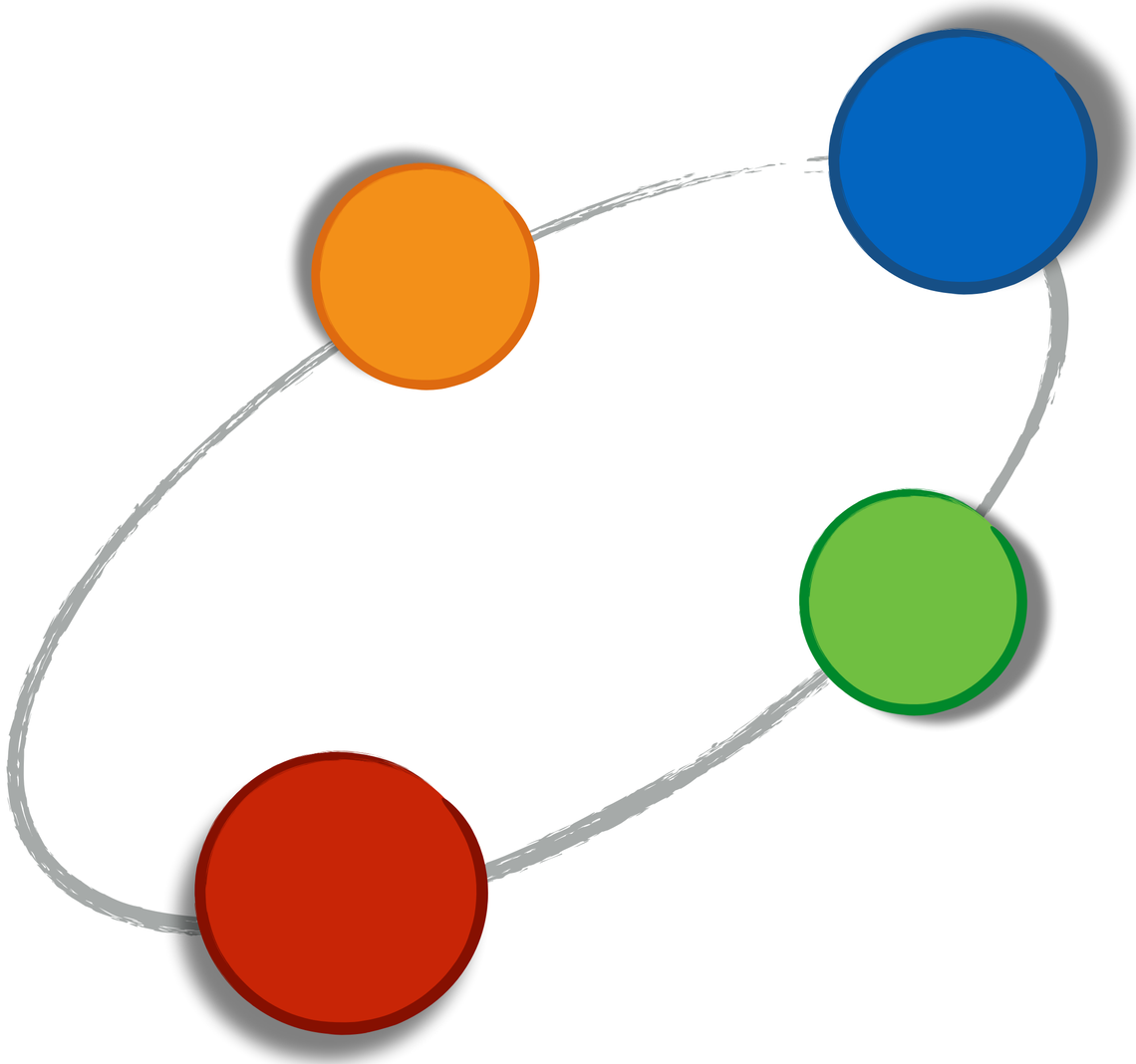
Conciseness. $|C| = |\boldsymbol{\pi}_i| = p(\lambda)$

Position binding. for any PPT A and $pp \leftarrow$Setup(1^λ, n)

$$\Pr[v_i \neq v'_i \wedge \text{Ver}(C, i, v_i, \boldsymbol{\pi}_i)=1 \wedge \text{Ver}(C, i, v'_i, \boldsymbol{\pi}'_i)=1 \mid (C, i, v_i, \boldsymbol{\pi}_i, v'_i, \boldsymbol{\pi}'_i) \leftarrow A(pp)]=\text{negl}$$

Our journey in vector commitments

- What are vector commitments?
- **Theory → Practice**
- Practice → Theory
  - Updatability
  - Subvector openings
  - Aggregation
- State of the art
- New frontiers
- Conclusion and open problems



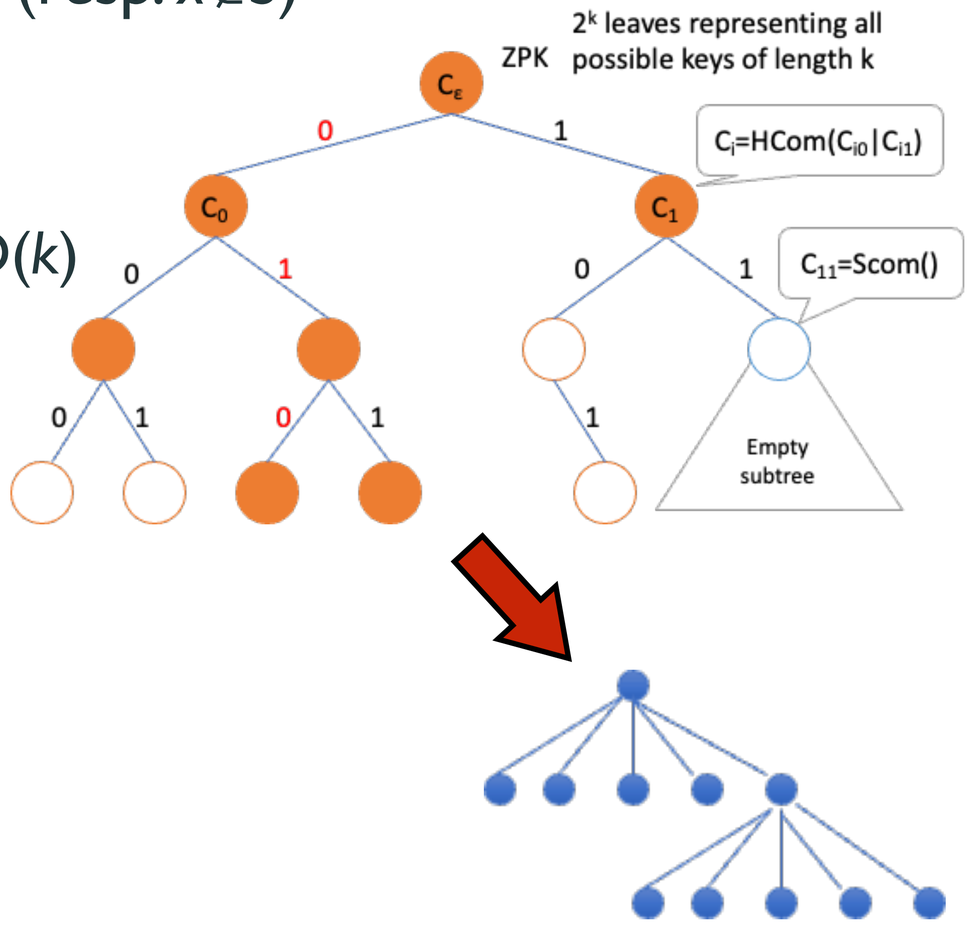
App#1: Zero-knowledge sets with short proofs

Disclaimer: no many details but interesting story about VCs' birth

ZKS [MRK03]: commit to set S and prove in ZK $x \in S$ (resp. $x \notin S$)

[MRK03,CHLMR05] ZKS construction uses a
sparse Merkle tree of "mercurial commitments"

(non)membership proof = opening comm.s in the path $O(k)$



In [CFM08] we asked:

Can we make ZKS trees shallow?

