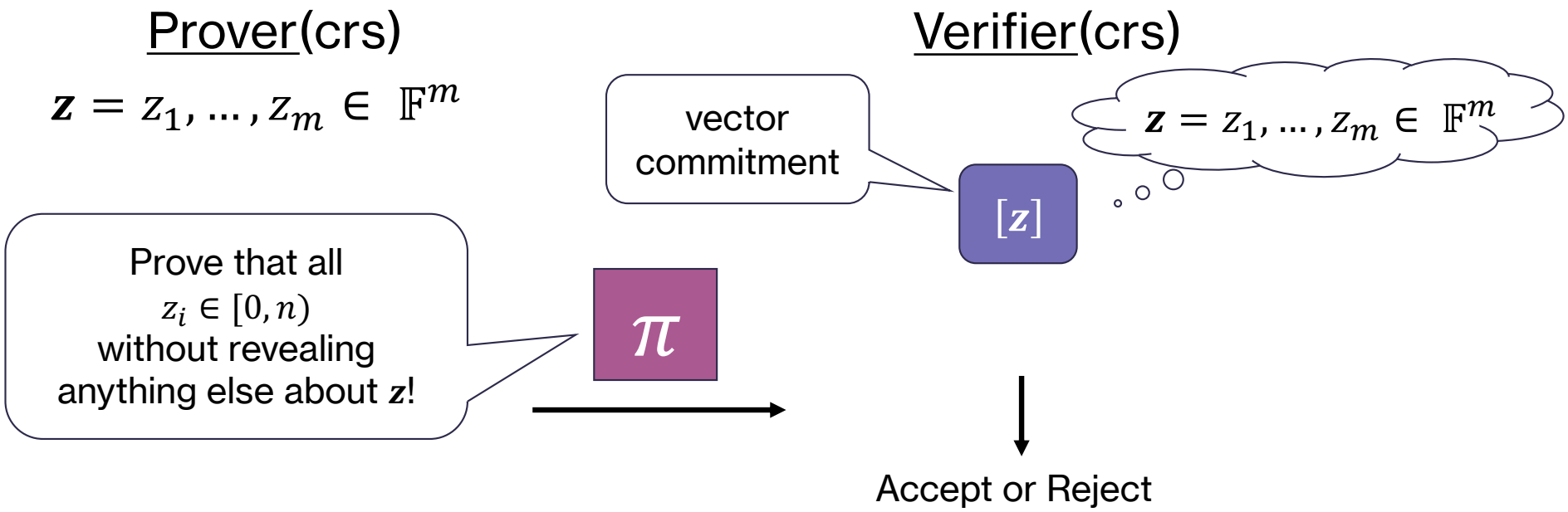
DEKARTPROOF: EFFICIENT ZK VECTOR RANGE PROOFS AND THEIR APPLICATIONS

Dan Boneh, **Trisha Datta**, Rex Fernando, Wicher Malten, Kamilla Nazirkhanova, Alin Tomescu

Stanford University, Aptos Labs

ZK Vector Range Proofs (ZKVRPs)



ZK Vector Range Proof (ZKVRP) Applications

- Publicly verifiable secret sharing (PVSS) schemes for field elements
  - Dealer encrypts secret shares and proves encrypted shares are “legitimate”
- PVSS used for threshold cryptography in distributed key generation
- PVSS for field elements often uses additively homomorphic ElGamal encryption

$$Encrypt\left(pk = g^{sk}, m, r\right) \rightarrow (g^m pk^r, g^r)$$

$$Decrypt\left((ct_1, ct_2), sk\right): DLOG(ct_1 / ct_2^{sk}) \rightarrow m$$

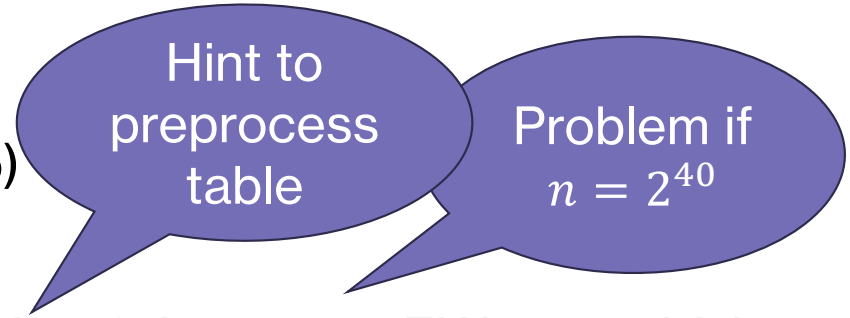
  - Decryption involves solving discrete log → proving “legitimacy” means proving plaintexts are small

Range size here is 2^{40}!

