Zero-Knowledge and/or Succinct Proofs or Arguments

Efficient Zero-Knowledge Proofs: A Modular Approach

Yuval Ishai

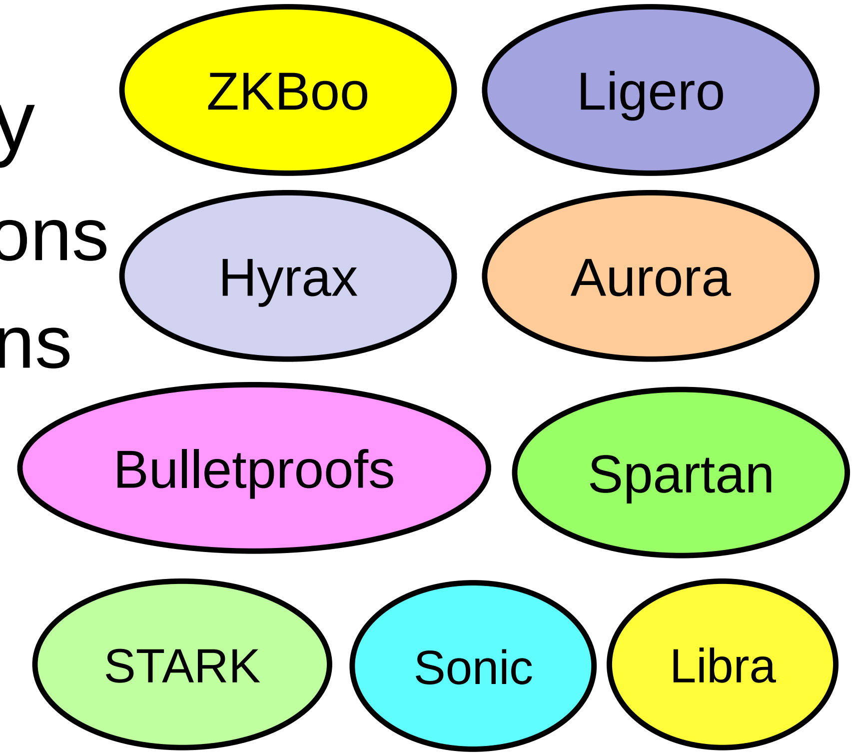
Technion

European Research Council

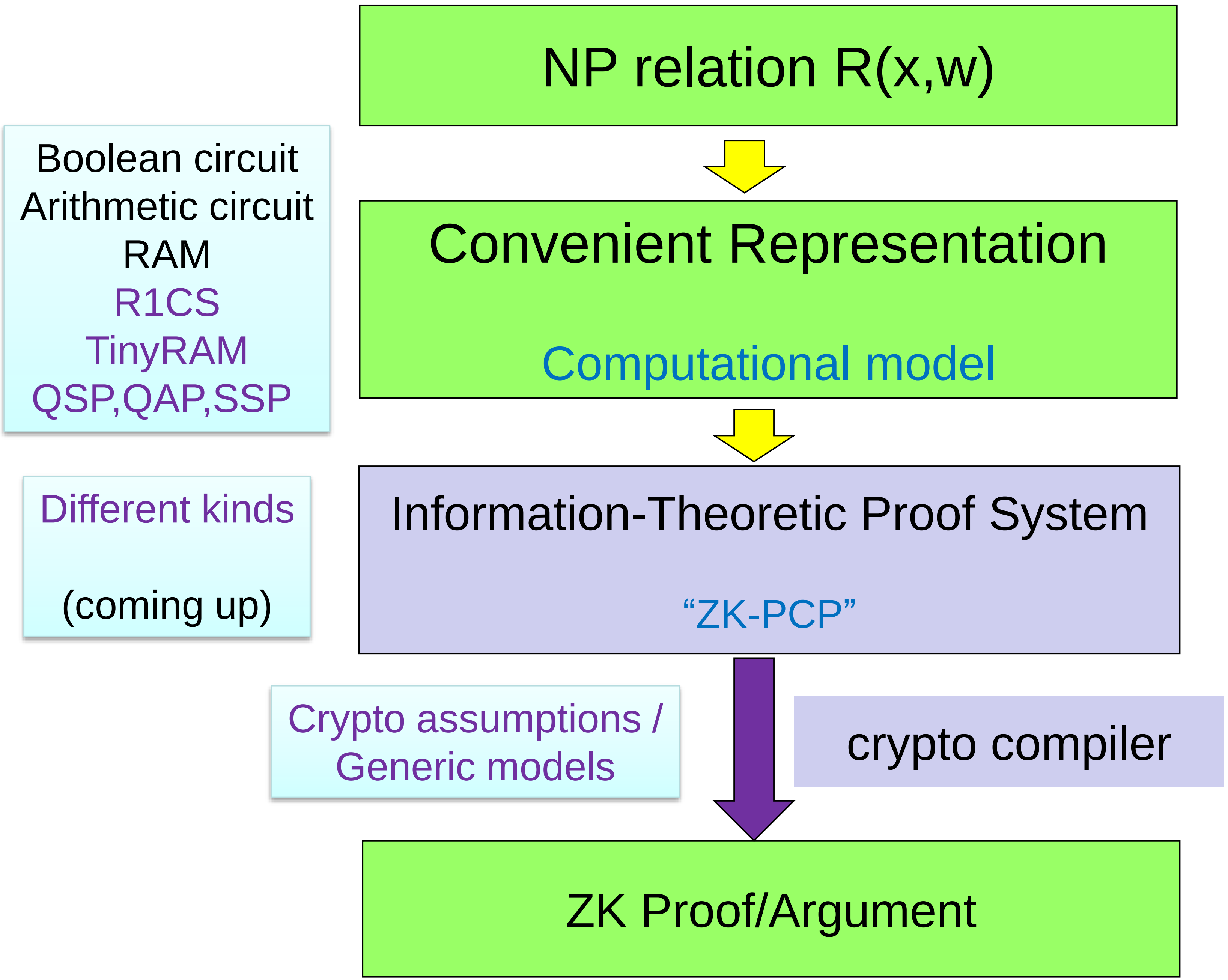
Foundational Aspects of Blockchain Technology, ICTS Bangalore, Jan 16, 2020

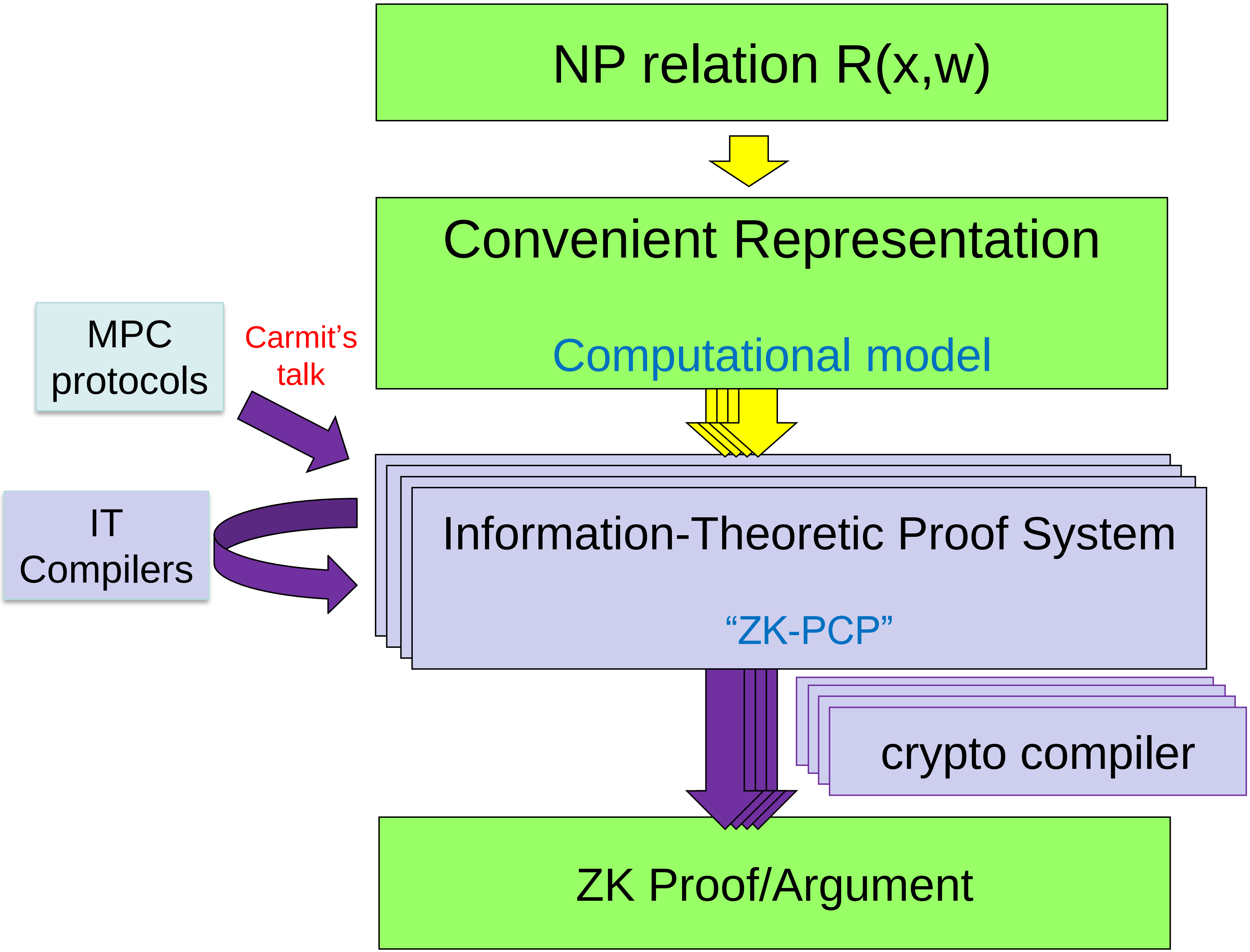
Broad Motivation

- ZK research is a big party
  - Many motivating applications
  - Many challenging questions
  - Many exciting results