replacing 2-to-1 with n-to-1 ⇒ proof size $O(\log_n(2^k))$

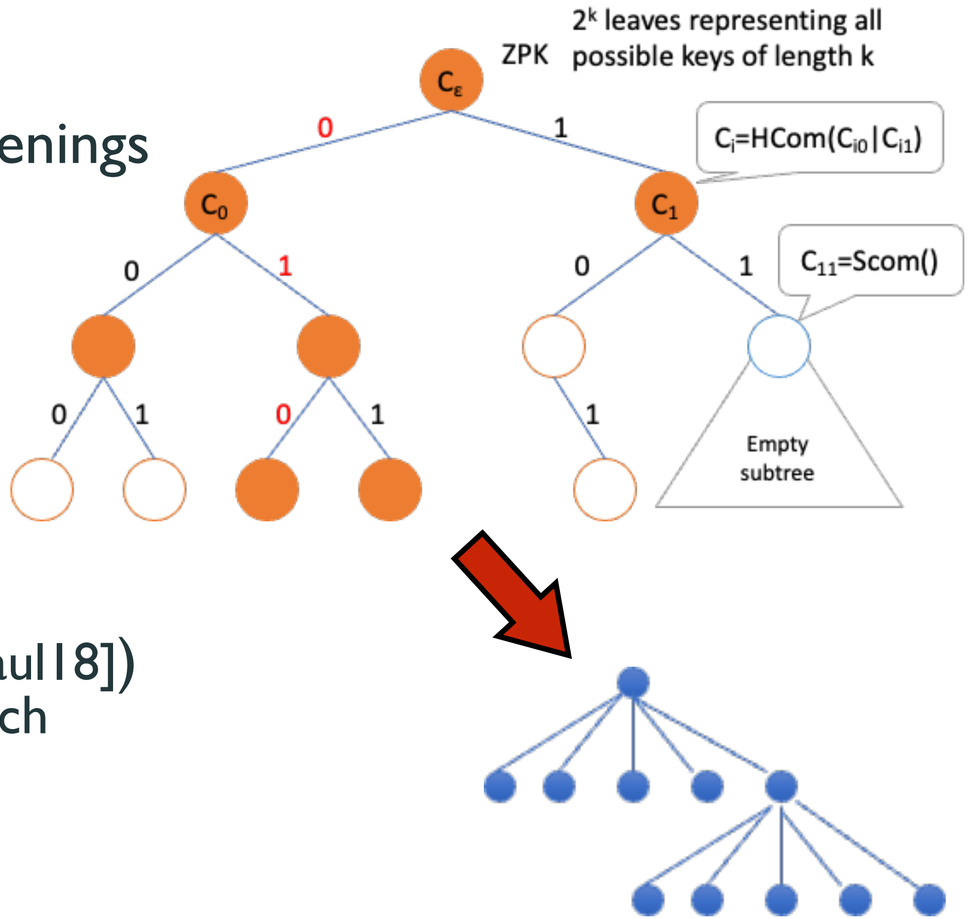
App#1: Zero-knowledge sets with short proofs

In [CFM08] we came up with the notion of *n*-TMC
(implicitly a VC with hard&soft openings to selected positions)

…but only realized O(1) soft openings

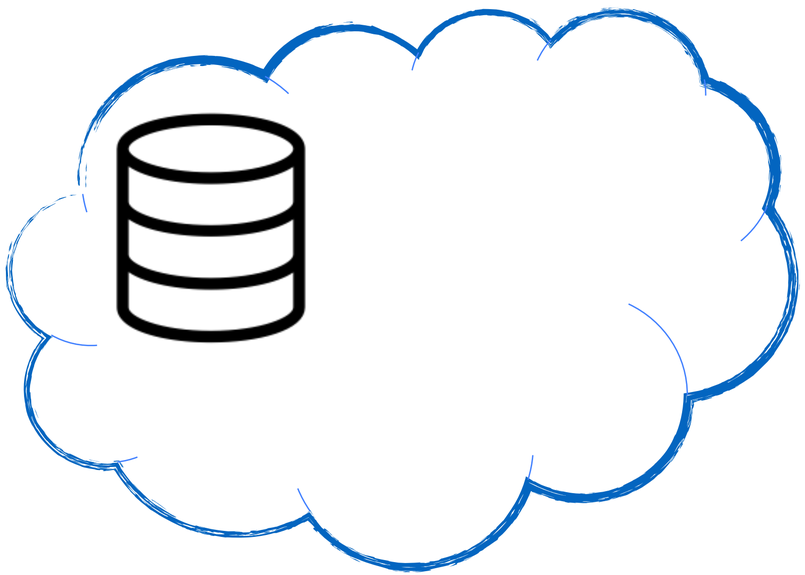
[LY10] first realization of *n*-TMC with *O(1)* (hard&soft) openings from the n-DHE assumption

[CF13] VC formalization and two constructions from standard assumptions (RSA, CDH)



(Shallow Merkle trees today known as Verkle trees [Kuszmaul18])
Lesson for me: importance of theoretical/foundational research

App#2: Outsourced storage



Please store

Are you still storing my data?

Here you go…

Think of
very large data

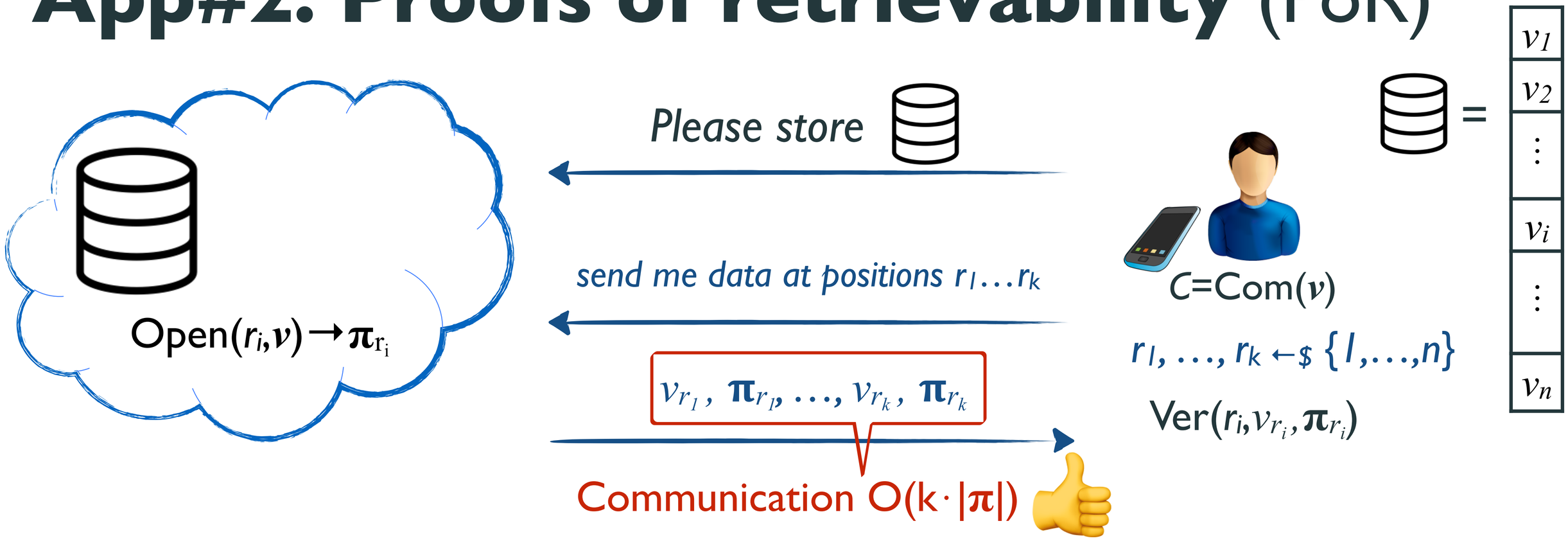
BIBLIOTECA NACIONAL DE ESPAÑA
BNE

INTERNET ARCHIVE

Trivial solution: Bob stores Hash( ) and Cloud sends

⇒ too much communication

App#2: Proofs of retrievability (PoR)



[JK07] PoR construction using Merkle trees (generalized to VCs by [Fisch18])

Bob stores C=Com(v) and deletes v

Periodically asks k≈λ random entries with their proofs and checks they are valid w.r.t. C

App#3: Stateless validation (account-based cryptocurrencies)

ethereum

1) state of the blockchain

key-value map

v_1	v_2	…	v_i	…	v_n
pk_1	pk_2	…	pk_i		pk_n

2) transaction

$$\mathbf{tx}=[\ pk_i \rightarrow pk_j : v\ ]$$

3) validating miner

is $v \leq v_i$?

Scaling problem: can validators work without storing the entire state?

App#3: Stateless validation via vector commitments

[Edrax,CPZ18]

1) state of the blockchain

key-value map

v_1	v_2	…	v_i	…	v_n
pk_1	pk_2	…	pk_i		pk_n

vector

$H(pk_1)\|\|v_1$	$H(pk_2)\|\|v_2$	…	$H(pk_i)\|\|v_i$	…	$H(pk_n)\|\|v_n$
1	2	…	i	…	n

C=Com( )

2) transaction

tx=[$pk_i \rightarrow pk_j : v,\ \boldsymbol{\pi}_i,\ v_i$]

3) validating miner

C

is $v \le v_i$ and

Ver(**C**, i, v_i, $\boldsymbol{\pi}_i$)=1?