- Other applications: confidential transactions, electronic voting

ZK Vector Range Proof (ZKVRP) Past Work

- Proving m values are in $[0, n)$:
  - Lookup arguments (e.g., cq, Lasso)
    - Very fast prover time
    - Provers must store preprocessing hint of size $O(n)$ or ZK is non-trivial
  - Bulletproofs: small proof size, fast prover, slower verifier
  - MissileProof: small proof size, fast verifier, slower prover
- DekartProof (our work): no hint, provably ZK, faster verifier than Bulletproofs, faster prover than MissileProof (asymptotically) and Bulletproofs (empirically)



ZK Vector Range Proofs (ZKVRP)

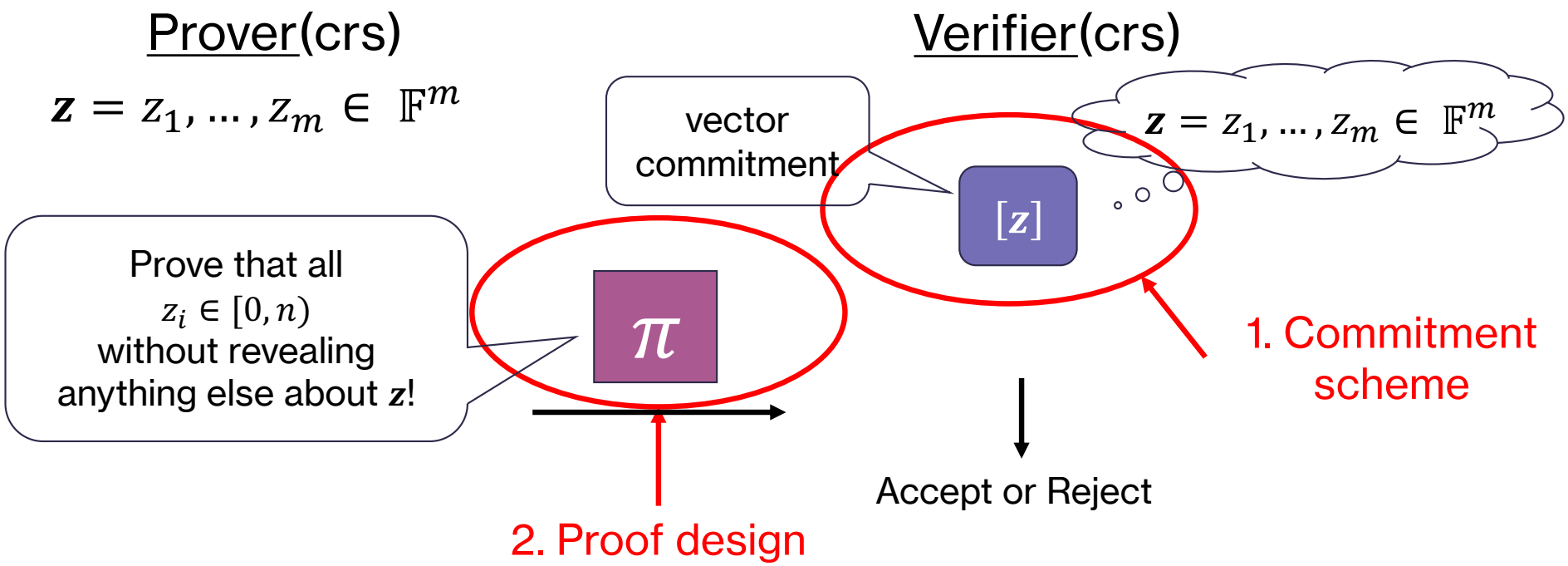
Need proof system for the following relation:

$$\mathcal{R}_{n,m} = \{ (com; \boldsymbol{z} \in \mathbb{F}^m) : com = Commit(\boldsymbol{z}), \boldsymbol{z} \in [0, n)^m \}$$

with the following properties (informal):

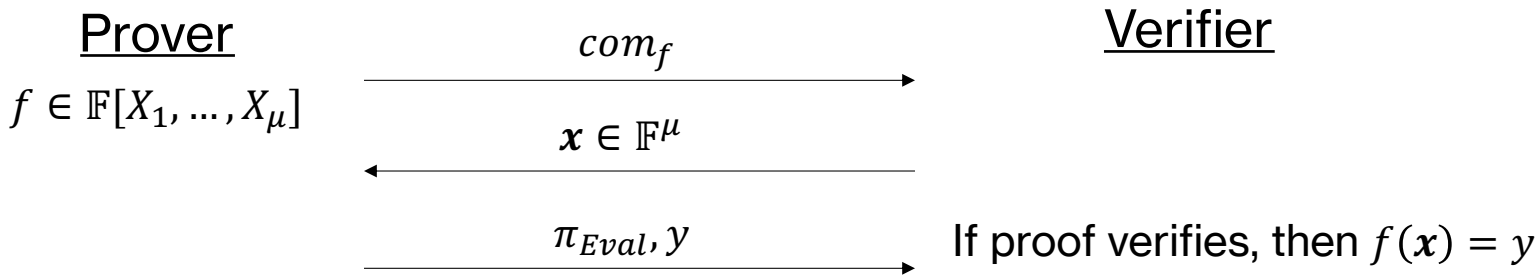
- Complete: honest provers succeed
- Sound: dishonest provers fail
- Zero-Knowledge: verifier only learns values are in the range and nothing more