- Big party → Big mess?
- This talk: advocating a modular approach
  - Separate "information-theoretic" and "crypto" parts
  - General cryptographic compilers (IT → crypto)
  - General information-theoretic compilers (IT → IT)


NP relation R(x,w)
Convenient Representation
Computational model
Boolean circuit
Arithmetic circuit
RAM
R1CS
TinyRAM
QSP,QAP,SSP
Information-Theoretic Proof System
"ZK-PCP"
Different kinds
(coming up)
Crypto assumptions / Generic models
crypto compiler
ZK Proof/Argument


NP relation R(x,w)
Convenient Representation
Computational model
MPC protocols
Carmit's talk
IT Compilers
Information-Theoretic Proof System
"ZK-PCP"
crypto compiler
ZK Proof/Argument

Why?

- Simplicity
  - Break complex tasks into simpler components
  - Easier to analyze and optimize
  - Potential for proving lower bounds
- Generality
  - Apply same constructions in different settings
  - Research deduplication, less papers to read/write
- Efficiency
  - Port efficiency improvements between settings
  - Mix & match different components
  - Systematic exploration of design space

ZK Zoo

(ignoring assumptions for now…)

Qualitative features

- Interactive?
- Succinct?
- Fast verification?
- Public verification?
- Public input?
- NP vs. P?
- Trusted setup?
- Symmetric crypto only?
- Post quantum?

Quantitative features

- Communication
- Prover complexity
- Verifier complexity

Commercialization efforts

Standardization process

Several talks in this workshop

Optimal ZKP protocol?

Food for thought…

- Which verifier is better?
  - V1: SHA256 hash
  - V2: PKE decryption

- V2 can be more obfuscation-friendly! [BISW17]
  - Relevant complexity measure: branching program size
  - Promising avenue for practical general-purpose obfuscation
  - Motivated "lattice-based" designated-verifier SNARKs

- Similar: MPC-friendly prover, etc.

Back to the 20^{th} Century

Theorem [GMW86]:
Bit-commitment → ZKP for all of NP

Theorem [GMW86+Naor89+HILL99]:
One-way function → ZKP for all of NP

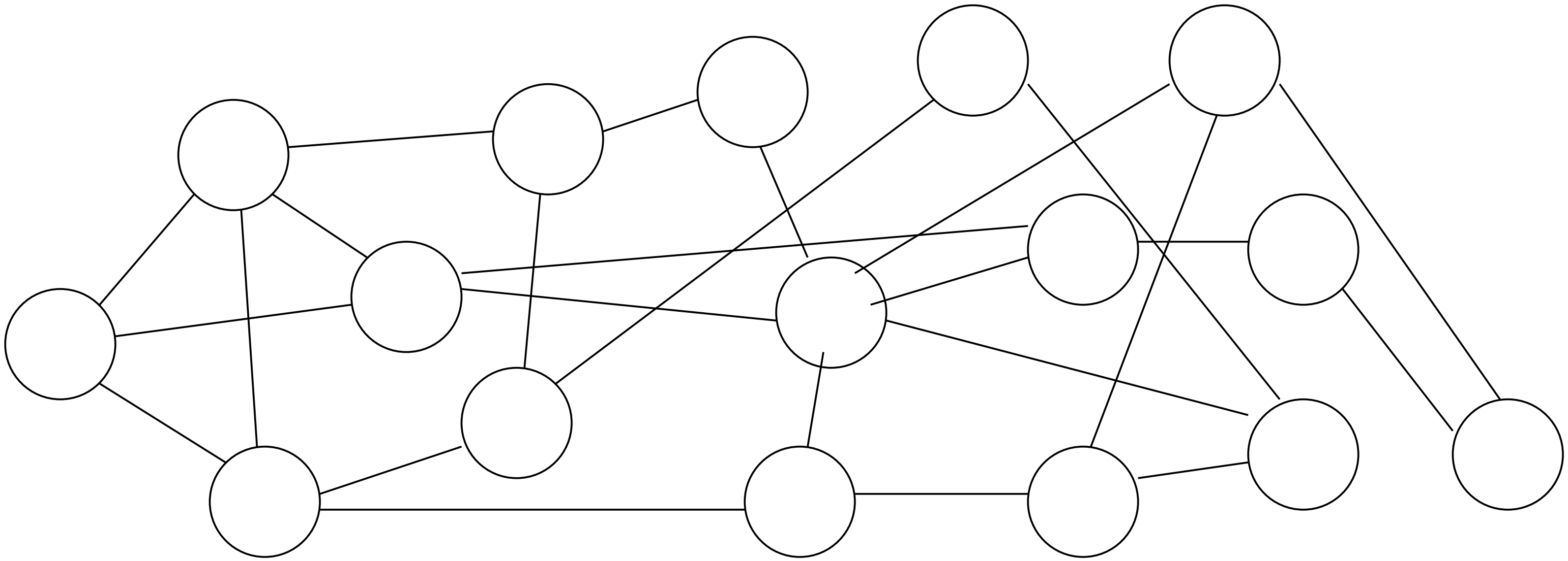
Theorem [OW93]:
ZKP for “hard on average” L in NP → i.o. one-way function

Are we done?

ZKP for 3-Colorability

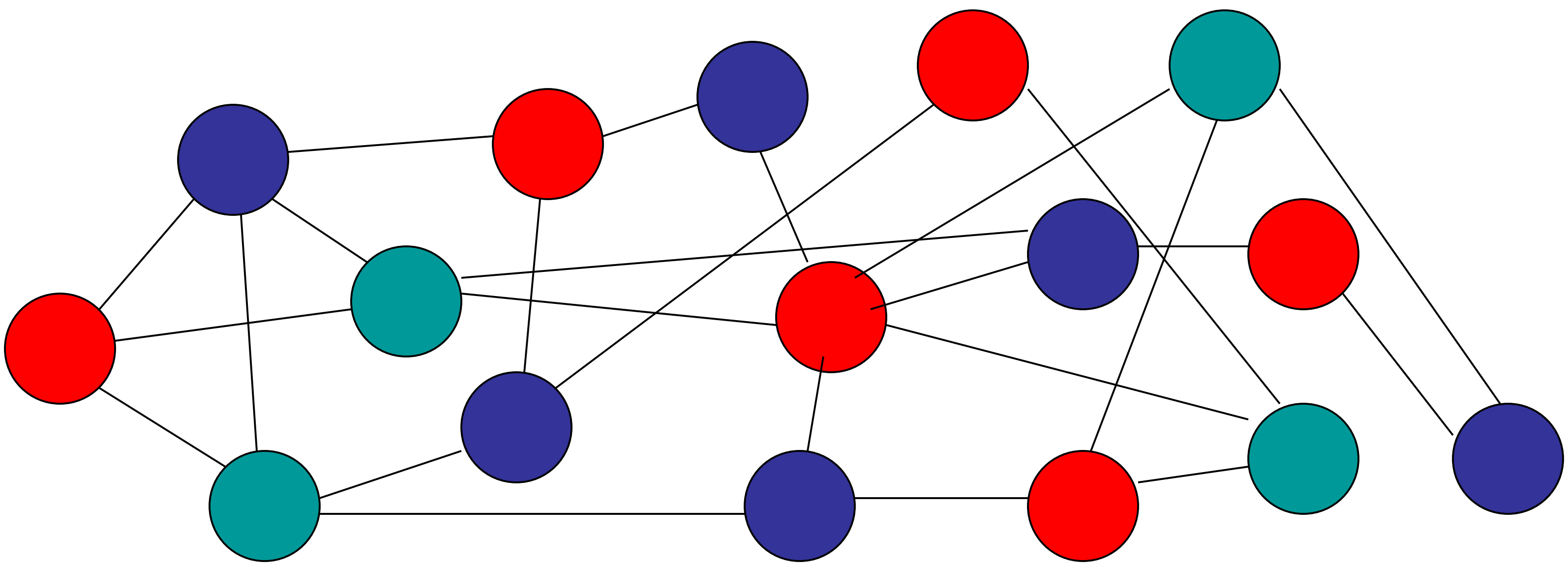
[GMW86]

