ZK-IOPs Approaching Witness Length

Noga Ron-Zewi

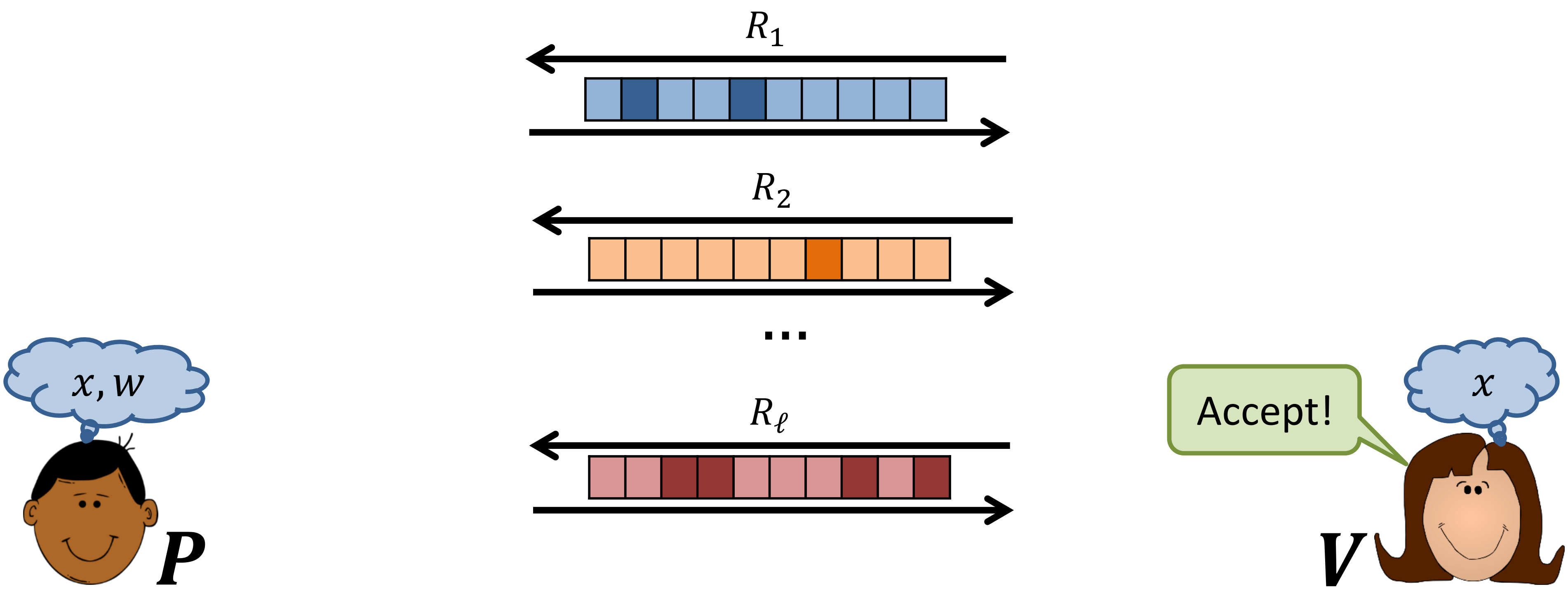
University of Haifa

Mor Weiss

Bar-Ilan University

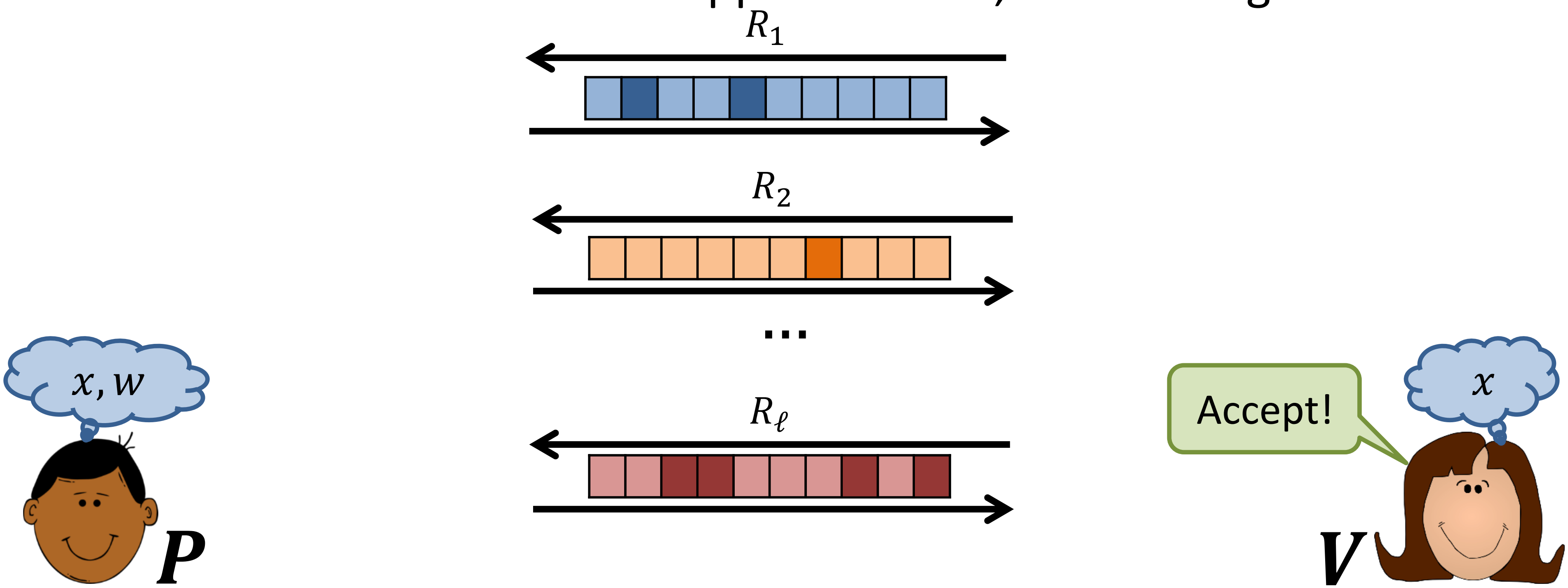
Interactive Oracle Proofs (IOPs) [BCS16,RRR17]