Using VCs → validators store $|C|$ and additionally receive $|\boldsymbol{\pi}_i|$ bits per *tx*

Theory → Practice

App#1: ZKS with short proofs — theoretical, it motivated the VC notion

now, idea of VCs for shallow Merkle trees is considered in practice

App#2: Proof of storage — practical, motivated by outsourced storage

App#3: Stateless blockchains — practical, motivated by blockchains

More applications: verifiable databases, accumulators, succinct arguments…

Two take-home messages

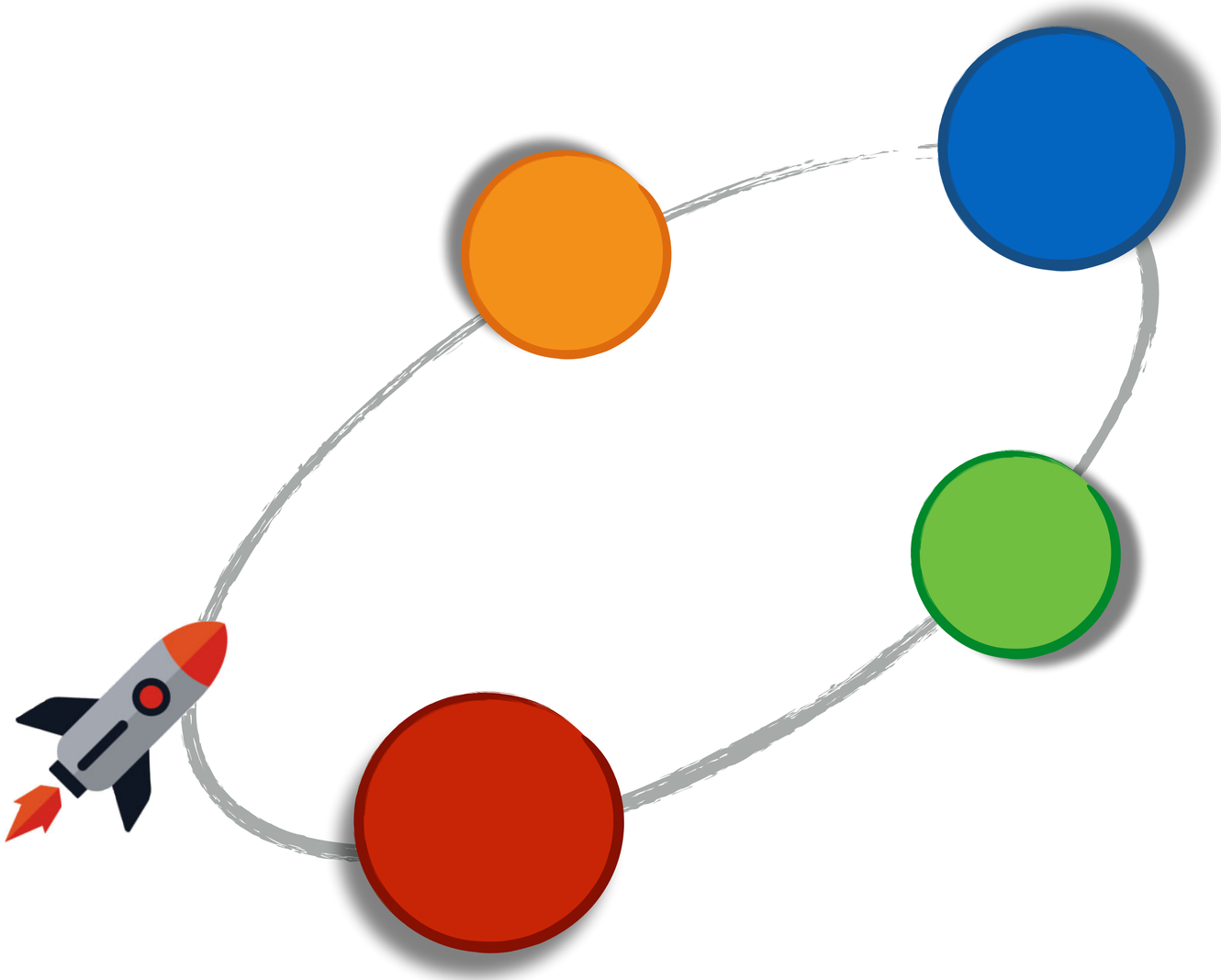
- *importance of* ***theory-motivated*** *research questions*: gave rise to VC notion!

- *importance of* ***practice-motivated*** *research questions*: gave rise to further VC ideas

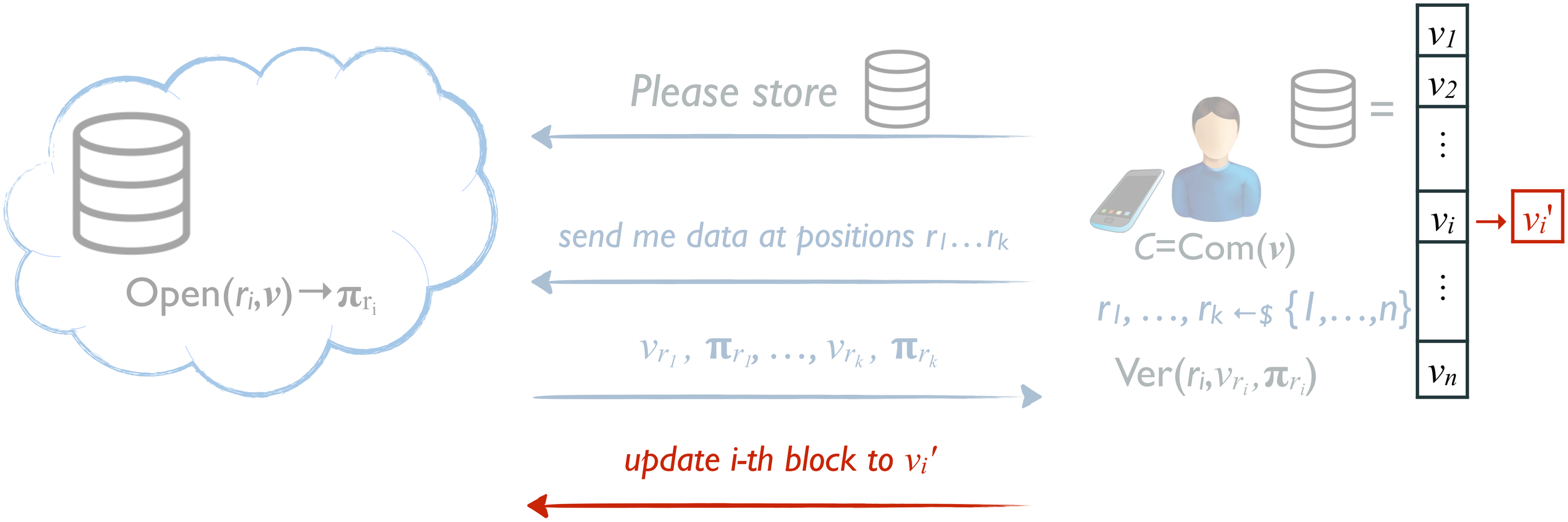
➜ see next!

Our journey in vector commitments

- What are vector commitments?
- Theory → Practice
- **Practice → Theory**
  - Updatability
  - Subvector openings
  - Aggregation
- State of the art
- New frontiers
- Conclusion and open problems



App#2 Outsourced storage: what if data changes?



Who recomputes the commitment to the new data?

Updatable VCs [CF13]

VC

Setup(1^λ, n)$\rightarrow pp$ Com(pp, $\boldsymbol{v}$)$\rightarrow C$

Open(pp, i, $\boldsymbol{v}$)$\rightarrow \boldsymbol{\pi}_i$ Ver(pp, C, i, v_i, $\boldsymbol{\pi}_i$)$\rightarrow$ 0/1

VC is updatable if there exist the following algorithms:

UpdateCom(pp, C, i, v_i, v'_i, [$\boldsymbol{\pi}_i$])$\rightarrow C'$ // C' commits to (v_1,…, v'_i,…, v_n,)

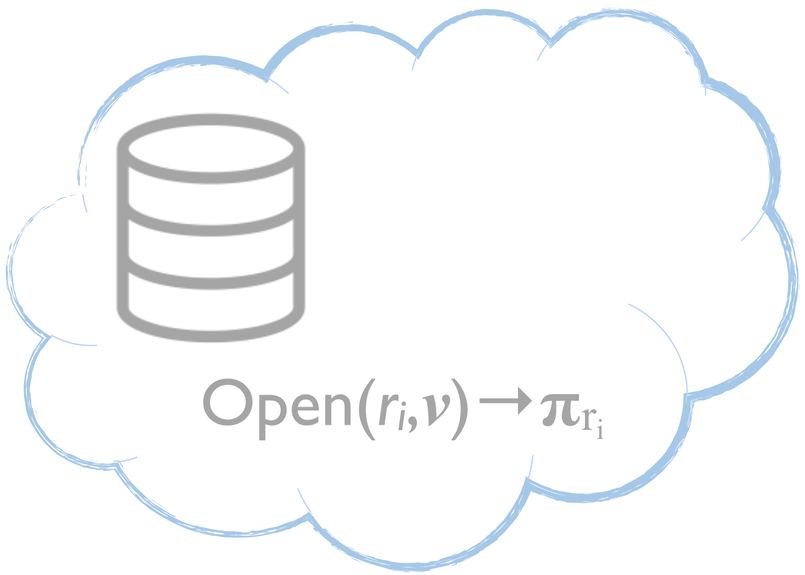
UpdateOpen(pp, $\boldsymbol{\pi}_j$, i, v_i, v'_i, [$\boldsymbol{\pi}_i$])$\rightarrow \boldsymbol{\pi}_j'$ // $\boldsymbol{\pi}_j'$ valid for the new commitment C'