- Prover wants to prove that a given graph is 3-colorable

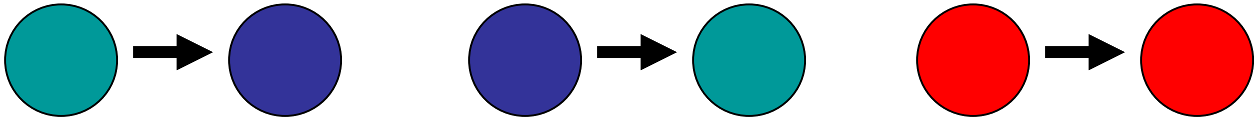


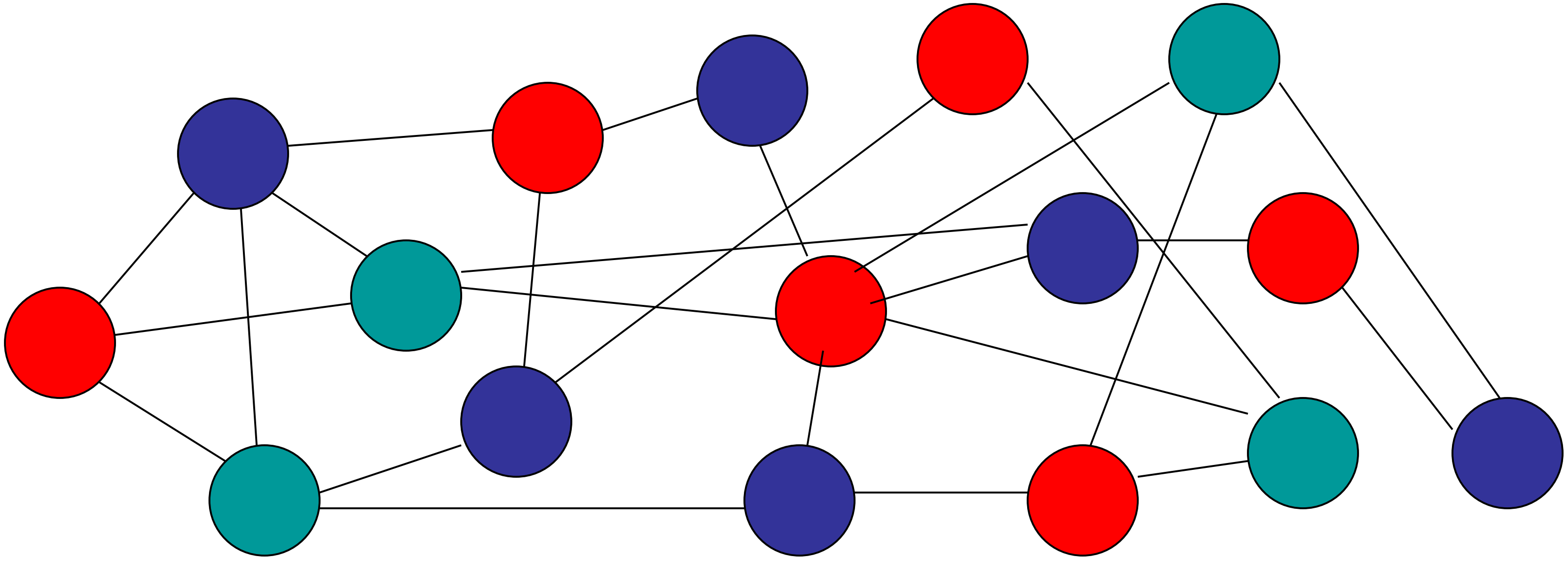
ZKP for 3-Colorability

- Prover wants to prove that a given graph is 3-colorable
  - x=graph w=coloring

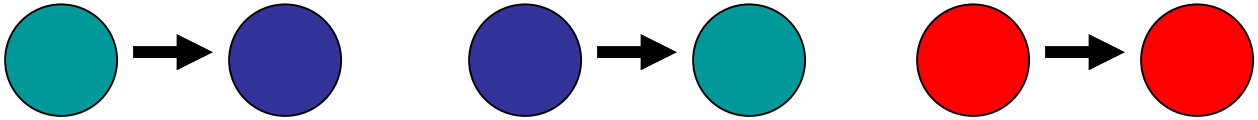


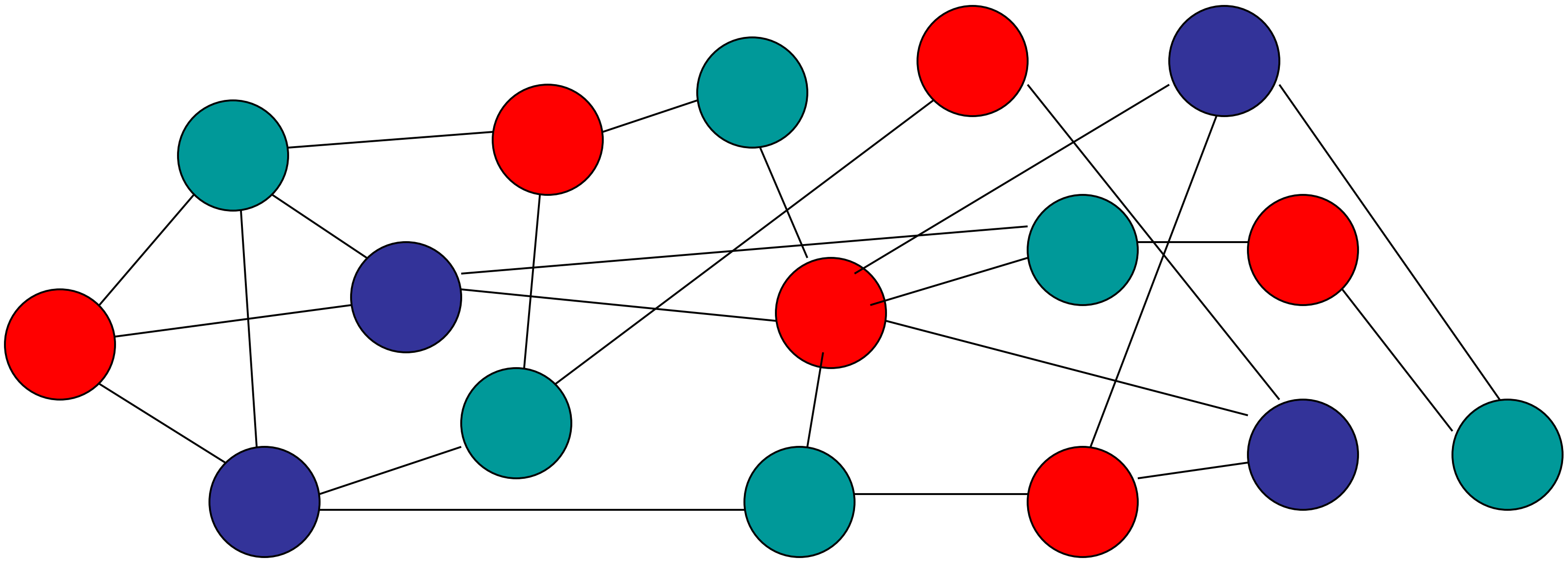
ZKP for 3-Colorability

- Prover randomly permutes the 3 colors (6 possibilities)
  - Say, 



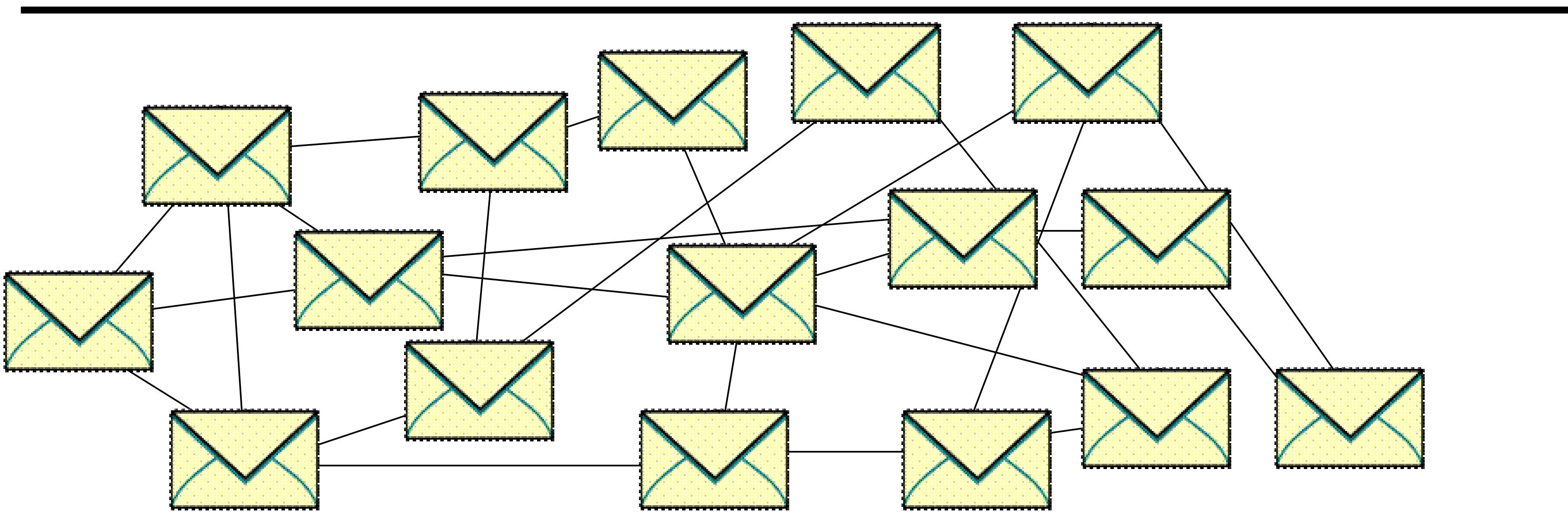
ZKP for 3-Colorability