ZKVRP Design Strategy



Polynomial Commitment Schemes (informal)

- Allows prover to “commit” to a polynomial and then prove its evaluations at certain points

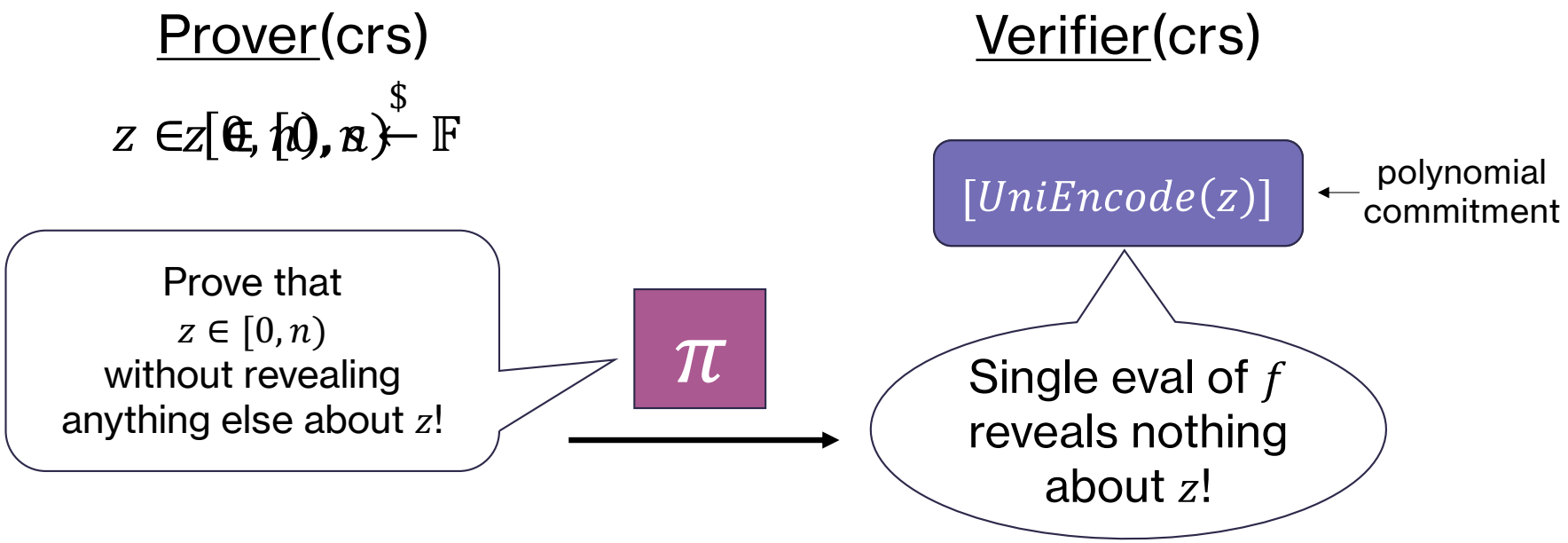


- Some PCSs are “homomorphic”

$$Commit(f) + Commit(f') = Commit(f + f')$$

- Why is this useful?
  - We will see that we can commit to vectors by committing to polynomials

Borgeaud's Range Proof for Single Value[1]



[1]https://solvable.group/posts/membership-proofs-from-polynomial-commitments/

Borgeaud's Range Proof for Single Value[1]

Prover(crs)

$z \in [0, n)$

1. Call $z_1, \ldots, z_{\log_2(n)}$ the bit decomposition of z

2. Send verifier commitments to each z_i

Verifier(crs)