Updatability model:

$\boldsymbol{\pi}_i$ **not** needed ↓ VC is updatable

$\boldsymbol{\pi}_i$ needed ↓ VC hint-updatable

Outsourced storage via updatable VCs



Please store

send me data at positions $r_1 \ldots r_k$

v_{r_1}, π_{r_1}, …, v_{r_k}, π_{r_k}

Update i-th block to v_i'

C=Com(v)

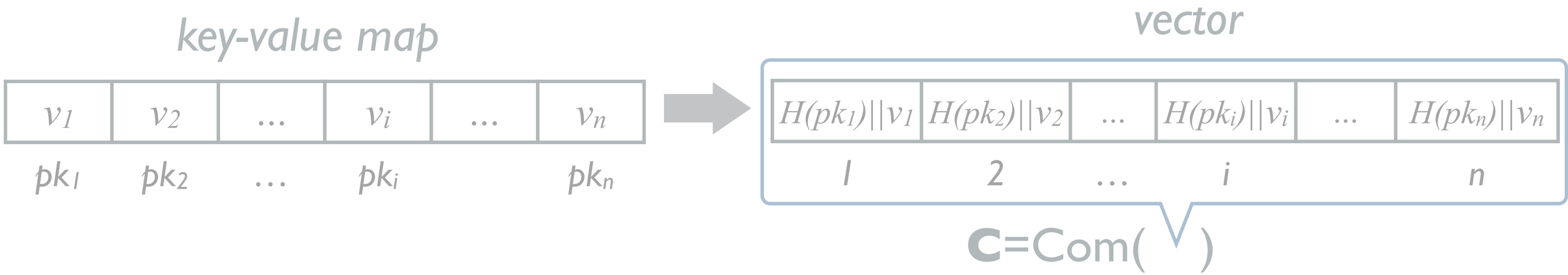
$r_1, \ldots, r_k \leftarrow\$ \{1,\ldots,n\}$

C'←UpdateCom(pp, C, i, v_i, v_i', $[\pi_i]$)

*Here updating openings is not crucial. It's important in other applications (e.g., decentralized storage)

App#3 Stateless validation: batches of transactions?

1) state of the blockchain



2) *m* transactions

$\{\mathbf{tx}_k=[\ pk_{i_k} \rightarrow pk_{j_k} : v_k,\ \boldsymbol{\pi}_{i_k},\ v_{i_k}\ ]\}_{k=1\ldots m}$

3) validating miner

C

Scaling problem: can we avoid storing all the *m* proofs in the new block?

VCs with subvector openings [BBF19,LM19]

Setup($1^\lambda, n$) → pp

Com(pp, v) → C

Open(pp, i, v) → π_i

Ver(pp, C, i, v_i, π_i) → 0/1

Open several positions with one proof

Open($pp, I=\{i_1,\ldots,i_m\}, v$) → π_I

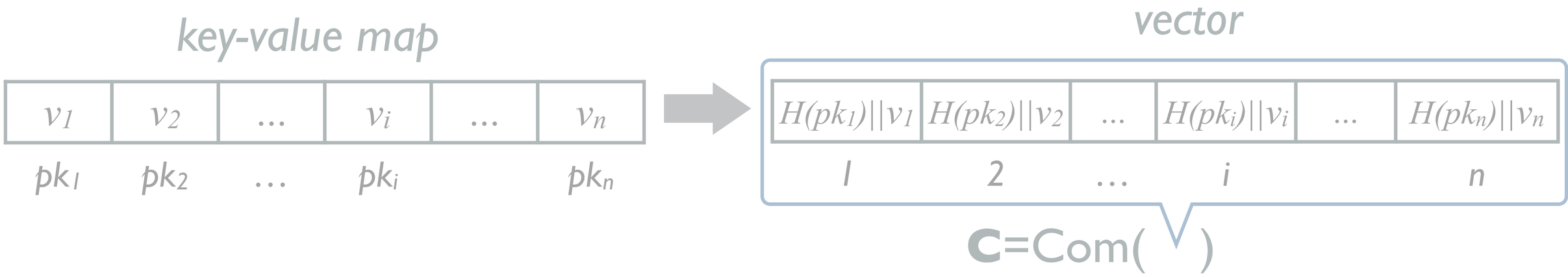
Ver(pp, C, I, v_I, π_I) → 0/1

of fixed size p(λ)

(independent of m, n)

Validating batches of transactions using aggregatable VCs

1) state of the blockchain



2) *m* transactions

$\{tx_k=[\ pk_{i_k} \rightarrow pk_{j_k} : v_k,\ \pi_{i_k},\ v_{i_k}\ ]\}_{k=1\ldots m}$

3) validating miner

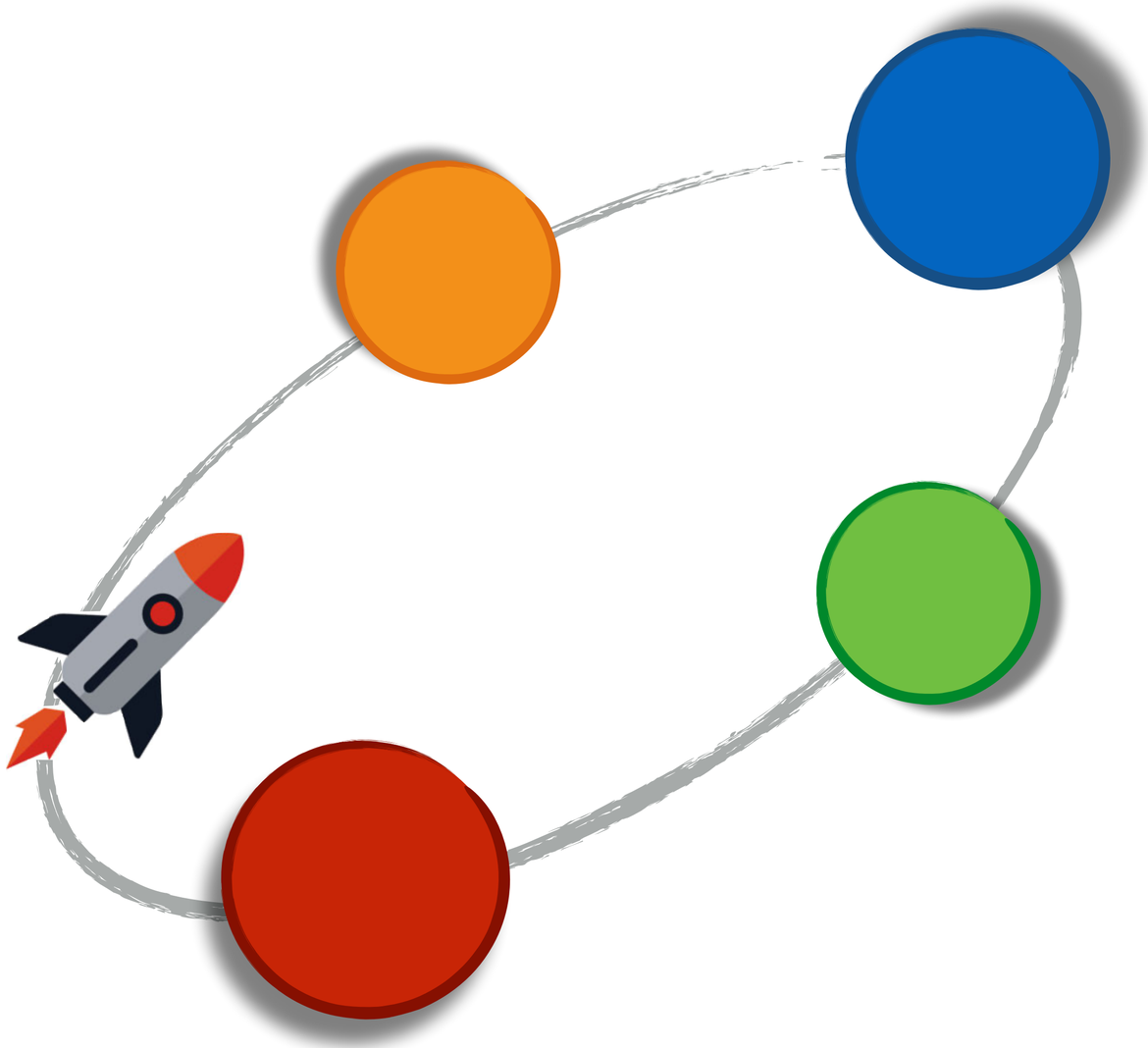
C

Using aggregatable VCs → send $\pi_{\{i_1, \ldots, i_m\}}$ instead of $\pi_{i_1}, \ldots, \pi_{i_m}$