- Prover randomly permutes the 3 colors (6 possibilities)
  - Say, 



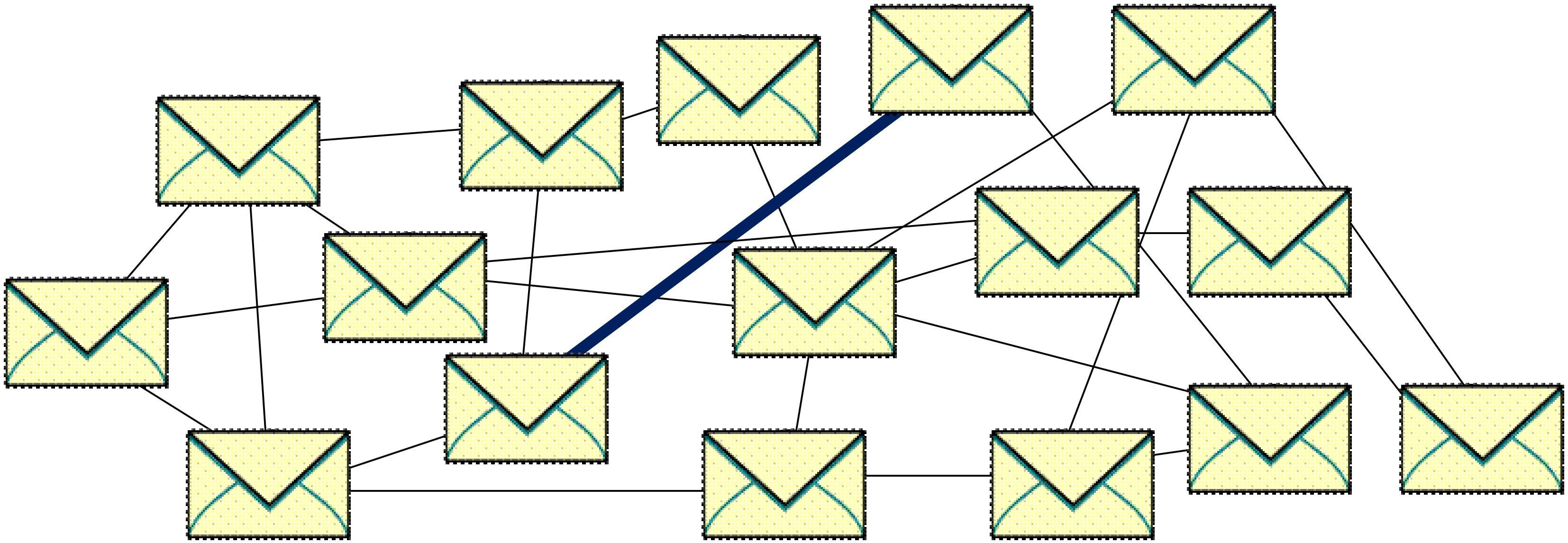
ZKP for 3-Colorability

- Prover separately commits to color of each node and sends commitments to Verifier



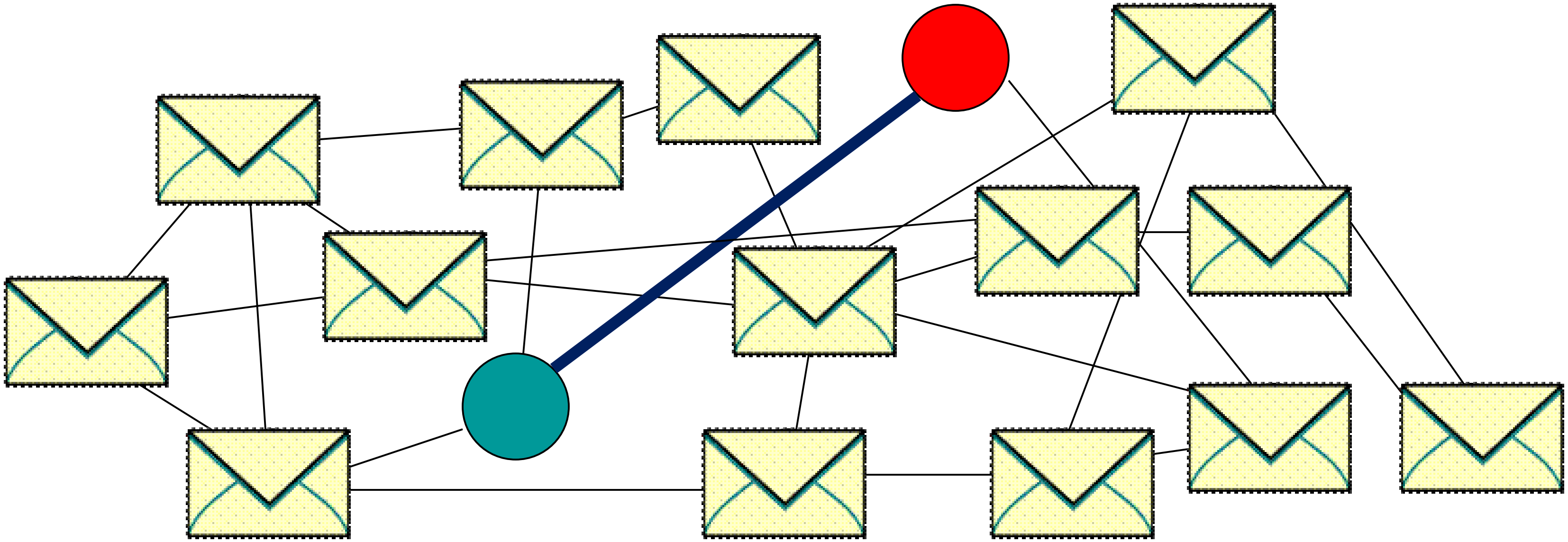
ZKP for 3-Colorability

- Verifier challenges Prover by selecting a random edge



ZKP for 3-Colorability

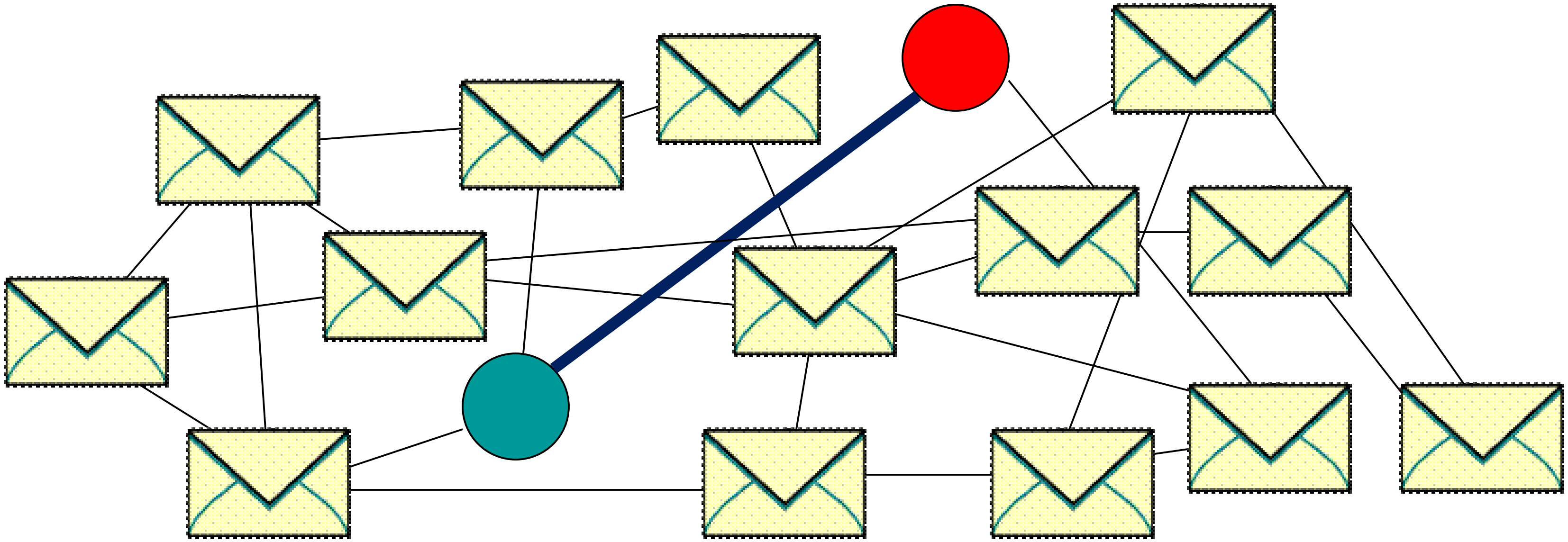
- Prover sends decommitments for opening the colors of the two nodes