- **Goal:** prove that $x \in L$
- $x \in L \quad \Rightarrow \quad$ verifier accepts whp
- $x \notin L \quad \Rightarrow \quad$ for any P^*, verifier rejects whp



Interactive Oracle Proofs (IOPs) [BCS16,RRR17]

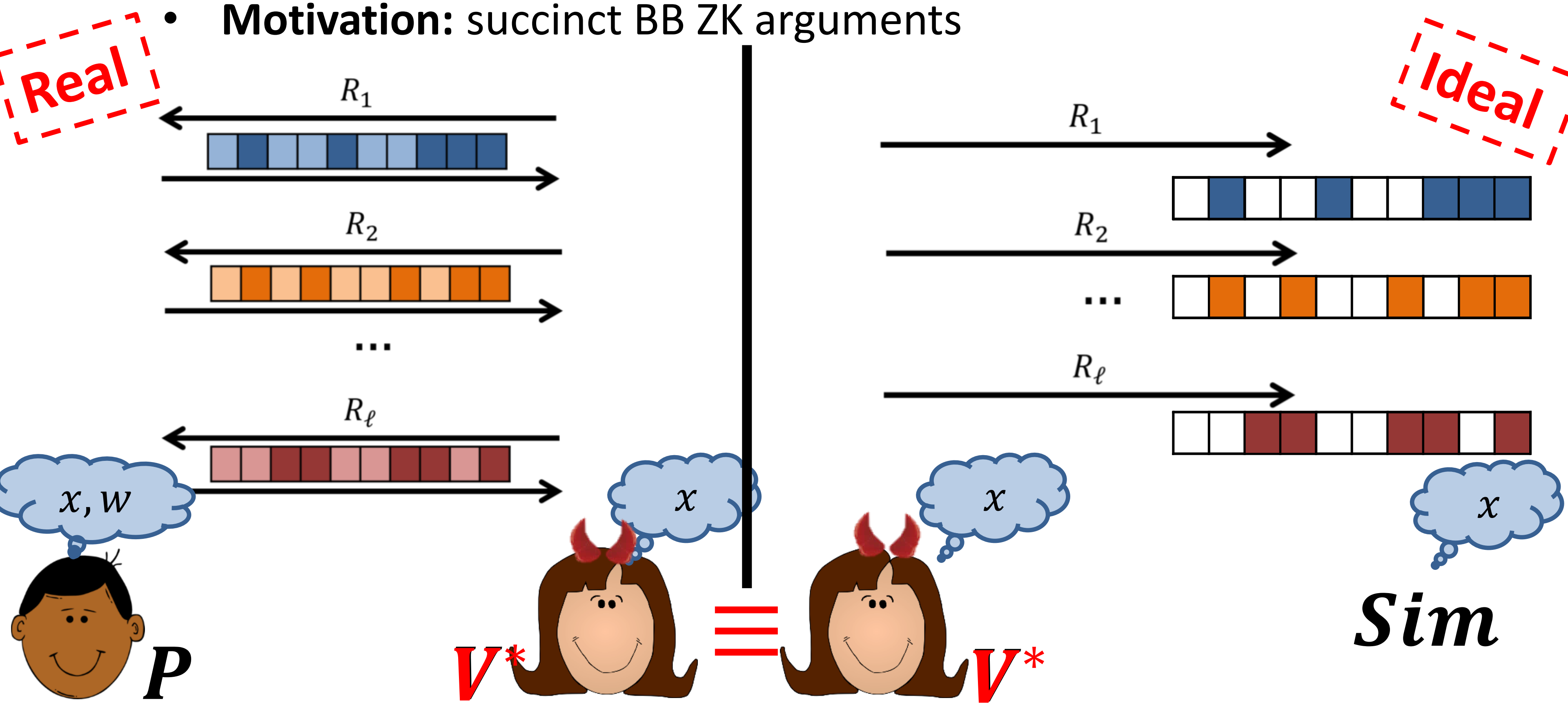
- **Goal:** prove that $x \in L$
- $x \in L \quad \Rightarrow \quad$ verifier accepts whp
- $x \notin L \quad \Rightarrow \quad$ for any P^*, verifier rejects whp
- *Probabilistically Checkable Proofs (PCPs)* [ALMSS92,AS92]: no verifier messages, one oracle from P
- **Motivation:** hardness of approximation, succinct arguments