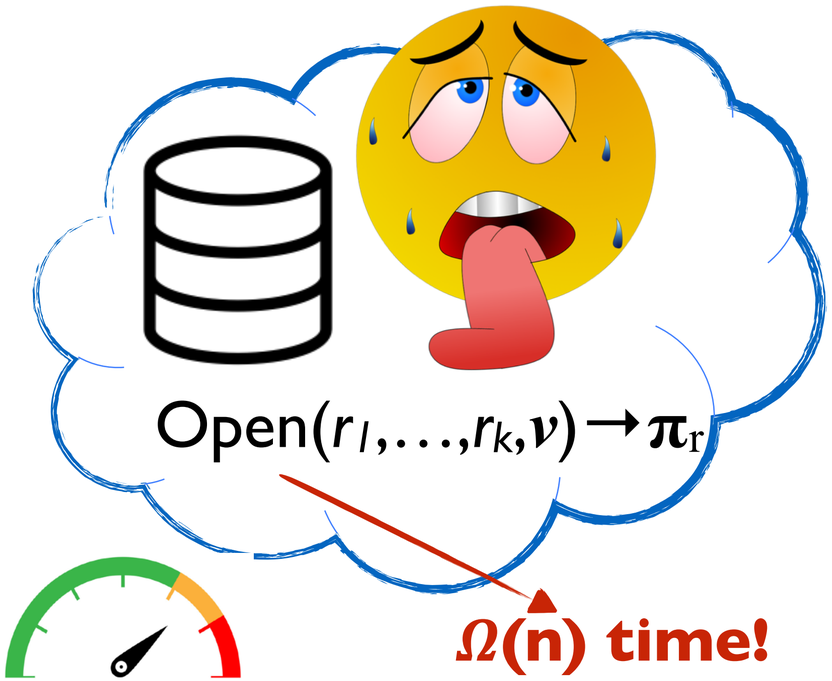
(recall, $|\pi_{\{i_1, \ldots, i_m\}}|$ doesn't depend on *m*)

Our journey in vector commitments

- What are vector commitments?
- Theory → Practice
- Practice → Theory
- Updatability
- Subvector openings
- **Aggregation**
- State of the art
- New frontiers
- Conclusion and open problems



A second look at efficiency of PoR



The VC opening problem

Please store

send me data at positions $r_1 \ldots r_k$

$v_{r_1}, \ldots, v_{r_k}$, π_r

C=Com(v)

$r_1, \ldots, r_k \leftarrow\$ \{1,\ldots,n\}$

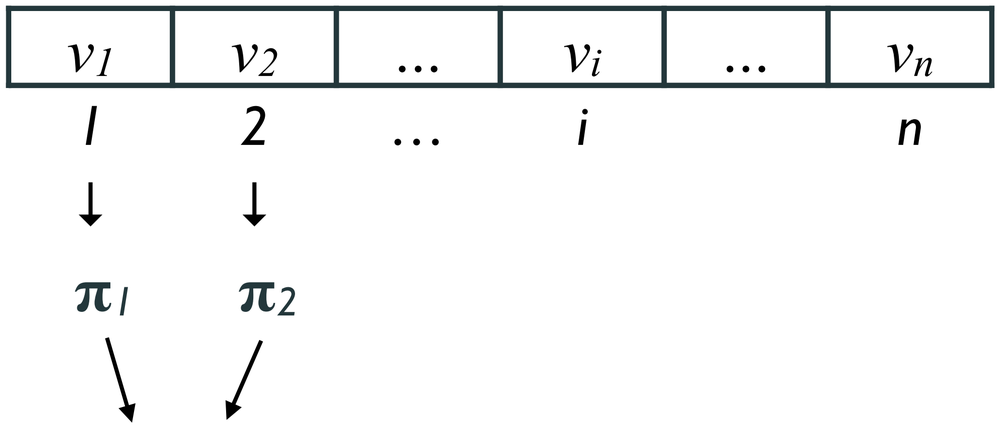
Ver($\{r_1,\ldots,r_k\}, v_{r_1}, \ldots, v_{r_k}, \pi_r$)

How is life for the prover?

Verifier's life:

- short storage
- short communication

Aggregation to rescue provers! [CFGKN20]



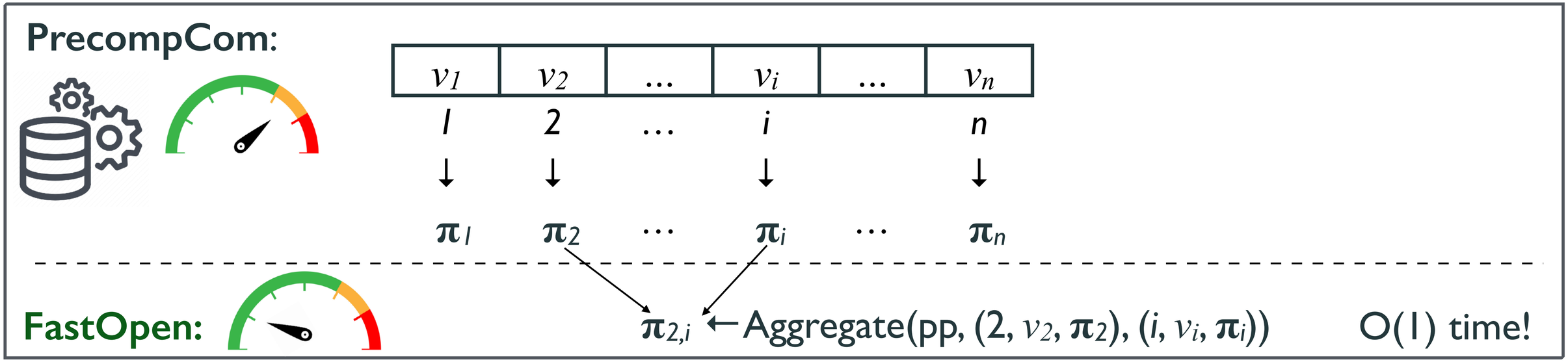
$\pi_{\{1,2\}} \leftarrow \text{Aggregate}(pp, C, (1, v_1, \pi_1), (2, v_2, \pi_2))$

more generally: $\text{Aggregate}(pp, C, (I, v_I, \pi_I), (J, v_J, \pi_J)) \rightarrow \pi_{I \cup J}$

key properties:

1. can be computed w/o v
2. $|\pi_I| = |\pi_J| = |\pi_{I \cup J}| = p(\lambda)$

Fast opening via precomputation



Is it guaranteed?

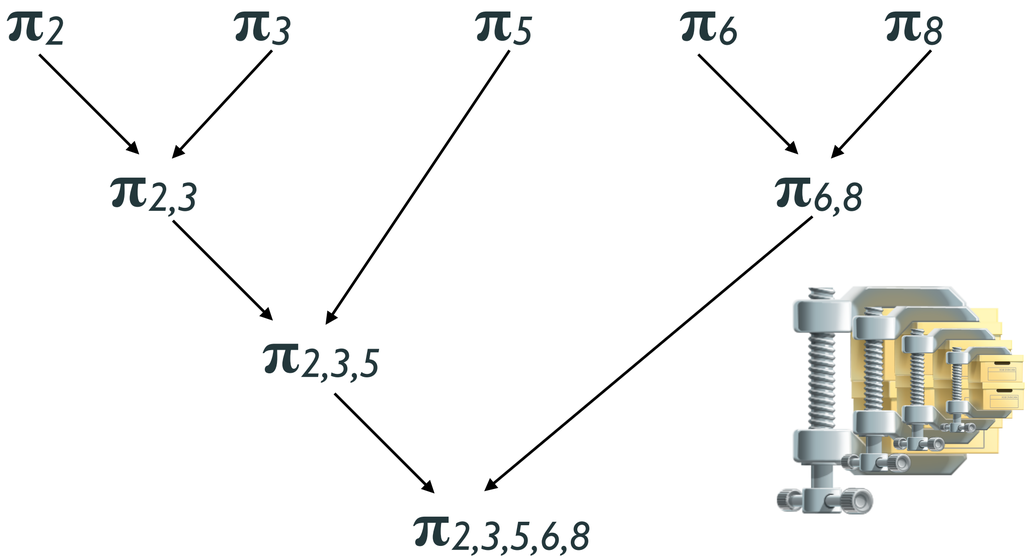
If Time(PrecompCom) = $O(n \log n)$, then

amortized opening time is $O(\log n)$ using $O(n\ |\pi|)$ storage

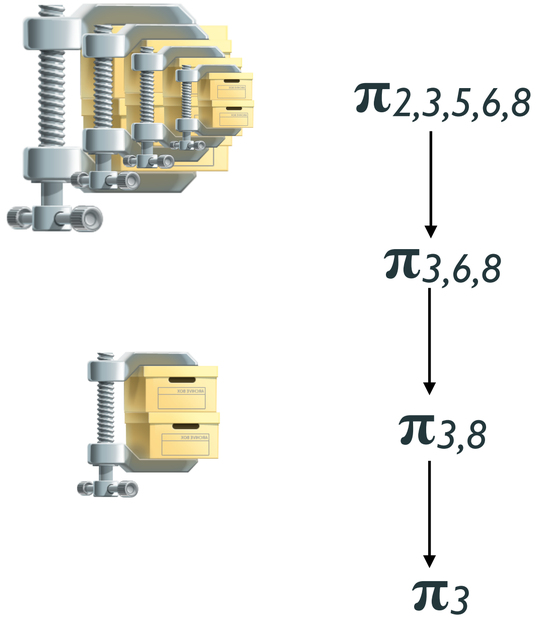
Can we do better?