ZKP for 3-Colorability

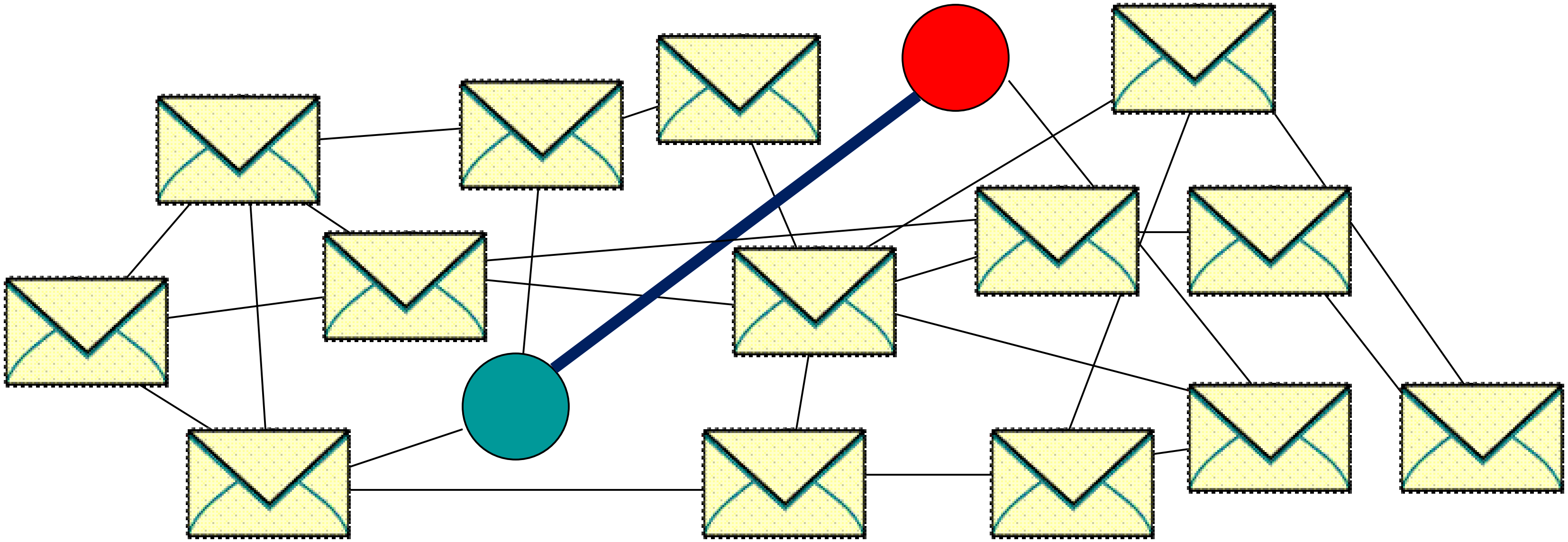
- Verifier accepts if both colors are valid and are distinct (otherwise it rejects).

- Repeat O(|E|) times to amplify soundness

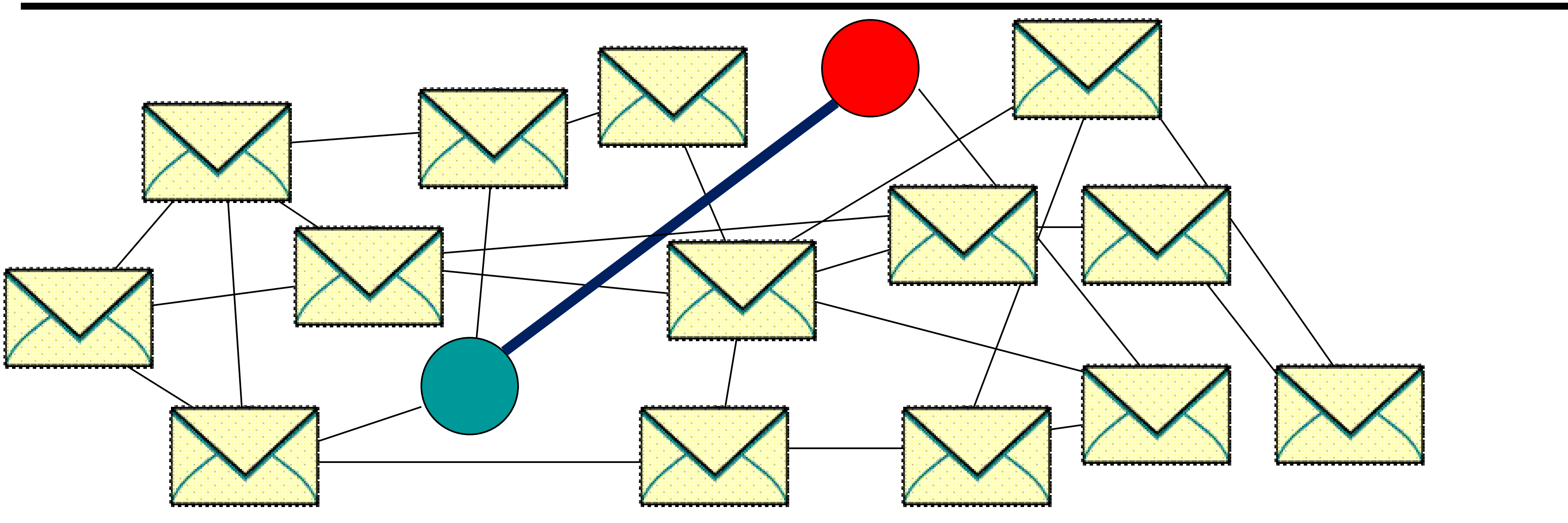


Issues

- Security proof more subtle than it may seem
  - Need to redo analysis for Hamiltonicity-based ZK?
- Two sources of inefficiency
  - Karp reduction
  - Soundness amplification (+ many rounds)



Abstraction to the rescue…



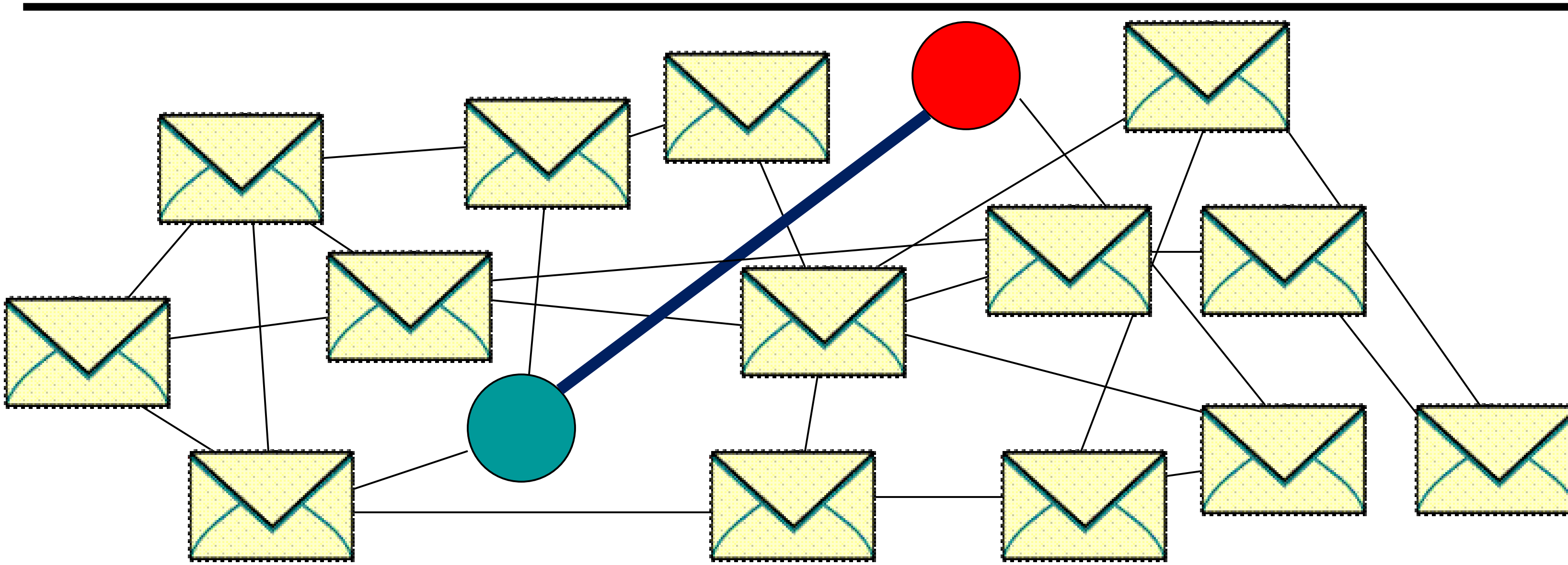
Information-Theoretic Proof System: ZK-PCP