Zero-Knowledge IOPs

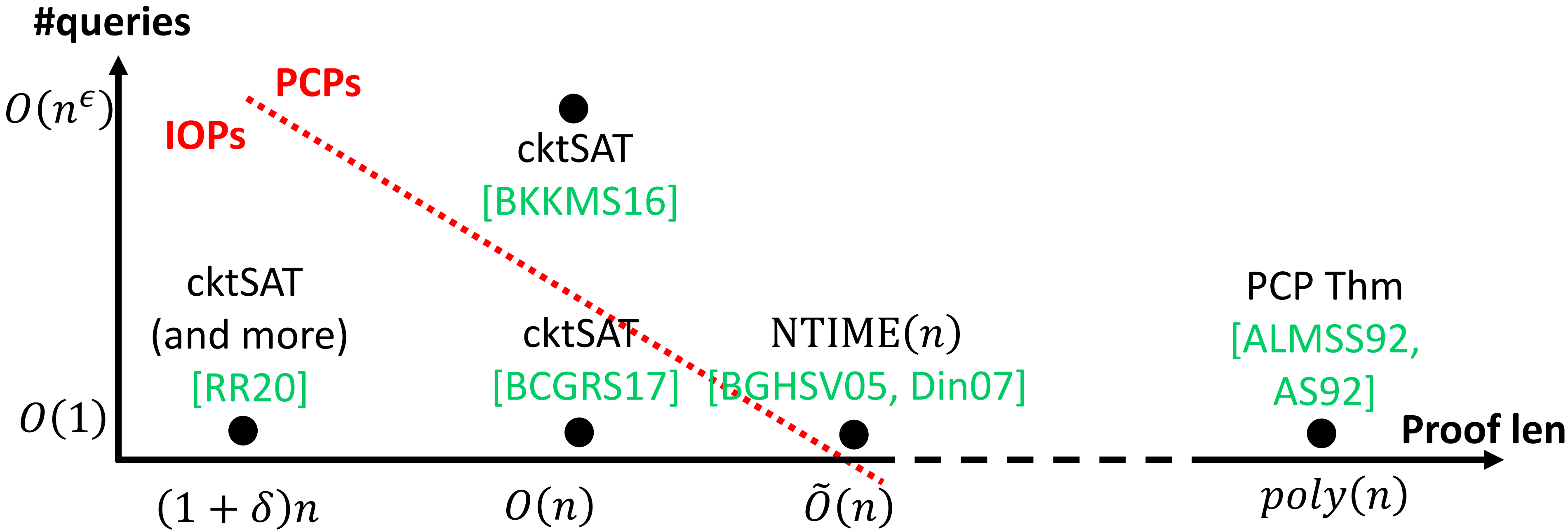
Verifier learns only that $x \in L$

- Proof generation (necessarily) randomized
- Cannot get ZK against any PPT verifier (as in standard ZK proofs): V^* can read entire oracles
- Instead, i.t. ZK against ***query restricted*** V^* (t-restricted verifier)
- **Motivation:** succinct BB ZK arguments



Proof Length in PCPs and IOPs

- **Large body of works on reducing PCP/IOP length**
  - Proof length: total length of all prover messages
  - n is instance length
- IOPs beat SotA PCP construction, overcome known limitations*



*Omitting many other works

Proof Length in **Zero-Knowledge** PCPs and IOPs

- First **ZK-PCPs** in late 1990s [KPT97]
- After ~30 years of research, still **no linear-length ZK-PCPs for NP**
- In particular, no "best of all worlds" ZK-PCPs
  - (Large) polynomial proof length [KPT97,IW14,IWY16]
  - Large query complexity of honest verifier [IKOS07,HVW21]
  - Inefficient honest prover [GOS24]
  - Adaptive honest verification [KPT97,IW14]
  - Inefficient ZK simulation [IWY16]
  - All constructions (except [GOS24]) eliminate algebraic structure of underlying (non-ZK) PCP
- **ZK-IOP** constructions significantly better than SotA ZK-PCPs: for n^{ϵ}-ZK
  - 2-round $\tilde{O}(n)$-length for $\mathrm{NTIME}(n)$, honest verifier makes $poly \log n$ queries [BCGV16,BCFGRS17,BBHR19,CHMMVW20]
  - $\boldsymbol{O(n)}$**-length** for R1CS over large $\mathbb{F}$ [BCRSVW19] (even with $O(n)$-time P [BCL22])
  - Constructions are algebraic in nature (similar to standard PCPs)
- ***No ZK-IOPs approaching witness length*** (even with ***honest-verifier*** ZK)

This Work: ZK-IOPs approaching Witness Len

- **ZK-IOPs approaching witness length:**
  for every constant $\delta > 0$ 3SAT has ZK-IOP of length $(1+\delta)m$
  - m is witness length (length of satisfying assignment)
  - m^{ϵ}-ZK, constant $\epsilon > 0$ depends on δ
  - $O(1)$ queries and rounds, constant soundness error
  - Sublinear-time verification (+ $poly(m)$ local preprocessing), prover runs in poly time
- Previously:
  - **Shortest IOPs:** IOPs approaching witness length [RR20], ***no ZK***
    - [RR20]'s IOPs for large class of languages, we focus on 3SAT
  - **Shortest ZK-IOPs:** $O(n)$-length ZK-IOPs [BCRSVW19,BCL22]
    - $(1+\delta)m$ vs. $O(n)$
    - n is ***instance length***, in worst case $n = m^3$
- **New constructions of main building blocks**
  - Strong **ZK properties** of general **tensor codes**
  - Sublinear-CC **ZK sumcheck** protocol for general tensor codes
  - Not just a means to an end!

How to Construct (ZK) IOPs

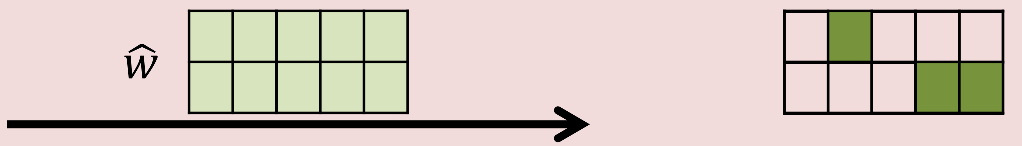
IOPs for 3SAT: Blueprint

Goal: Prove that $\varphi \in 3SAT$

Encoding (Arithmetization)

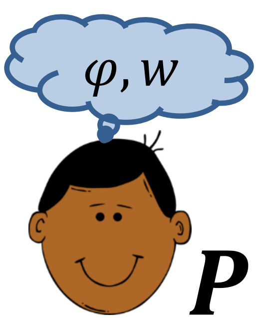
$\hat{w} := C(w)$

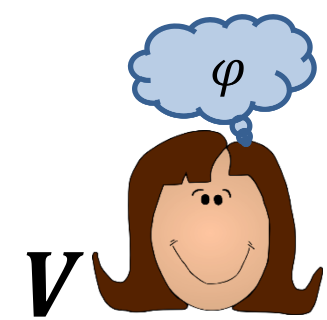
Usually, C is Low-Degree Extension (LDE)



Check $\hat{w} \in C$

Verification





IOPs for 3SAT

Approaching Witness Length [RR20]

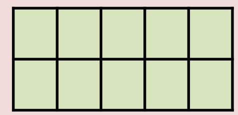
Goal: Prove that $\varphi \in 3SAT$

Encoding (Arithmetization)