Incremental aggregation [CFG+20]

unbounded aggregation



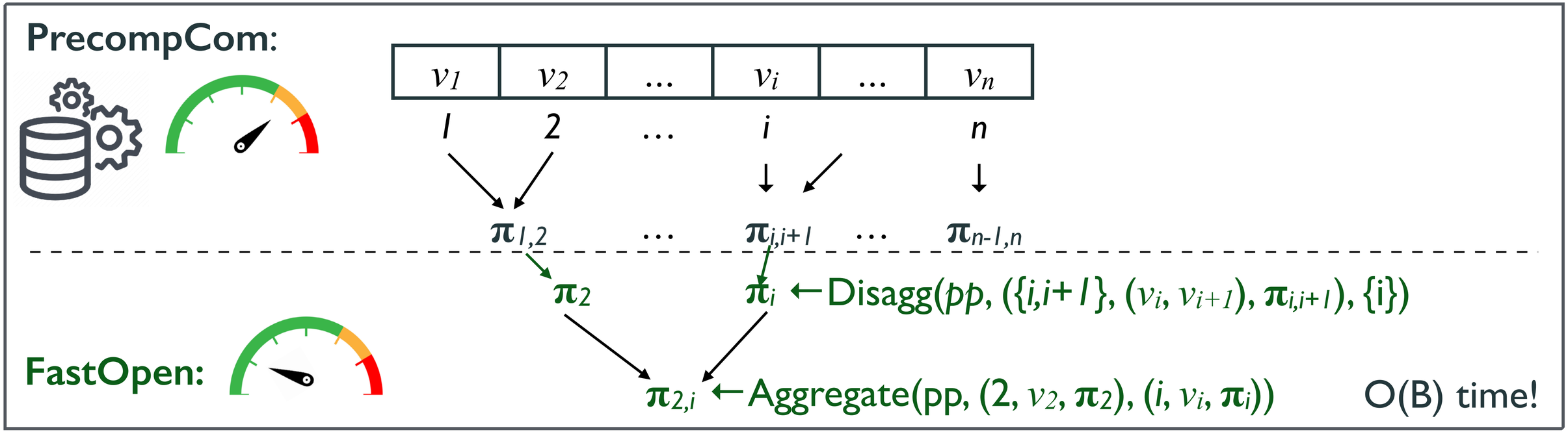
unbounded disaggregation



we can get even **more applications** from incremental aggregation

⇒ (fast) opening with precomputation

Better tradeoffs for opening with precomputation



Incremental aggregation → O(n $|\pi|$ / B) storage and O(B log n) opening

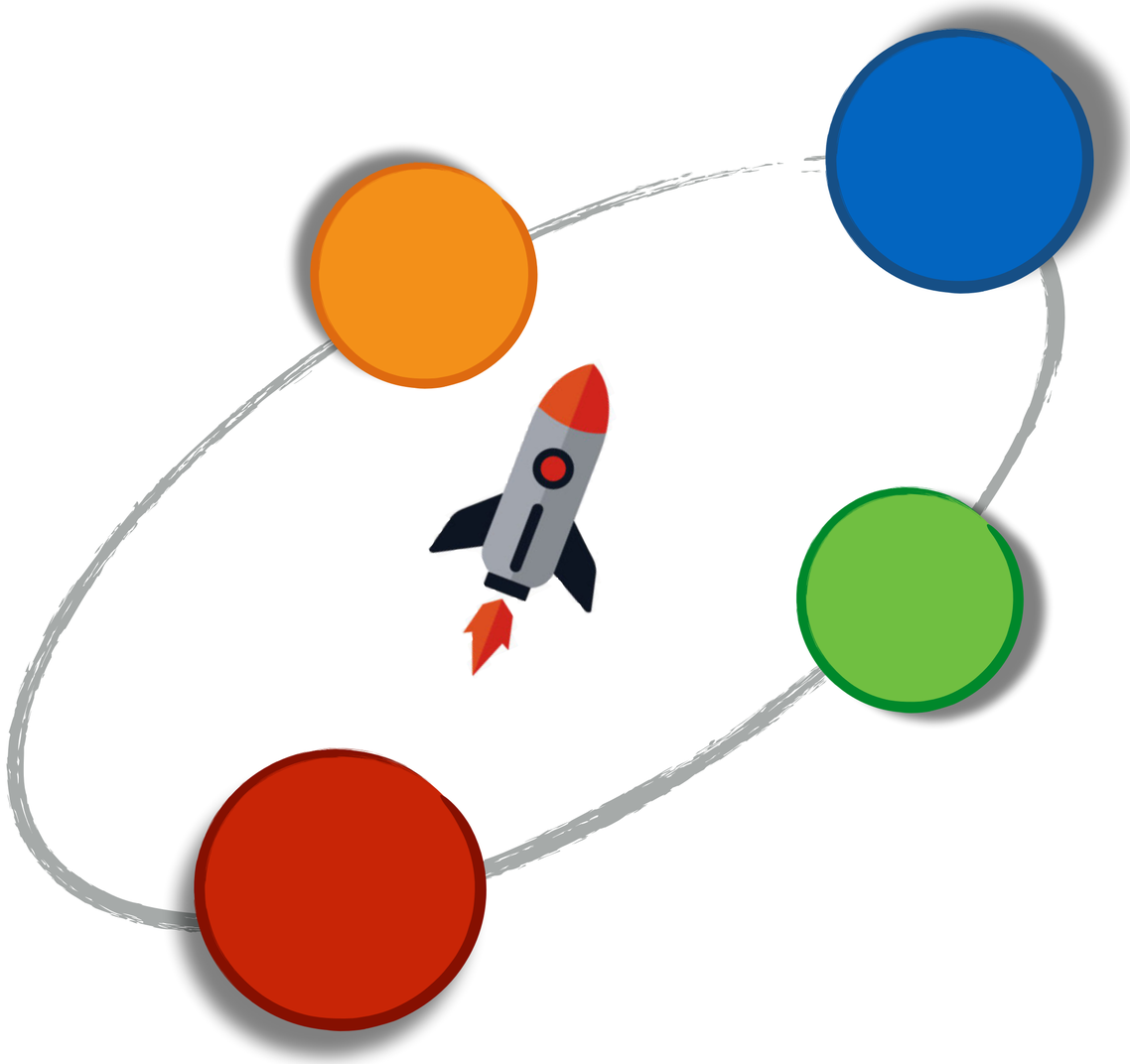
What's the guarantee that Time(PrecompCom) = O(n log n)?

Theorem (incremental agg. ⇒ quasilinear precomputation) [CFG+20]. Assume VC has incremental aggregation s.t. Aggregate (resp Disaggregate) on sets of size k takes time $\Theta(k)$. For any $B \leq n$:

Time(PrecompCom) = O(n log(n/B)) and T(FastOpen)=O(B m log m).

Our journey in vector commitments

- What are vector commitments?
- Theory → Practice
- Practice → Theory
- Updatability
- Subvector openings
- Aggregation
- **State of the art**
- New frontiers
- Conclusion and open problems



State of the art: many constructions, few planets

*considering only O(1)-openings
(except for MT & lattice-based VCs)

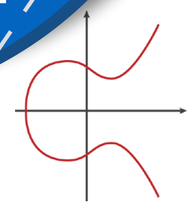
Vector Commitments

CRHFs
- Merkle trees

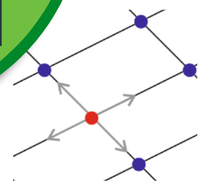
Groups of unknown order
- [CF13]-RSA
- [LM19]-RSA
- [BBF19]
- [CFG+20]-2
- [CFG+20]-1
- [AR20]