Prover: (x,w) $\rightarrow \pi$

$\pi =$

1	3	1	2	1	3	1	2	1	1	3	1	3	1	2	1

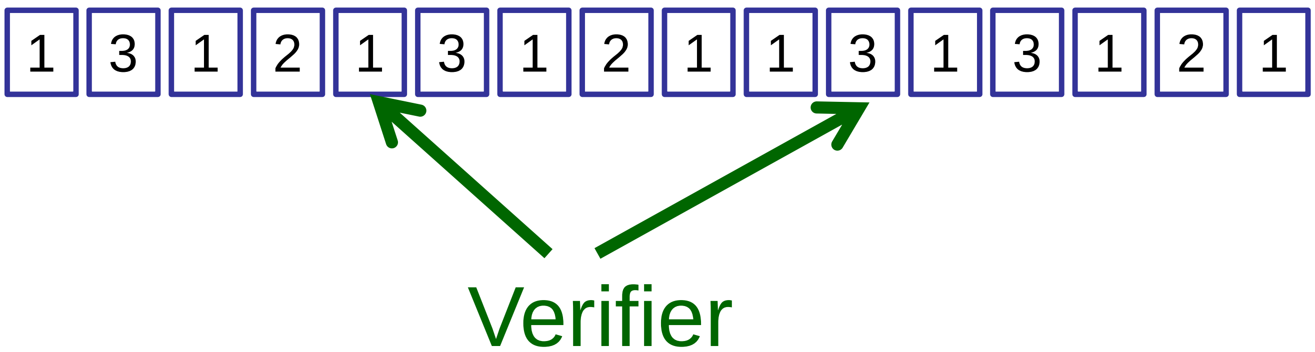
Verifier



Information-Theoretic Proof System: ZK-PCP

Prover: (x,w) $\rightarrow \pi$

$\pi =$

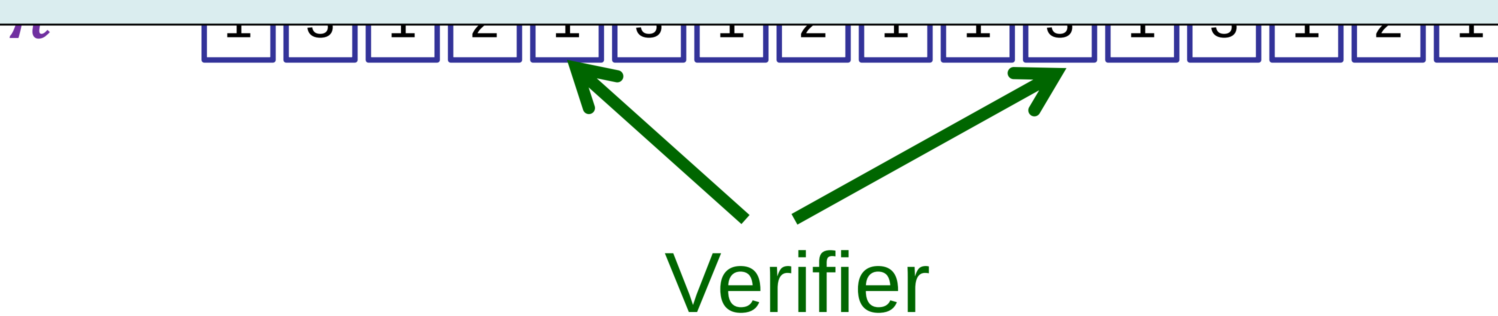


- Simple security definition
  - Completeness
  - Perfect (public-coin) ZK
  - Soundness error ϵ
    (amplified via repetition)

- Clean efficiency measures
  - Alphabet size
  - Query complexity
  - Prover computation
  - Verifier computation

Information-Theoretic Proof System: ZK-PCP

- Here: ZK for queries made by honest verifier
- More difficult: ZK for t-bounded malicious verifiers [KPT97, IMS12, IWY16]



- Simple security definition
  - Completeness
  - Perfect (public-coin) ZK
  - Soundness error ϵ
    (amplified via repetition)

- Clean efficiency measures
  - Alphabet size
  - Query complexity
  - Prover computation
  - Verifier computation

Information-Theoretic Proof System: ZK-PCP

Prover: (x,w) → π

π = | 1 | 3 | 1 | 2 | 1 | 3 | 1 | 2 | 1 | 1 | 3 | 1 | 3 | 1 | 2 | 1 |

Verifier

Crypto compilers

+Stat-binding commit [GMW86, PW09] → ZK in plain model

+Stat-hiding commit [GK96, IMS12] → ZK in plain model

+Random oracle [FS86,Mic00] → NIZK in ROM

Information-Theoretic Proof System: ZK-PCP

Prover: (x,w) $\rightarrow \pi$

π = 1 3 1 2 1 3 1 2 1 1 3 1 3 1 2 1

Verifier

Recent line of work…

Crypto compilers

+Stat-binding commit [GMW86, PW09]

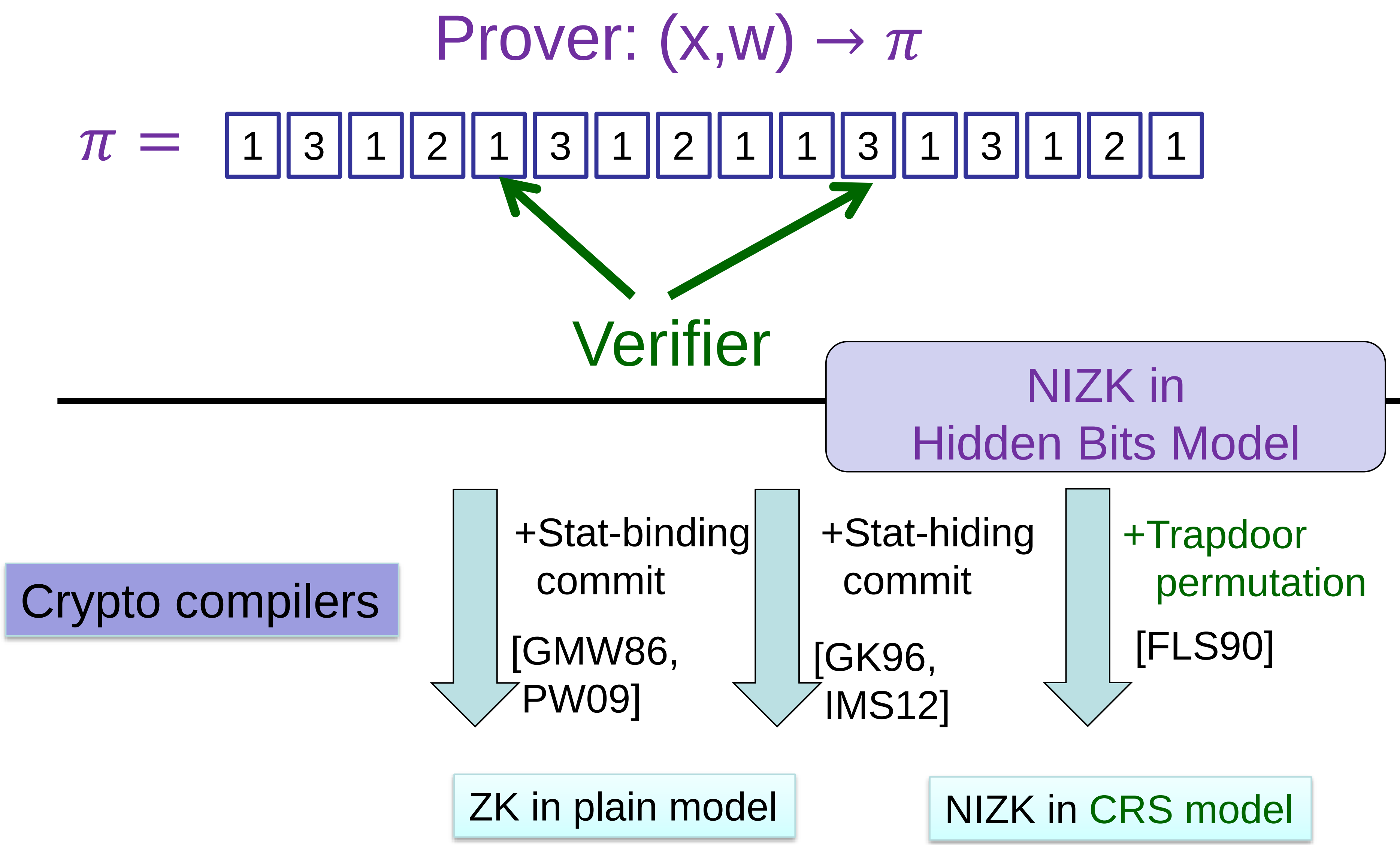
+Stat-hiding commit [GK96, IMS12]