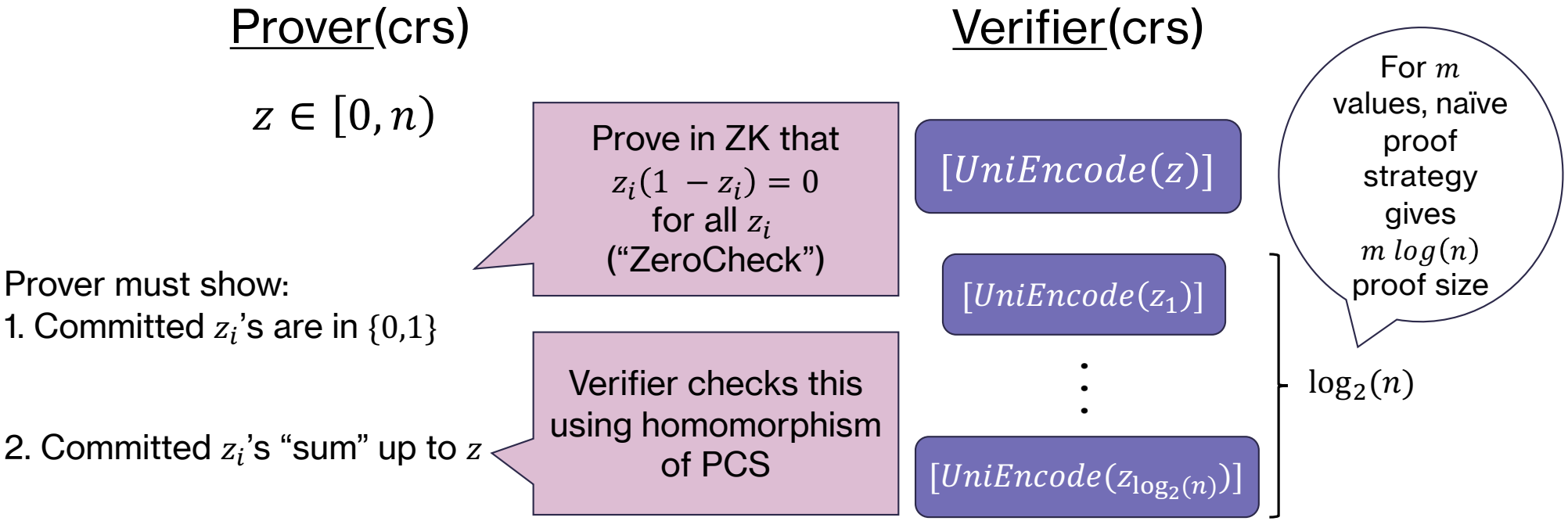
$[UniEncode(z)]$

$[UniEncode(z_1)]$

$\vdots$

$[UniEncode(z_{\log_2(n)})]$

Borgeaud's Range Proof for Single Value[1]



This scheme is complete, sound, and zero-knowledge!
...but it is pretty inefficient

How to Batch Borgeaud Efficiently

- Where does inefficiency come from in Borgeaud?
  - Each commitment encodes only one value
- How to address this inefficiency?
  - Encode more than one value in each commitment
- How to encode more than one value?
  - Replace univariate polynomials with multilinear polynomials → leads to prover time savings

Multilinear Encoding of Vector

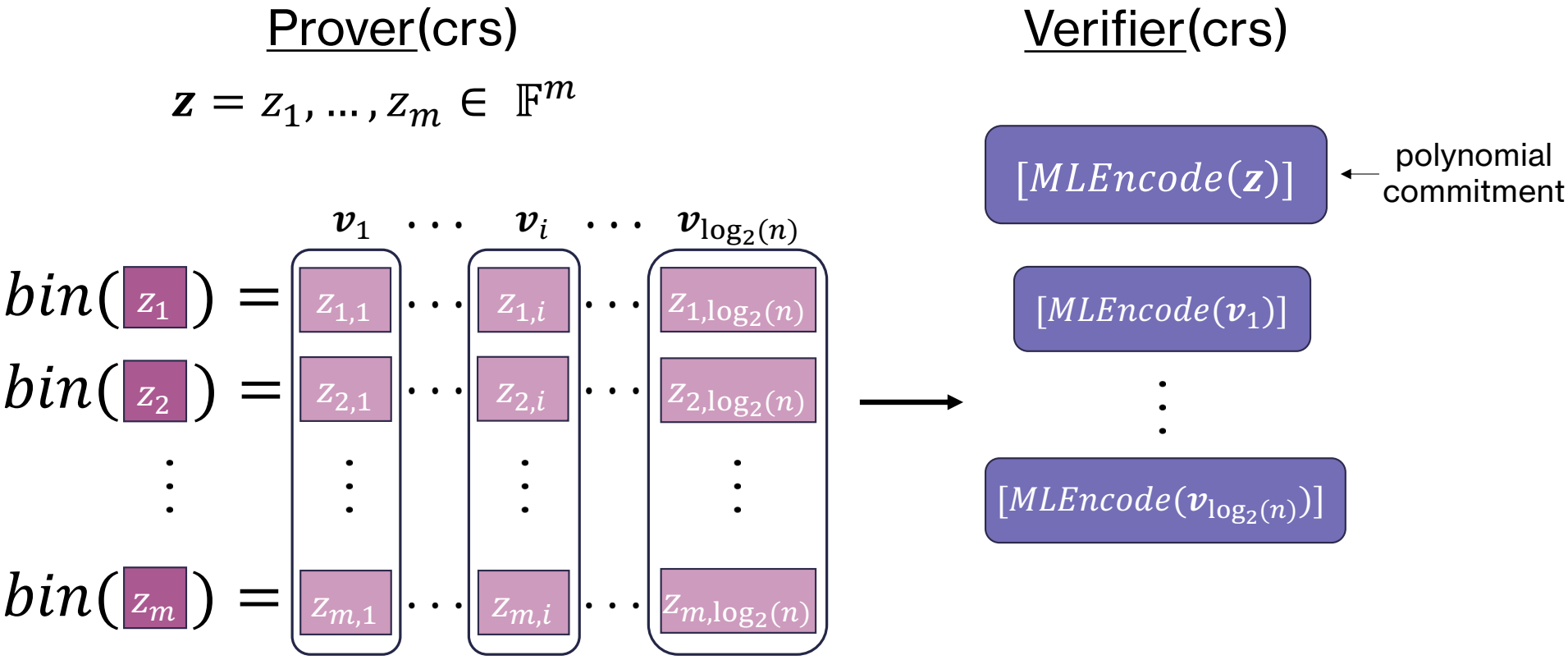
- To do ZKVRP for m values, we use μ-dimensional hypercube where $\mu = \log_2(m+1)$
  - Assume for simplicity $m+1$ is a power of 2
  - Let $bin(i) \in \{0,1\}^{\mu}$ be the μ-dimensional binary representation of i
- For $\boldsymbol{z} \in \mathbb{F}^m$, let f be the unique μ-variate multilinear polynomial where:

$$f\big(bin(0)\big) = s \xleftarrow{\$} \mathbb{F}$$

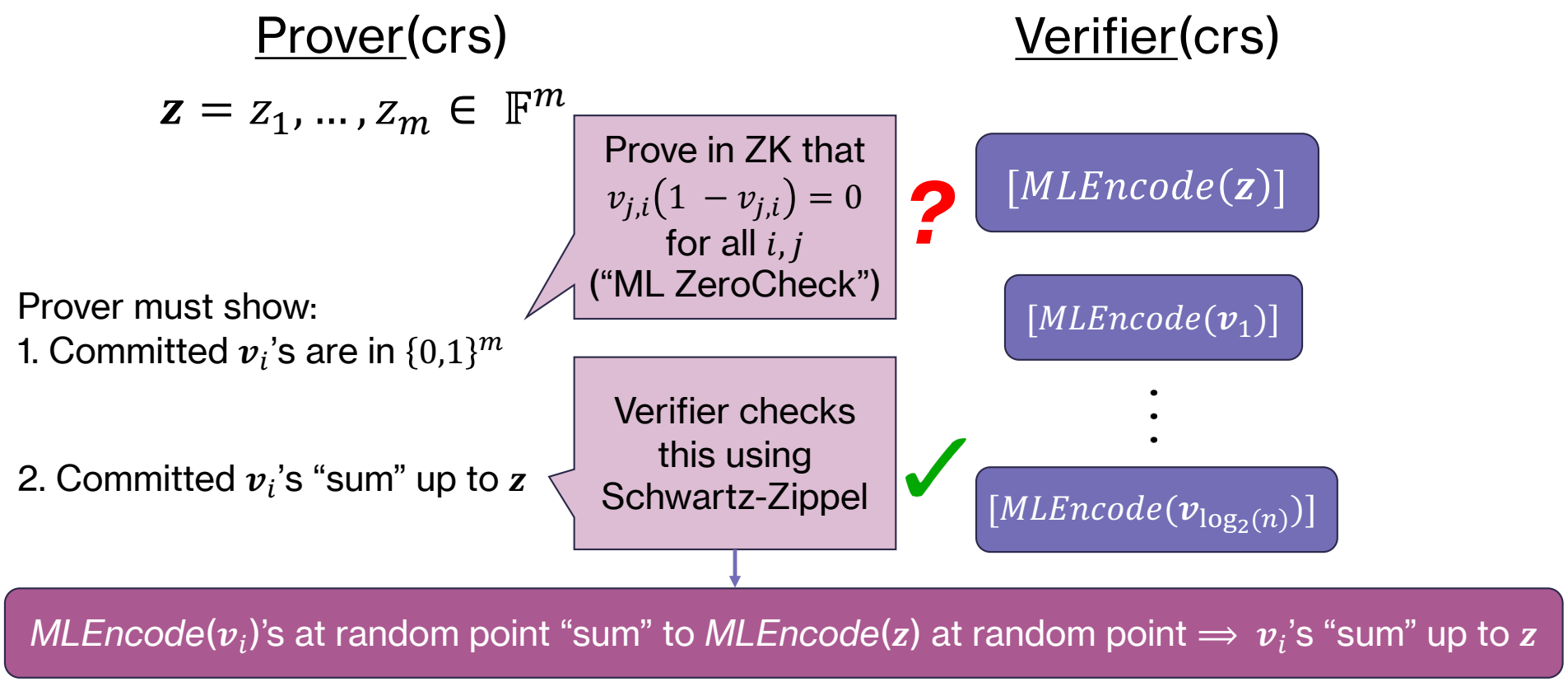
$$f\big(bin(i)\big) = z_i \text{ for } i = 1, \dots, m$$