$|\mathbb{G}| = ?$

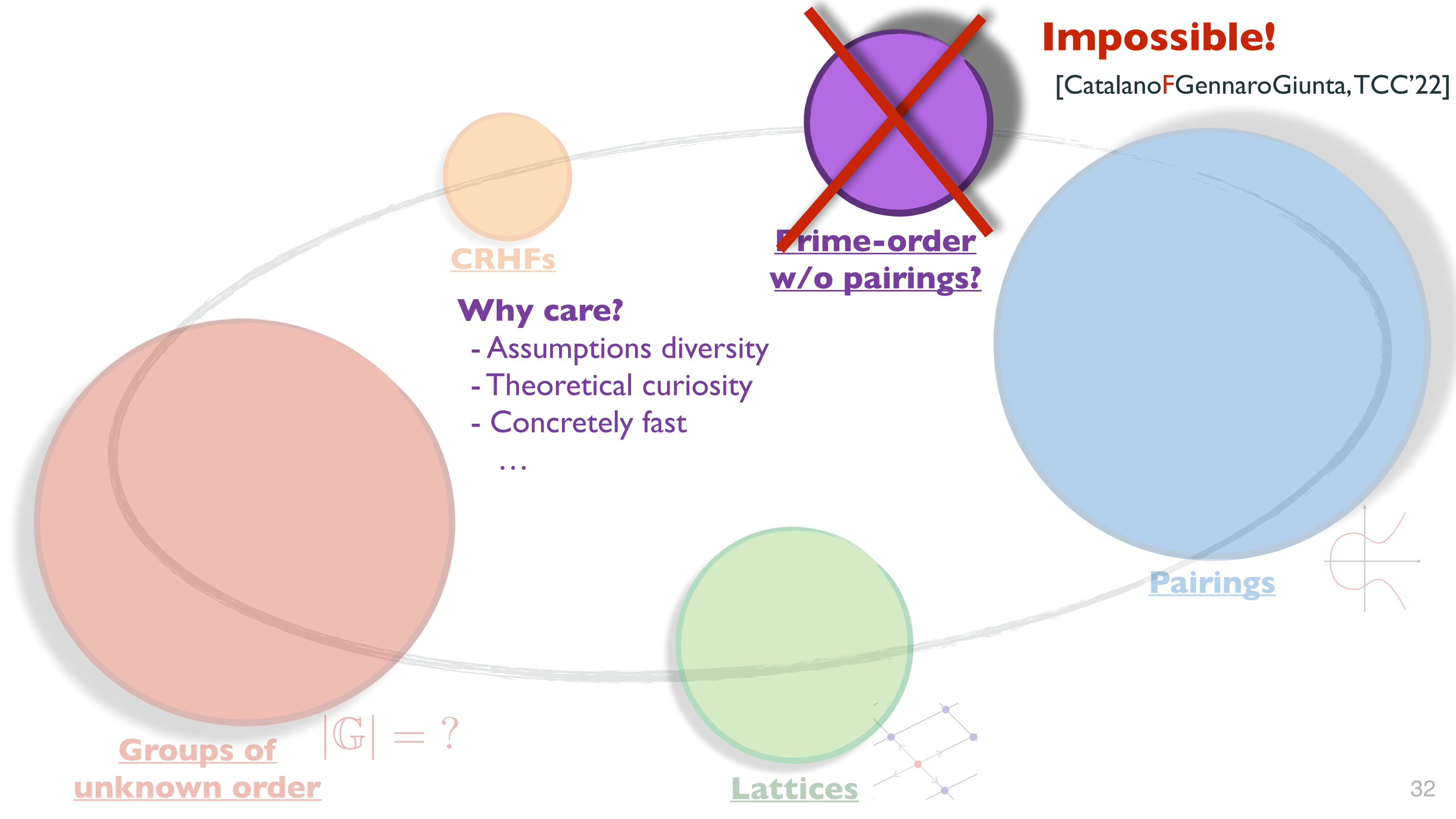
Pairings
- [KZG10]
- [LY10]
- [TAB+20]
- [GLWZ20]
- [CF13]-CDH
- [LM19]-CDH



Lattices
- [PSTY13]
- [PPS21]
- [ACL+22]



Legend:
- updatable, hint upd.
- subvector openings
- (incremental) aggregation


Impossible!
[CatalanoFGennaroGiunta, TCC'22]
CRHFs
Prime-order w/o pairings?
Why care?
- Assumptions diversity
- Theoretical curiosity
- Concretely fast
…
Pairings
Groups of unknown order
$|\mathbb{G}| = ?$
Lattices

State of the art and open problems

VC scheme	|pp|	|C|	|π|	Assumption	Updatable	SVC	Aggregation
Merkle trees	$O(1)$	$O(1)$	$O(\log n)$	CRHF	hint	✗	✗
[KZG10]	$O(n)$	$O(1)$	$O(1)$	n-SDH	yes	—	—
[LY10]	$O(n)$	$O(1)$	$O(1)$	n-DHE	yes	—	—
[CF13] (pairings)	$O(n^2)$	$O(1)$	$O(1)$	CDH	yes	—	—
[CF13] (RSA)	$O(n)$			RSA			
[PSTY13]	$O(\log^2 n)$	$O(\log n)$	$O(\log^3 n)$	SIS	yes	✗	✗
[LM19] (pairings)	$O(n^2)$	$O(1)$	$O(1)$	Cube-DH	yes	✓	—
[LM19] (RSA)	$O(n)$			SDPPR			
[BBF19]	$O(1)$	$O(1)$	$O(1)$	Strong RSA	hint	✓	—
[CFG+20] (acc)	$O(1)$	$O(1)$	$O(1)$	GGM	✗	✓	incremental
[CFG+20] (CF13)				SDPPR&Low-order	yes		
[AR20]	$O(1)$	$O(1)$	$O(1)$	Generalized RSA	yes	✓	1-hop
[GLWZ20]	$O(n)$	$O(1)$	$O(1)$	n-wBDHE*	yes	✓	1-hop cross-com
[TAB+20]	$O(n)$	$O(1)$	$O(1)$	n-SBDH	yes	✓	1-hop
[PPS21] (basic)	$O(n^2)$	$O(\log n)$	$O(\log n)$	SIS	yes	✗	✗
[PPS21] (tree)	$O(\log^2 n)$		$O(\log^3 n)$		yes		

— Do we have a best in class? —

Asymptotically yes: hidden-order-groups VCs have best combination of features

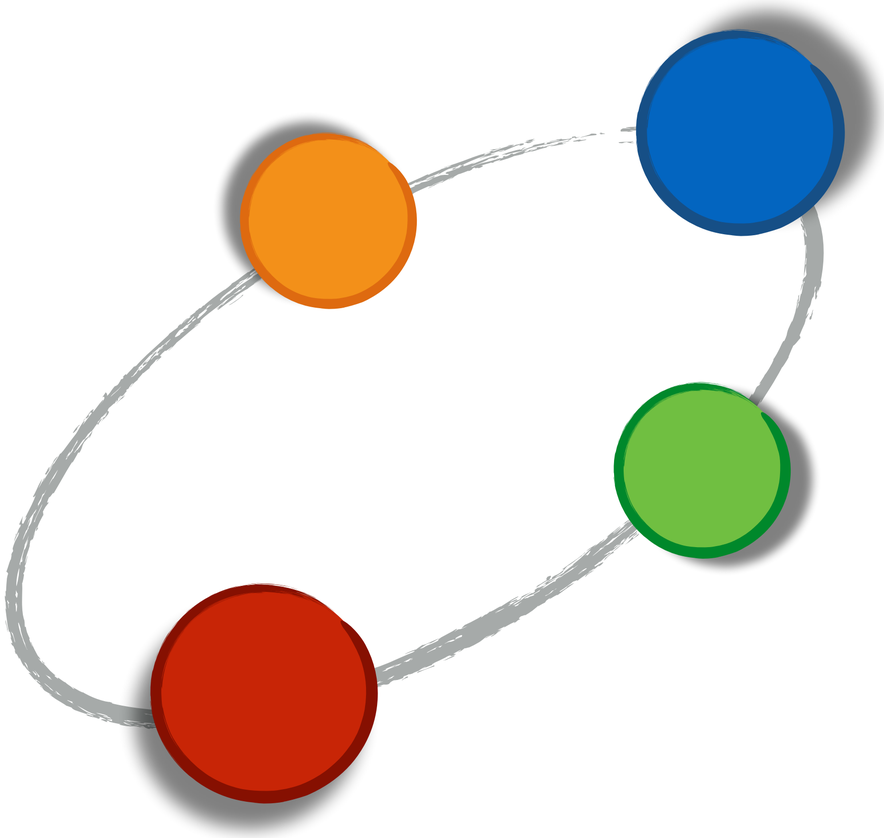
(concretely, less desirable: |C|=1 RSA group element ≈ 3000 bits)

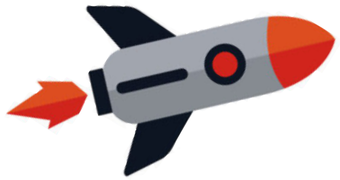
VCs in **prime-order** groups have large pp | **lattice-based** ones have large π