~~+Random oracle~~ [FS86,Mic00]

ZK in plain model

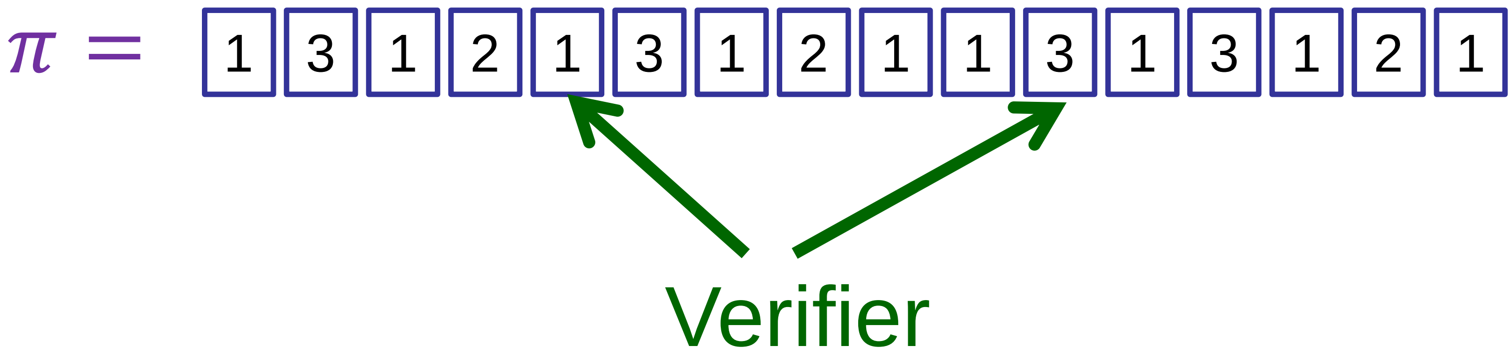
NIZK in CRS model

Information-Theoretic Proof System: ZK-PCP



Information-Theoretic Proof System: ZK-PCP

Prover: (x,w) $\rightarrow \pi$

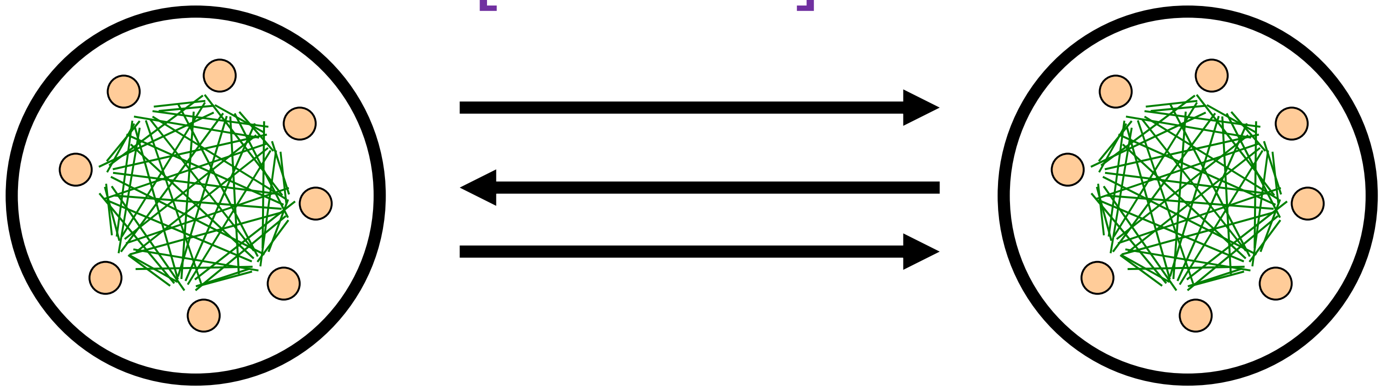


Less “magical”?

Better parameters?

IT Compilers: MPC → ZK-PCP

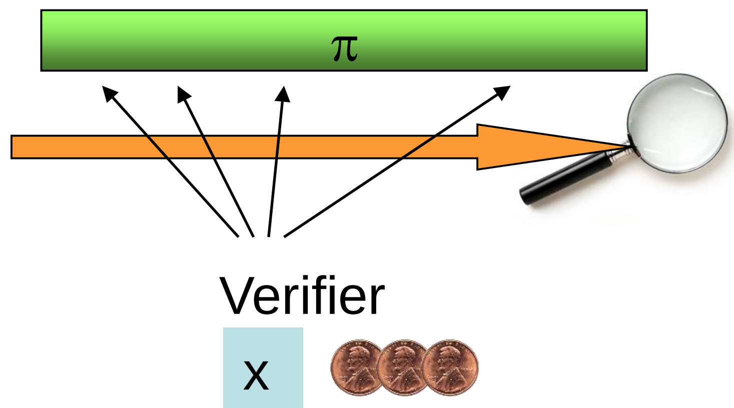
[IKOS07]



- Simple ZK proofs using:
  - (2,5) or (1,3) semi-honest MPC [BGW88,CCD88,Mau02]
  - (2,3) or (1,2) semi-honest MPC^{OT} [Yao86,GMW87,GV87,GHY87, HV16]
  - Practical [GMO16,CDG+17,KKW18] ➔ post-quantum signatures!
- ZK proofs with O(|C|) communication
  - (n/5,n) malicious MPC based on AG codes [CC06,DI06,IKOS07]
- Hitting the circuit-size barrier?
  - Sublinear ZK for special tasks: linear algebra, non-abelian groups,…
  - Going (somewhat) sublinear in general: Ligero [AHIV17] – Carmit's talk

Going fully sublinear?

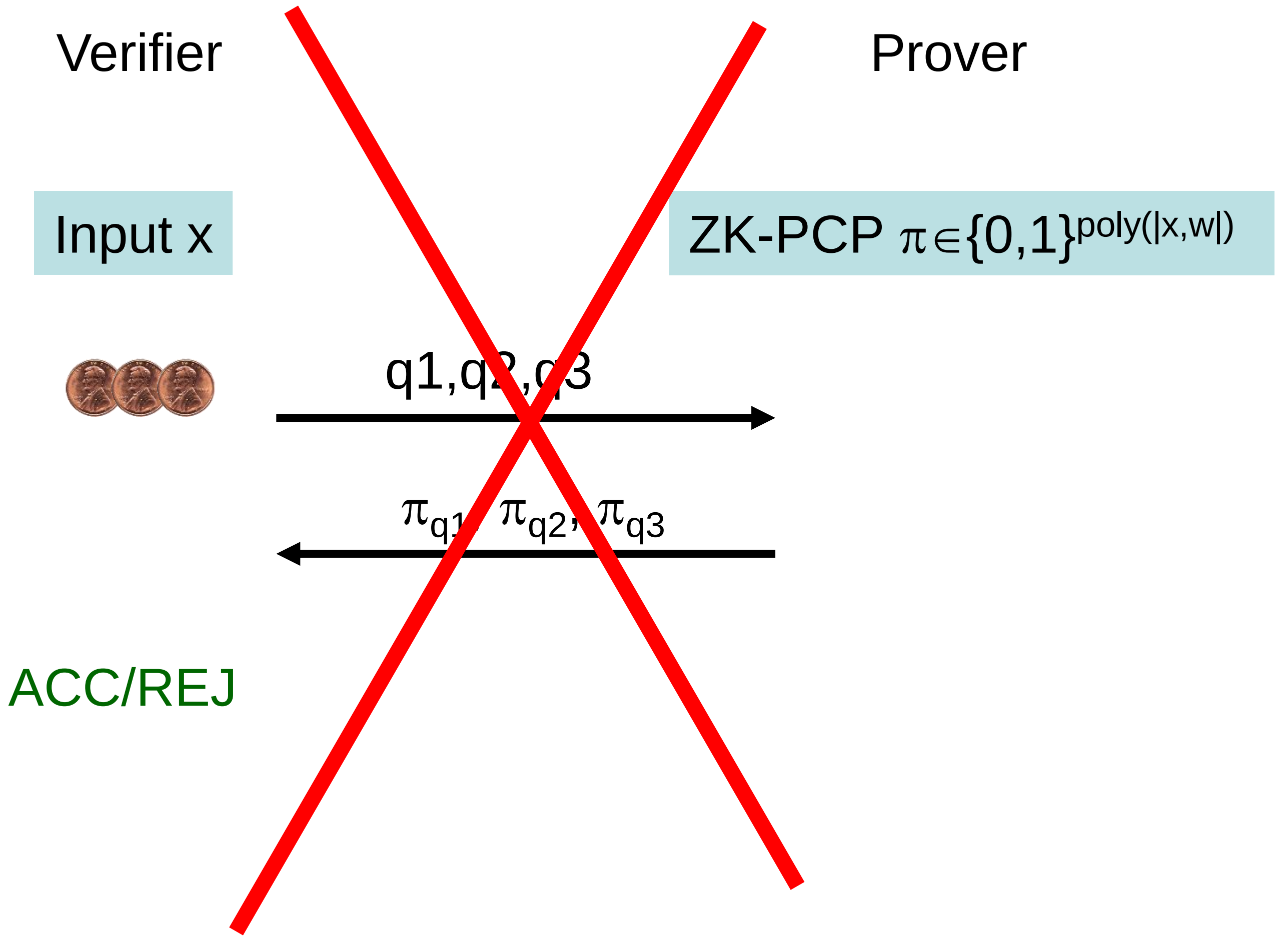
Traditional PCPs



- x∈L ➔ ∃π Pr[Verifier accepts π] =1
- x∉L ➔ ∀π* Pr[Verifier accepts π*]≤1/2

- PCP Theorem [AS92,ALMSS92,Dinur06]:
  NP statements have polynomial-size PCPs in which the verifier reads only O(1) bits.
  - Can be made ZK with small overhead [KPT97,IW04]