- Let $MLEncode(\boldsymbol{z}) \in \mathbb{F}^{\leq 1}\big(X_1, \dots, X_\mu\big)$ denote this f
  - A single evaluation of f reveals nothing about $\boldsymbol{z}$

Batching Borgeaud's Range Proof



Batching Borgeaud's Range Proof



ZK ZeroCheck for Multilinear Polynomials

- ZeroCheck for multilinear polynomials can be performed with sum-check → but sum-check is not zero-knowledge!
- Prior ZK sum-checks in literature, but they are not performant or not fully ZK
- So we design a new zero-knowledge sum-check!

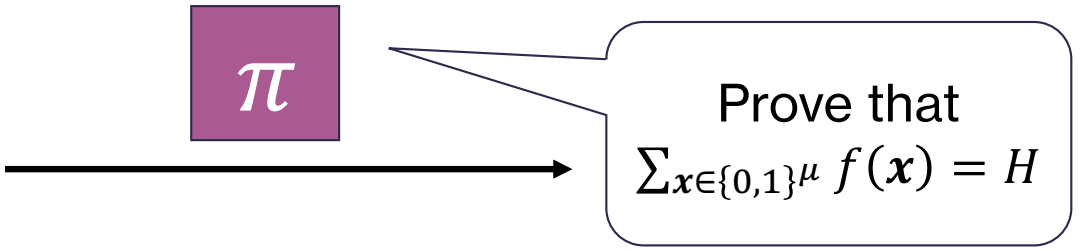