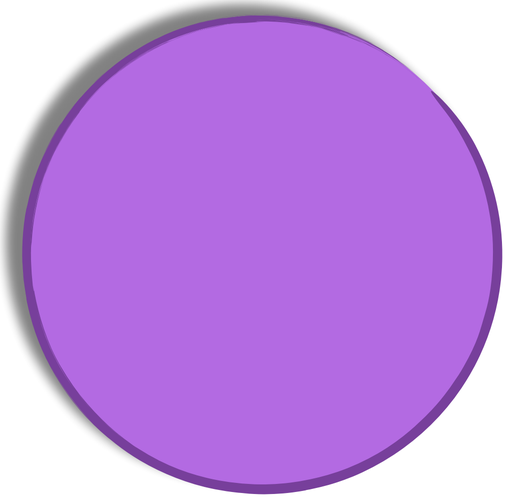
State of the art and open problems

VC scheme	|pp|	|C|	|π|	Assumption	Updatable	SVC	Aggregation
Merkle trees	$O(1)$	$O(1)$	$O(\log n)$	CRHF	hint	✗	✗
[KZG10]	$O(n)$	$O(1)$	$O(1)$	n-SDH	yes	—	—
[LY10]	$O(n)$	$O(1)$	$O(1)$	n-DHE	yes	—	—
[CF13] (pairings)	$O(n^2)$	$O(1)$	$O(1)$	CDH	yes	—	—
[CF13] (RSA)	$O(n)$			RSA			
[PSTY13]	$O(\log^2 n)$	$O(\log n)$	$O(\log^3 n)$	SIS	yes	✗	✗
[LM19] (pairings)	$O(n^2)$	$O(1)$	$O(1)$	Cube-DH	yes	✓	—
[LM19] (RSA)	$O(n)$			SDPPR			
[BBF19]	$O(1)$	$O(1)$	$O(1)$	Strong RSA	hint	✓	—
[CFG+20] (acc)	$O(1)$	$O(1)$	$O(1)$	GGM	✗	✓	incremental
[CFG+20] (CF13)				SDPPR&Low-order	hint		
[AR20]	$O(1)$	$O(1)$	$O(1)$	Generalized RSA	yes	✓	1-hop
[GLWZ20]	$O(n)$	$O(1)$	$O(1)$	n-wBDHE*	yes	✓	1-hop cross-com
[TAB+20]	$O(n)$	$O(1)$	$O(1)$	n-SBDH	yes	✓	1-hop
[PPS21] (basic)	$O(n^2)$	$O(\log n)$	$O(\log n)$	SIS	yes	✗	✗
[PPS21] (tree)	$O(\log^2 n)$		$O(\log^3 n)$		yes		
Some open problems:							
Short pp (1)	***o(n)***	$O(1)$	$O(1)$	**Prime-order groups?**	yes	✓	
Short C, π (2)		$O(1)$	$O(1)$	**Lattices?**			
Incremental aggr. (3)	$O(1)$			**Prime-order groups?**			incremental

A new frontier of vector commitments



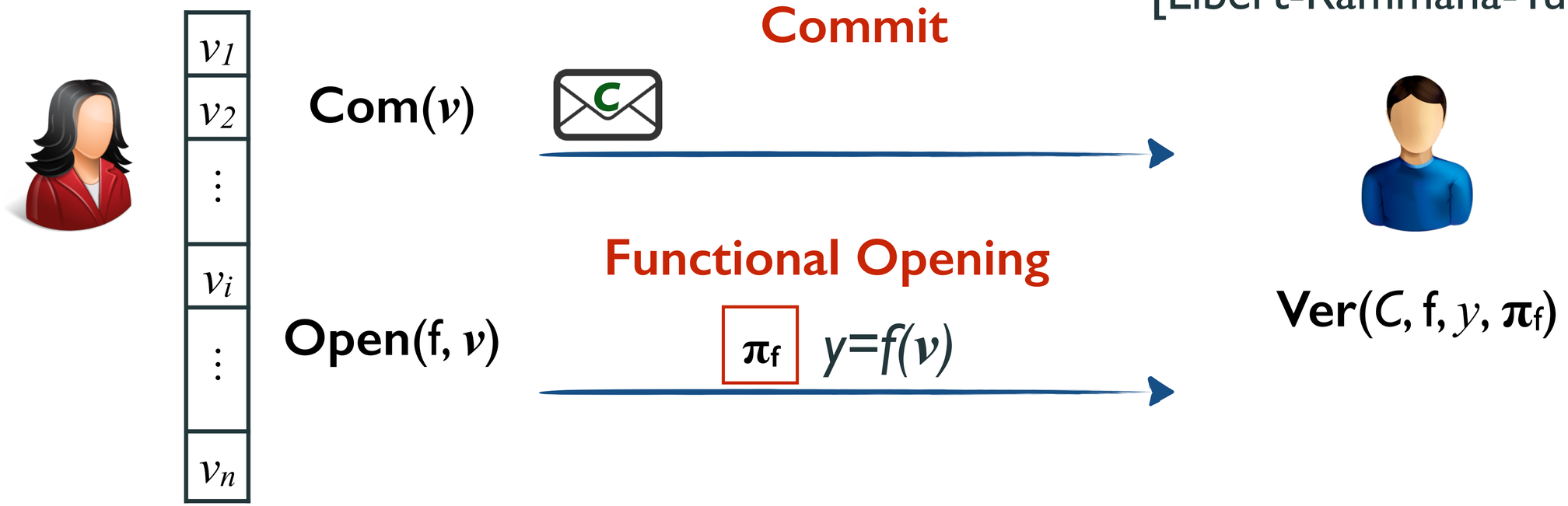




Functional
Commitments

Functional (Vector) Commitments

[Libert-Rammana-Yung, ICALP'16]



Basic idea: commit to a vector and open to functions of it

Evaluation binding: hard to open C to $y \neq y'$ for the same f

Conciseness: $|C|$ and $|\pi_f|$ short

State of the art in functional commitments

FC scheme	Functions $\mathcal{R}^n \to \mathcal{R}^\ell$	|pp|	|C|	|π|	Updatable
[LRY16, LM19]	linear maps	$O(n)$	$O(1)$	$O(1)$	yes
[LP20]	sparse polynomials	$O(m)$	$O(\ell)$	$O(1)$	no
[CFT22, ACL+22]	polynomials of deg $d=O(1)$	$O(n^{2d})$	$O(d)$	$O(d)$	yes
new [BCFL22]	circuits of bounded width w	$O(w^5)$	$O(1)$	$O(depth)$	yes
[dLP22, WW22]	circuits of bounded depth D				yes

➜ *constructions are still quite theoretical* e.g. $|pp| = O(n^5)$, $|\pi_f|=O(depth)$

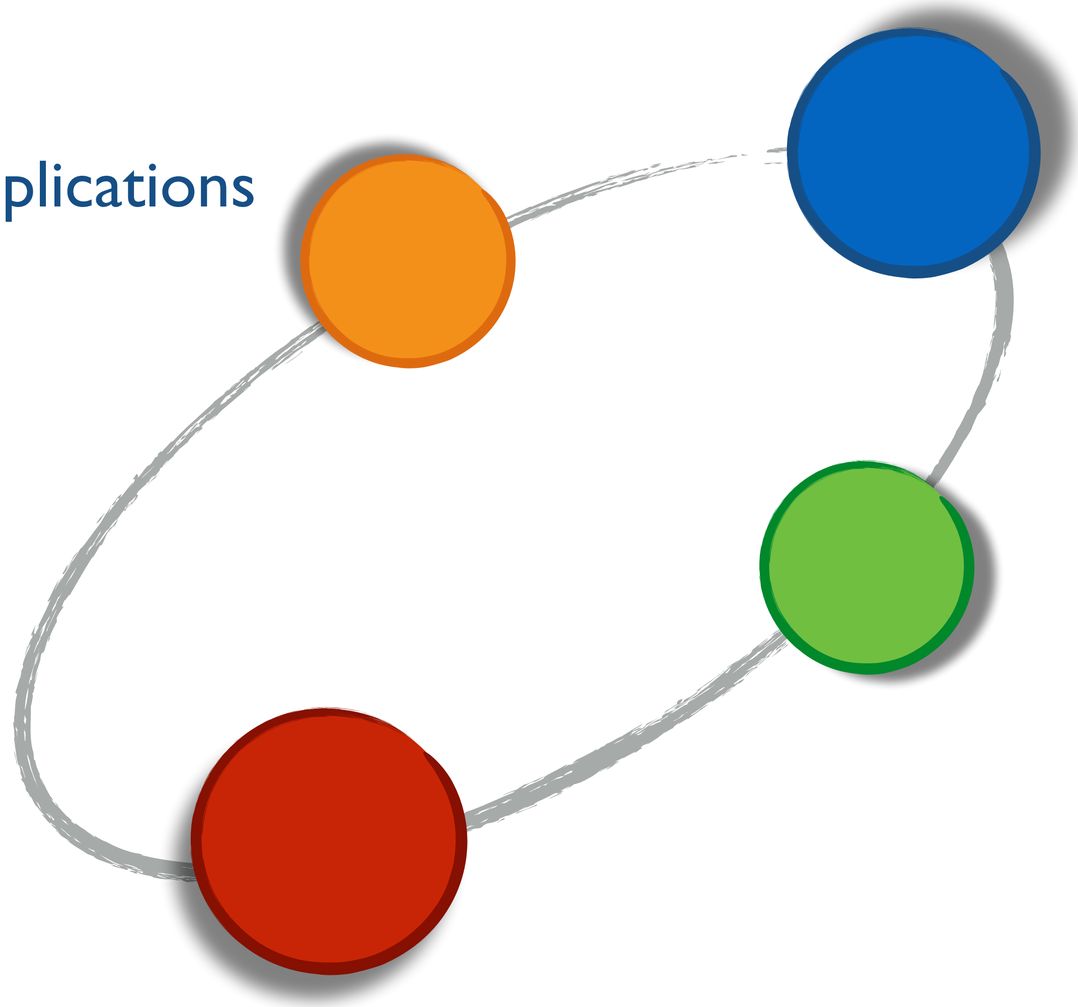
Some questions for future work: $|\pi_f|=O(1)$ and/or $|pp| = O(n)$?

Conclusions

- Useful primitive for compressing information in a verifiable way

  Conciseness is the key

- **Past and present:** nice interplay theory↔applications
- **Future:** exciting potential of functional VCs!



Questions