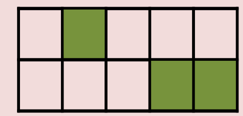
$c := C(w)$

~~Usually, C is Low-Degree Extension (LDE)~~

c



C is high-rate encoding



Check $c \in C$

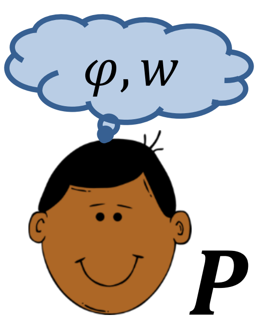
Verification

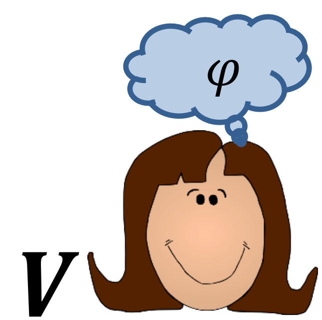
Sublinear CC

Σ☑

on $\hat{w}$: verify w satisfies φ

"code switching":
Emulate sumcheck on $\hat{w}$ ***using c***





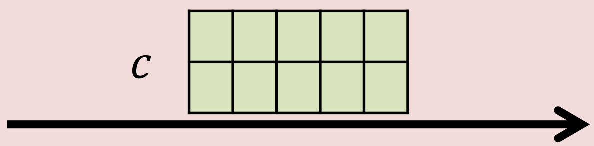
Zero-Knowledge IOPs for 3SAT: Blueprint

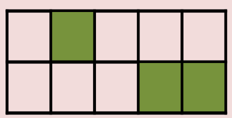
Goal: Prove that $\varphi \in 3SAT$

Encoding (Arithmetization)

$c \coloneqq C(w)$

C is high-rate **zero-knowledge** code





Check $c \in C$

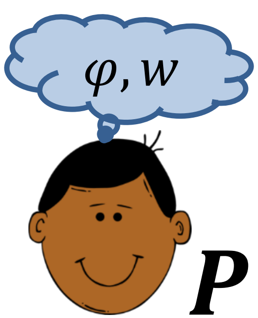
Verification

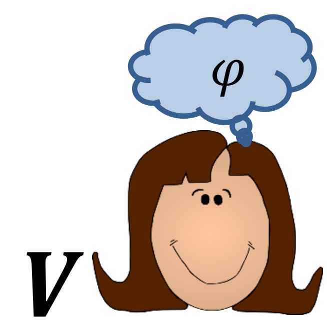
zero-knowledge

Sublinear CC

Σ ☑

on $\hat{w}$ using c: verify w satisfies φ





Zero-Knowledge IOPs for 3SAT: Blueprint

Goal: Prove that $\varphi \in 3SAT$

Encoding (Arithmetization)

$c := C(w)$

C is high-rate
zero-knowledge code

c

Check $c \in C$

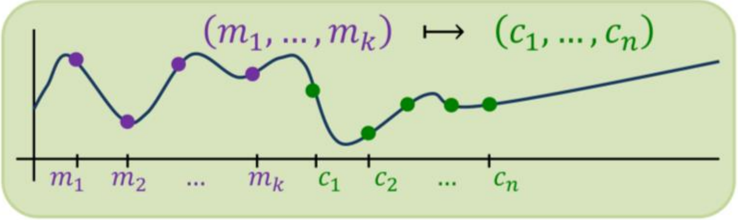
Verification

zero-knowledge
Sublinear CC
Σ ☑
on $\hat{w}$ using c: verify
w satisfies φ

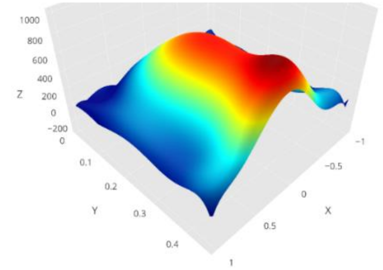
Tensors of Zero-Knowledge Codes

Tensor Codes 101

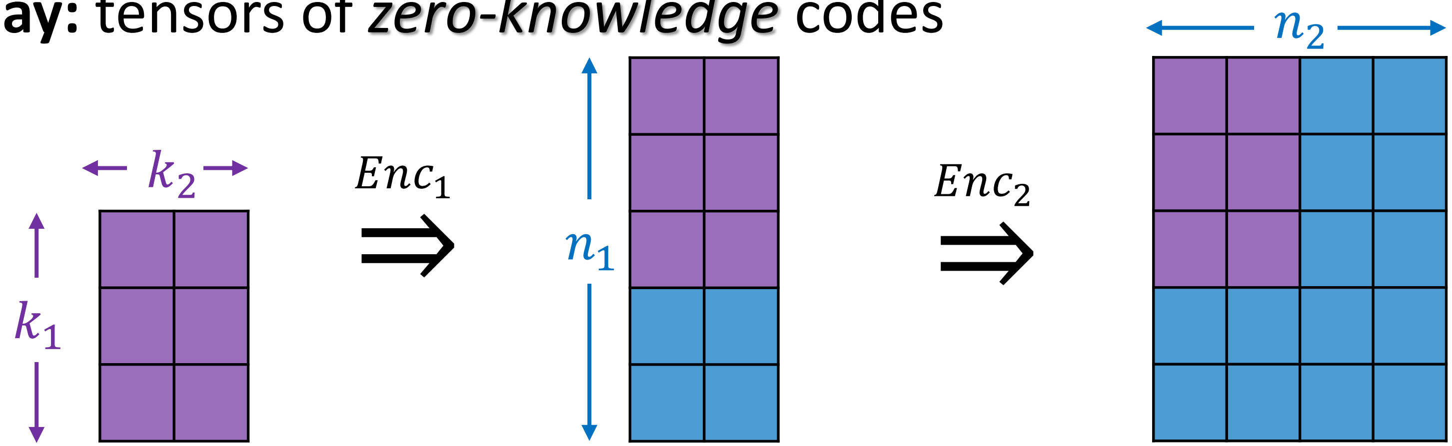
- **Base codes:** $C_1 : \mathbb{F}^{k_1} \to \mathbb{F}^{n_1}$, $C_2 : \mathbb{F}^{k_2} \to \mathbb{F}^{n_2}$ with encoding functions Enc_1, Enc_2



- Their **tensor** $\boldsymbol{C_1 \otimes C_2}$: $\mathbb{F}^{k_1 \cdot k_2} \to \mathbb{F}^{n_1 \cdot n_2}$
  - Columns $\in C_1$, rows $\in C_2$
  - Naturally extends to higher dimensions