Sum-Check is not ZK

Prover

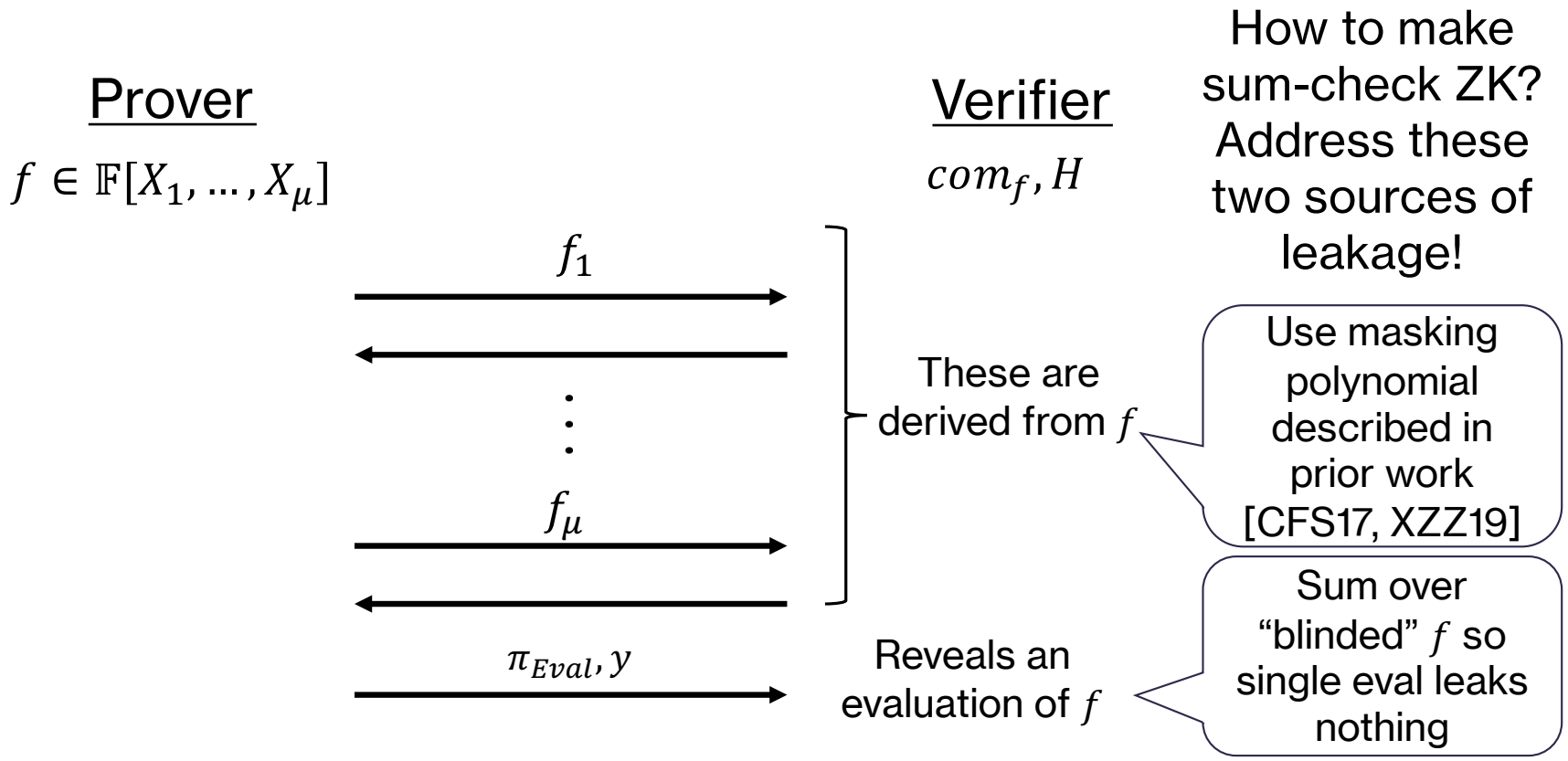
$f \in \mathbb{F}[X_1, \ldots, X_\mu]$

Verifier

com_f, H



Sum-Check is not ZK



Proving ZK Sum-Check is Sound and ZK

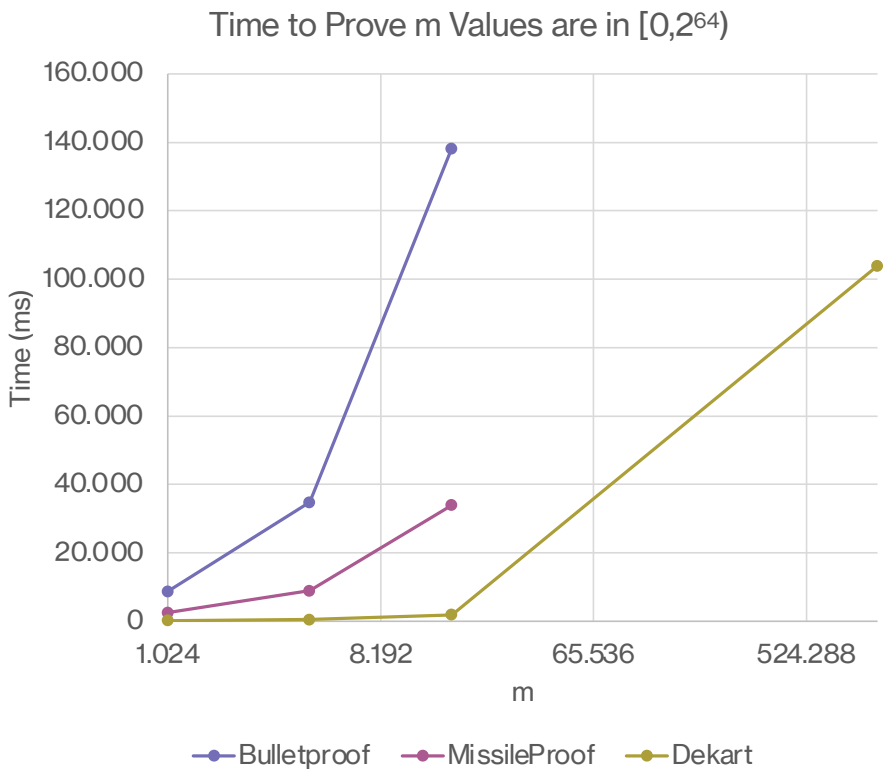
- Normal polynomial interactive oracle proofs cannot capture our construction
  - E.g., our verifier must query sum of polynomial oracles
- We introduce a new information theoretic proof system called

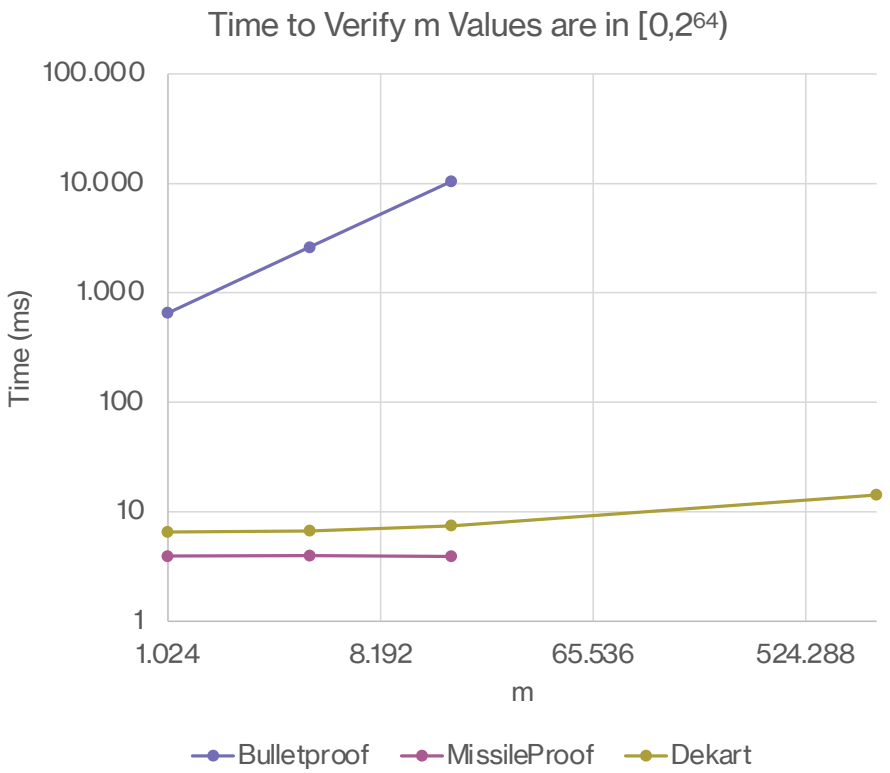
  Homomorphic Polynomial Interactive Oracle Proofs