- Very useful: tensor codes underly many PCP\IOP constructions
  - Special case: Low-Degree Extension (LDE) - tensor of Reed-Solomon
- **Today:** tensors of *zero-knowledge* codes

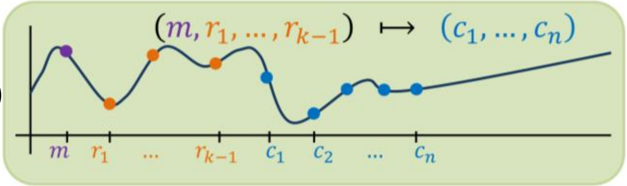


Zero-Knowledge Codes

- $C: \mathbb{F}^k \to \mathbb{F}^n$ with ***randomized*** encoding function Enc
- ***t*-ZK:** t codeword symbols reveal nothing about msg
- Tensors of ZK codes are very useful
  - Bivariate Shamir (tensor of Reed-Solomon) used in MPC protocols
  - 2-dim tensors used for verifiable secret sharing and MPC [CDM00]
  - Tensors of ZK codes underly ZK-IOPs [BCGV16,….,BCL22]

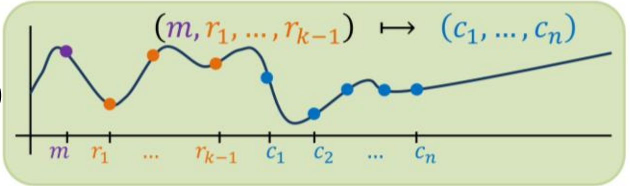
- **Main question:** how does tensoring affect ZK?
  - Our work: ***m-dimensional*** tensors of general codes
  - This talk: **2**-dimensional tensors $C_1 \otimes C_2$

Does Tensoring Preserve ZK?



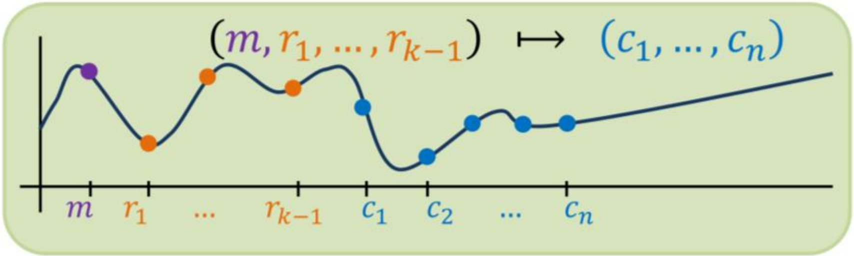
- For $i = 1{,}2$, $C_i\colon \mathbb{F}^{k_i} \to \mathbb{F}^{n_i}$ has t_i-ZK
- What ZK properties does $C_1 \otimes C_2$ have?
- We focus on a (specific) natural randomized encoding function
  - Encoding used in Shamir sharing, and by [BCL22]
- [BCL22] show $C_1 \otimes C_2$ has $\min\{t_1, t_2\}$-ZK
- [BCL22] asked: can bound be improved?
  - In particular, does $C_1 \otimes C_2$ have $\max\{t_1, t_2\}$-ZK?
- **We show:** $C_1 \otimes C_2$ has ZK against:
  - Adversaries reading t_1 *full rows*
  - Adversaries reading t_2 *full columns*
  - In particular, answer to [BCL22]'s question is: YES!

Does Tensoring Preserve ZK?



Why rows\columns?

- What we need for our short ZK-IOPs
- Secret shares in Bivariate Shamir



ncoding function

22]

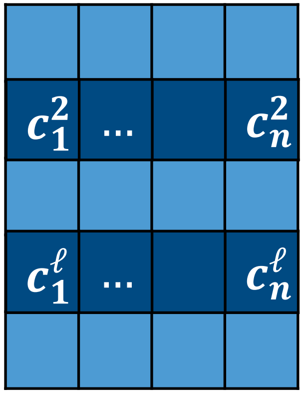
ZK?

- **We show:** $C_1 \otimes C_2$ has ZK against:
  - Adversaries reading t_1 *full rows*
  - Adversaries reading t_2 *full columns*
  - In particular, answer to [BCL22]'s question is: YES!
- **Our results are more general:**
  - *Only one* of the codes needs to have ZK
    - E.g., C_1 has t_1-ZK $\Rightarrow C_1 \otimes C_2$ has ZK against t_1 full rows (even if C_2 has no ZK guarantees)
  - *Ask now, decide later:* ZK against adversaries that make point queries, then decide whether to query rows or columns (if C_1, C_2 have "uniform" ZK)

Does Tensoring Preserve ZK? (Cont.)

- For $i = 1{,}2$, $C_i : \mathbb{F}^{k_i} \to \mathbb{F}^{n_i}$ has t_i-ZK
- **We show:** $C_1 \otimes C_2$ has ZK against:
  - Adversaries reading t_1 *full rows*
  - Adversaries reading t_2 *full columns*

$\boldsymbol{t_1 \cdot t_2}$ **codeword symbols** $\leq$



- **Question:** can we get ZK against *arbitrary* $t_1 \cdot t_2$ point queries?
- **We show the answer is NO:** $\exists$ t-ZK C s.t. $C \otimes C$ ***not*** $\omega(t)$-ZK
  - See paper for details

Zero-Knowledge IOPs for 3SAT: Blueprint

Goal: Prove that $\varphi \in 3SAT$

Encoding (Arithmetization)

$c := C(w)$

We show: high-rate C with ZK against *row\column adversaries*

Check $c \in C$

Verification

zero-knowledge

Sublinear CC

Σ ☑

on $\hat{w}$ using c: verify w satisfies φ

φ, w

P

φ

V

Zero-Knowledge IOPs for 3SAT: Blueprint

Goal: Prove that $\varphi \in 3SAT$

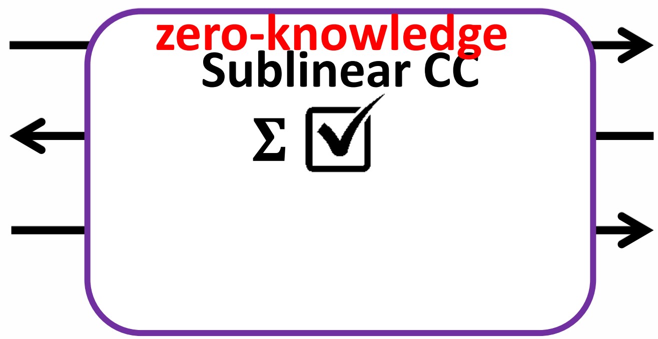
Encoding (Arithmetization)

$c := C(w)$

We show: high-rate C with ZK against *row\column adversaries*

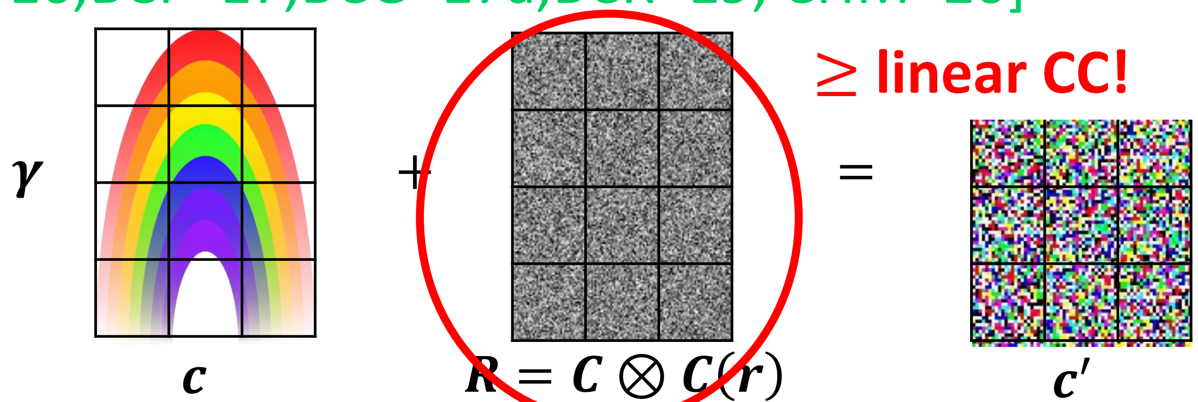
Check $c \in C$

Verification



The Sumcheck Protocol [LFKN90,Mei13]

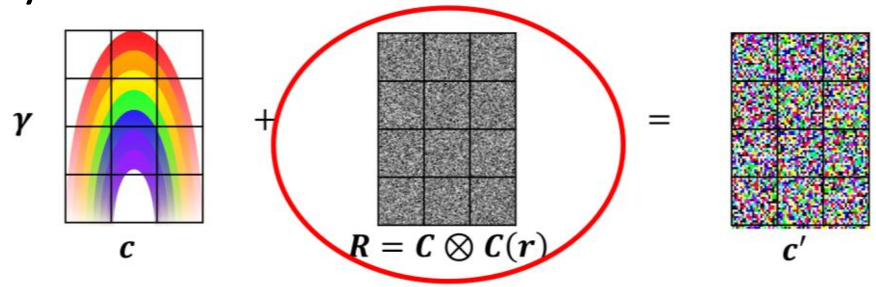
- **The sumcheck protocol:** IOP for checking $\Sigma_{i,j\in[k]} m(i,j) = \alpha$
  - Using encoding $c \in C \otimes C$ of $m \in \mathbb{F}^{k \cdot k}$
  - Amazingly, requires ***only one*** query to c!
- Many ZK-IOPs use ***Zero-Knowledge*** **sumchecks**
  - **ZK:** verifier's view efficiently simulatable with ***few queries to c***
    - How few? One query\ same as verifier\ slightly more than verifier
  - Prover's messages reveal (almost) nothing on m!
- Existing ZK sumcheck IOPs: apply IOP on randomly shifted codeword c' (use standard sumcheck as BB)* [BCGV16,BCF+17,BCG+17a,BCR+19, CHM+20]



*Omitting sublinear-CC sumchecks for ***polynomial codes*** (with HVZK [XZZ+19] or for sparse polynomials [BCL22])

ZK Sumcheck with Sublinear CC?

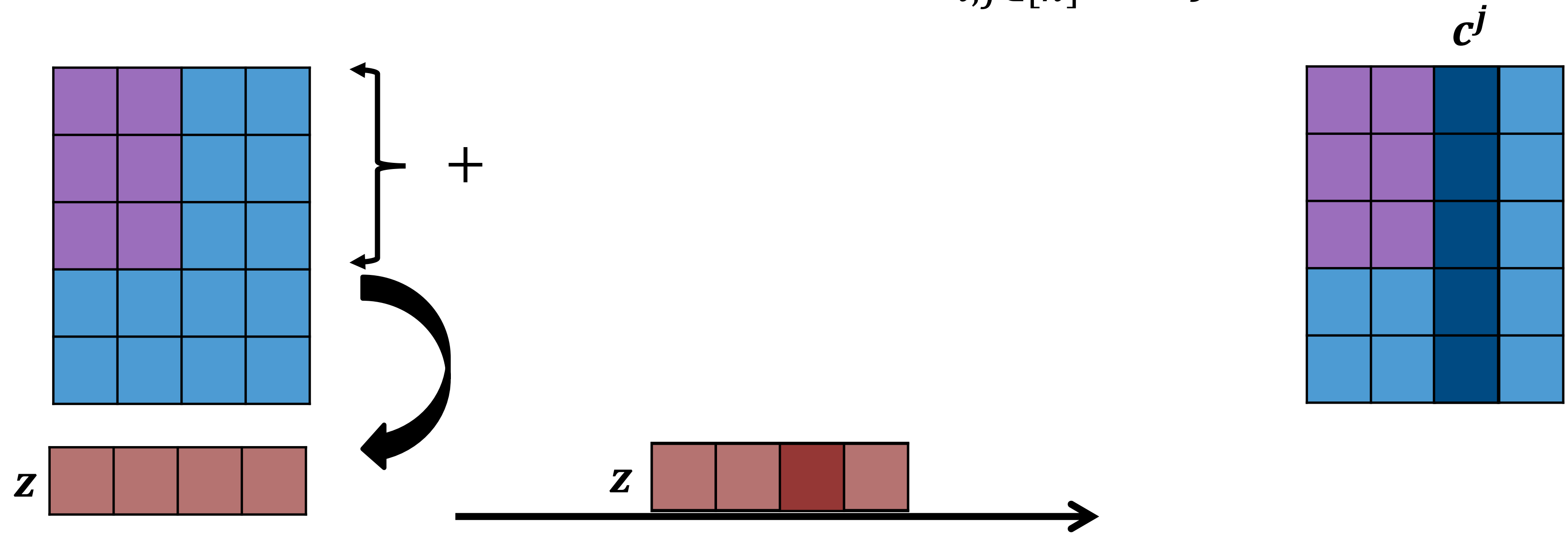
- Existing ZK sumchecks have ≥ linear CC [BCGV16, BCFGRS17, BCGRSSVW19, ZXZS20, CHMMVW20]
  - Long masking hides (all but few) symbols of c
  - BB in underlying sumcheck