  - A model for instantiating polynomial interactive oracle proofs with homomorphic PCS
  - We give compilation theorem with round-by-round knowledge soundness proof
  - See paper for details

Experimental Comparisons

- Dekart has very fast prover (>10x faster) and relatively fast verifier (only 2x slower)

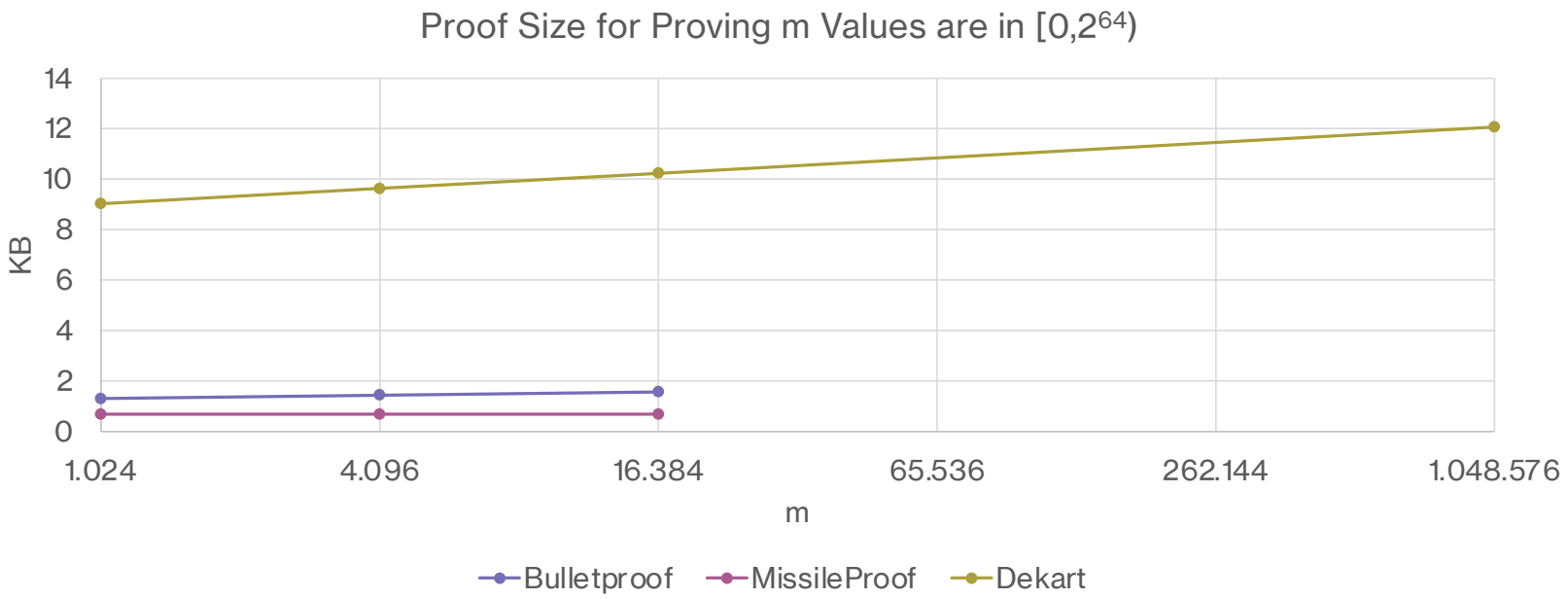




Experimental Comparisons

- BUT Dekart has much larger proof size

→ Could be desirable tradeoff in some situations (e.g., PVSS)



Conclusions

- ZKVRPs are very useful in several blockchain applications (PVSS for field elements, confidential transactions, electronic voting, etc.)
- We introduce DekartProof, a new ZKVRP with efficient prover time, efficient verifier time, and short proofs
- As a building block, we introduce a ZK sum-check and prove its security

- Open question: efficient ZKVRPs in the lattice setting