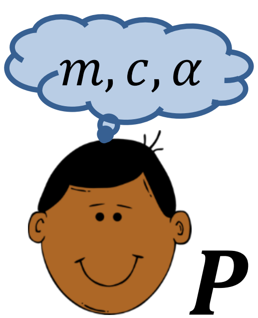
- Sublinear-CC ZK sumcheck requires shorter mask
- **High-level idea for reducing randomness:**
  - Exploit structure of *specific* sumcheck protocol (we use [RR20])
  - Tailor randomness to hide type of information sumcheck reveals
- Our ZK sumcheck reveals full *columns* of c
  - More than fully-masked sumchecks…
  - … but combined with our new results on tensors of ZK codes, still suffices for ZK-IOPs

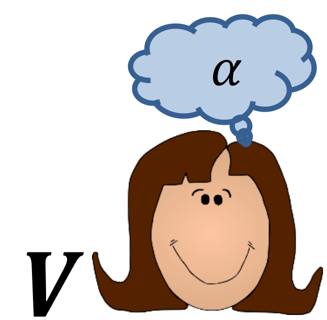
The Sumcheck of [RR20] (Simplified)

- $c \in C \otimes C$ encoding $m \in \mathbb{F}^{k \cdot k}$, $\alpha = \Sigma_{i,j \in [k]} m(i,j)$



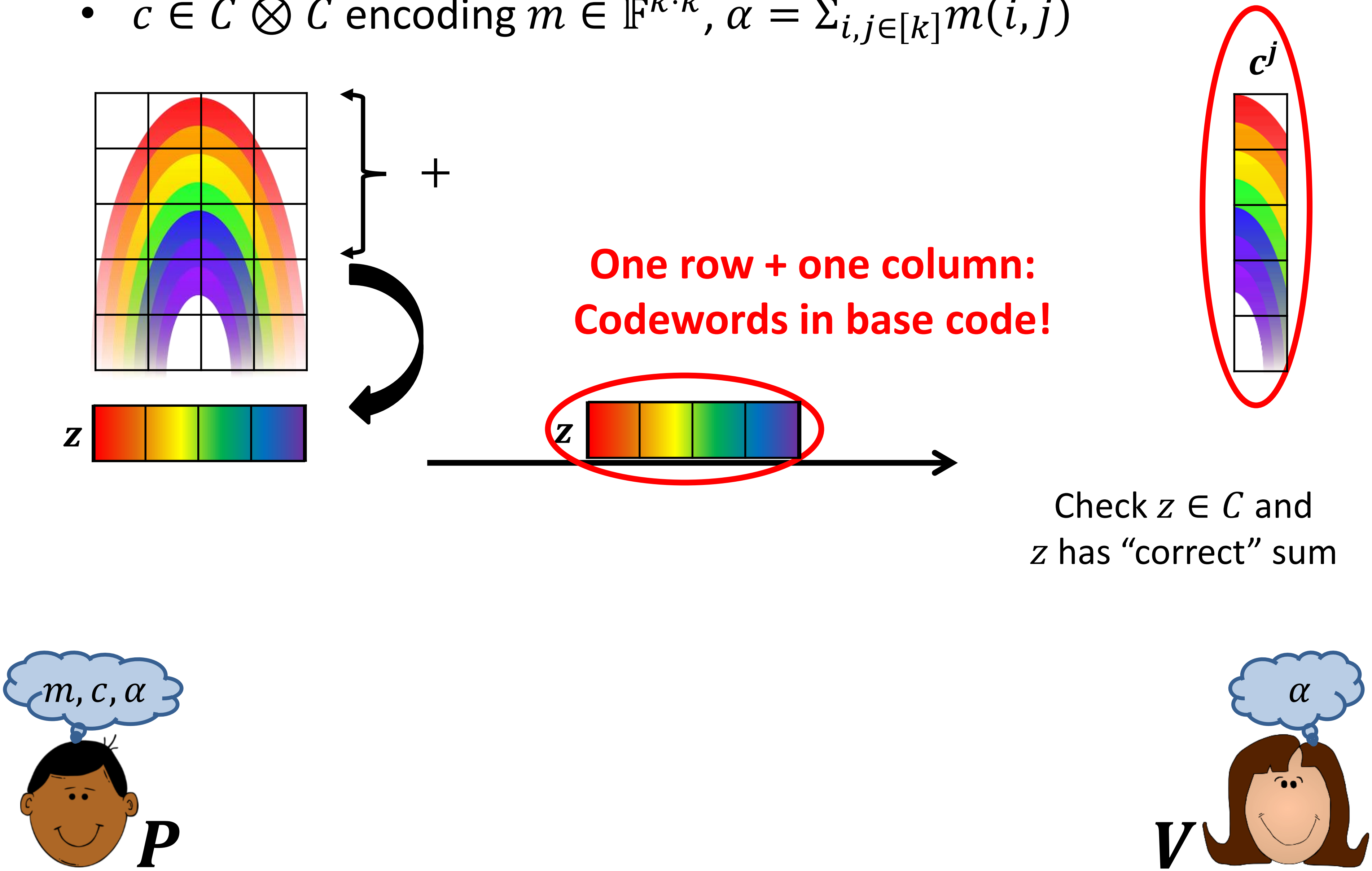
Check $z \in C$ and
z has “correct” sum





Information Revealed in [RR20]'s Sumcheck

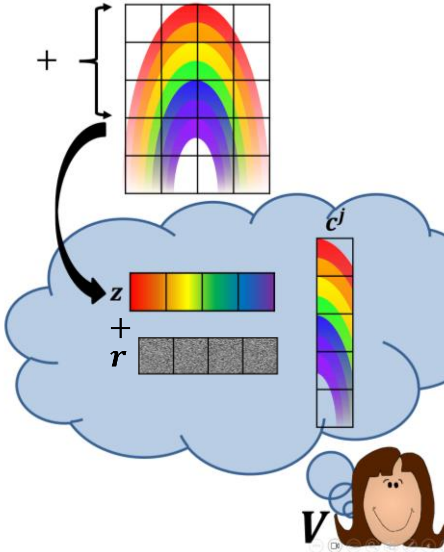
- $c \in C \otimes C$ encoding $m \in \mathbb{F}^{k \cdot k}$, $\alpha = \Sigma_{i,j \in [k]} m(i,j)$



sum

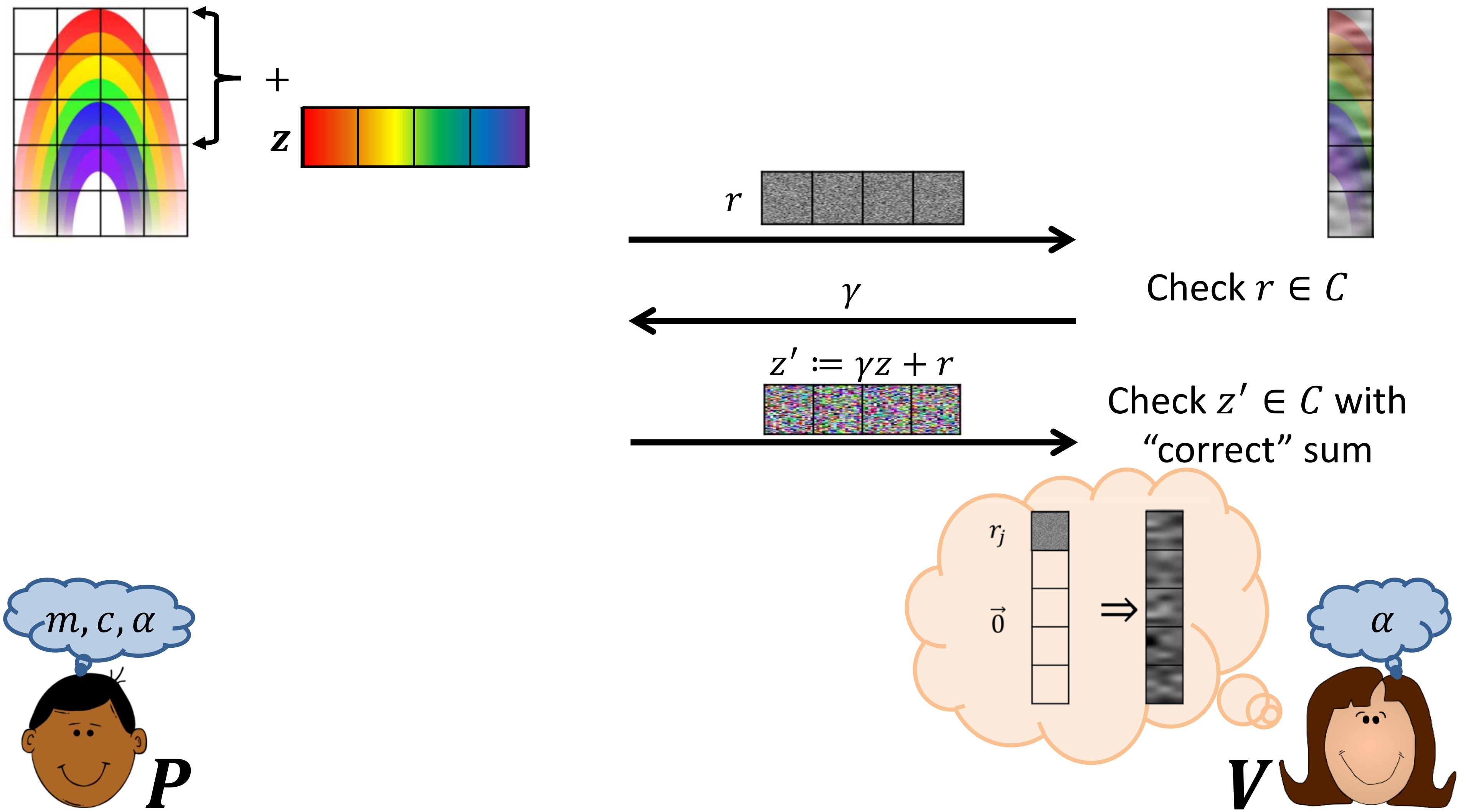
Masking [RR20]'s Sumcheck

- [RR20]'s sumcheck reveals row + column
  - Codewords in base code!
- Our sublinear-CC ZK sumcheck: mask in ***base code***
  - Sending mask requires ***sublinear*** CC



Our ZK Sumcheck with Sublinear CC (Simplified)

- $C: \mathbb{F}^k \to \mathbb{F}^n$
- $c \in C \otimes C$ encoding $m \in \mathbb{F}^{k \cdot k}$



Wrapping Up: ZK-IOPs Approaching Witness Len

- Strong **ZK properties for tensor codes**
  - ZK against $\max\{t_1, t_2\}$ queries
  - ZK against rows\columns
- **Limitations:** can't achieve $t_1 \cdot t_2$-ZK in general case

We show: high-rate C with ZK against *row\column adversaries*

We show:
Sublinear-CC ZK sumcheck
(reveals columns of c)

How? Mask in base code

$\Downarrow$

First *ZK*-IOPs *approaching witness length*

For details: eprint.iacr.org/2024/816

Thank you!