Still need crypto compiler…



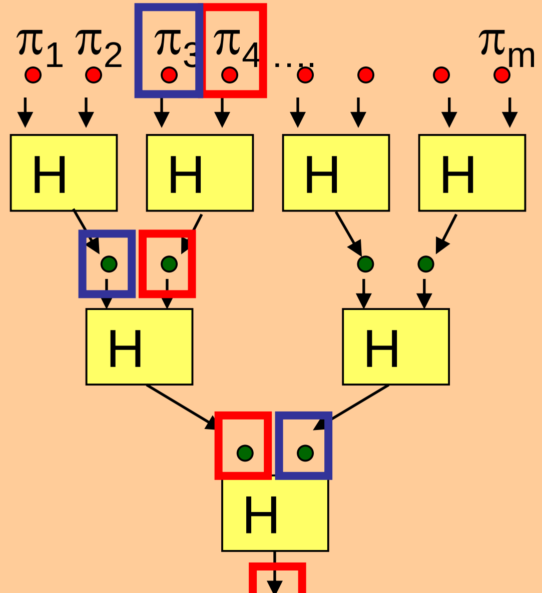
Crypto Compiler

[Kil93,Mic94]

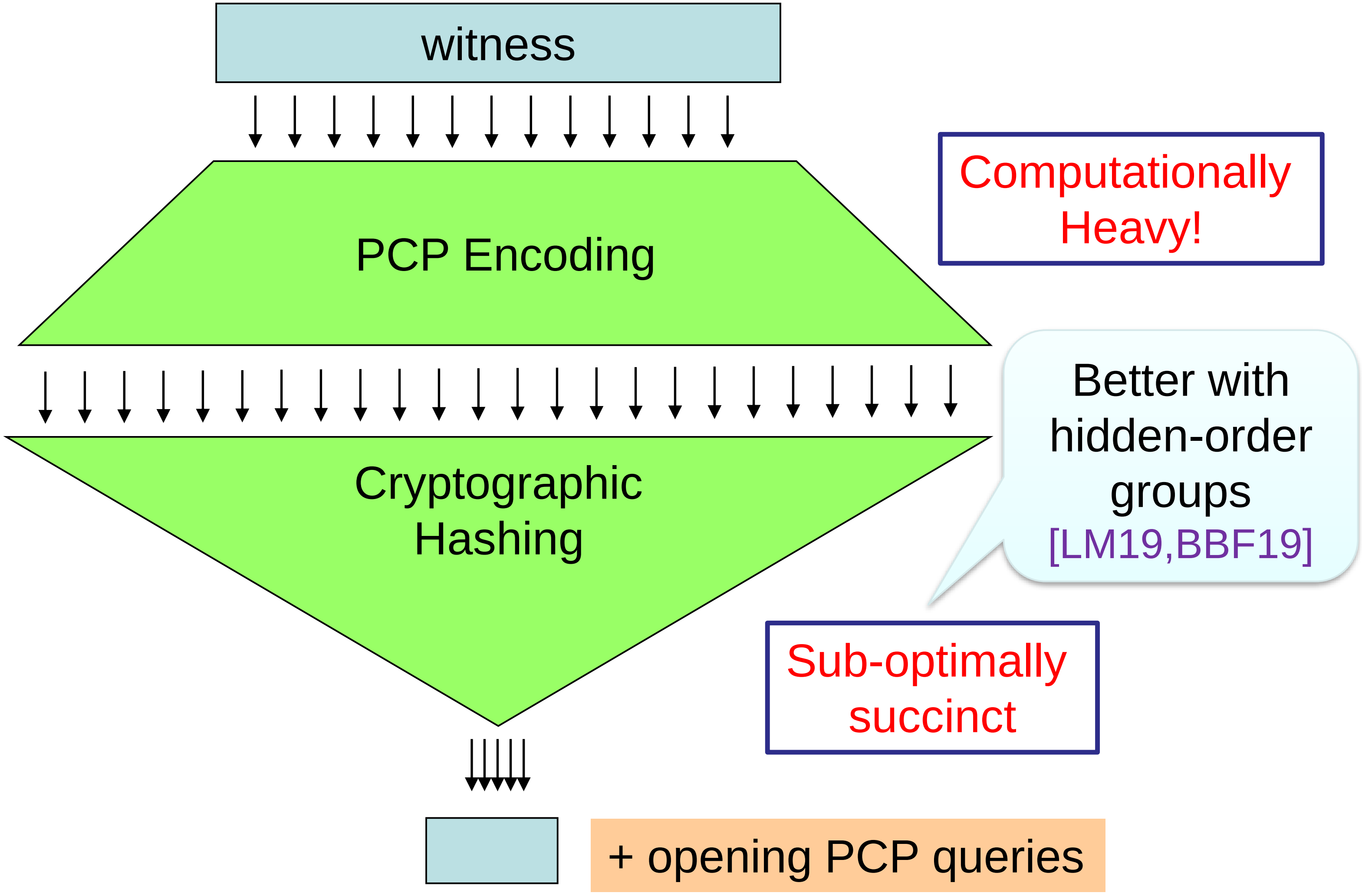
Merkle Tree construction

H = collision resistant hash function

$H:\{0,1\}^* \rightarrow \{0,1\}^k$



Limitations



Relaxing PCP model 1: Interaction

Prover

$\pi_1 =$ 1 3 1 2 1 3 1 2 1 1 3 1 3 1 2 1

Verifier

Challenge

$\pi_2 =$ 1 3 1 2 1 3 1 2 1 1 3 1 3 1 2 1

Interactive PCP [KR08,GIMS10]
IOP [BCS16,RRR16]

Practical systems:
STARK, Aurora

Verifier

Challenge

Relaxing PCP model 2: Linear PCP

[ALMSS98,IKO07,BCIOP13]

over a (large) finite field F

Prover

$\pi =$	4	3	1	2	8	3	1	2	1	9	3	1	6	1	2	1

inner product

$q_1 =$	5	3	6	2	1	3	1	2	1	1	6	1	3	1	8	1
$q_2 =$	7	3	1	2	4	3	1	2	7	1	3	1	7	1	2	1
$q_3 =$	1	2	1	2	1	9	1	2	5	1	4	1	3	1	3	1

Verifier

a_1

a_2

a_3

x

ACC / REJ

Advantages of Linear PCPs

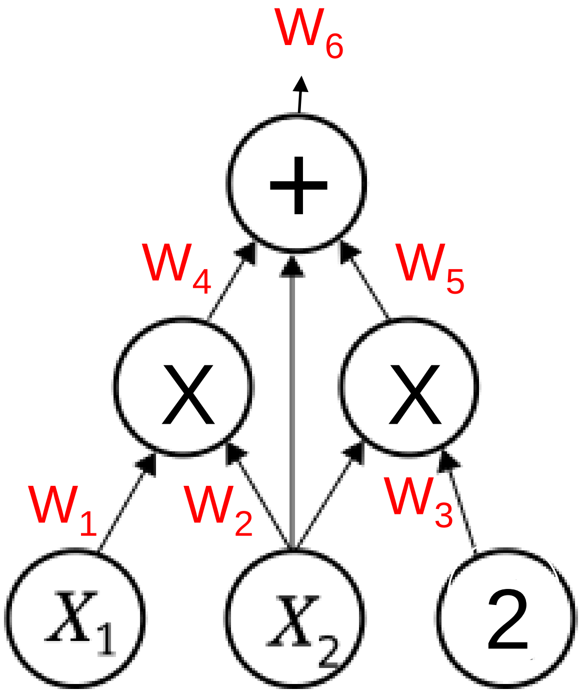
- Simple!
  - Coming up…
- Short, efficiently computable
  - $O(|C|)$-size, quasi-linear time via QSP/QAP [GGPR13, …]
- Negligible soundness error with O(1) queries
  - Reusable soundness

$$\Pr[\pi^* \text{ is accepted}] \text{ is either 1 or } O(1/|F|)$$

  - Near-optimal succinctness
  - In fact, 1 query is enough! [BCIOP13]

Example: The Hadamard PCP

[ALMSS91,IKO07]



$$W_6 = 0$$
$$W_6 - (W_2 + W_4 + W_5) = 0$$
$$W_5 - (W_2 x W_3) = 0$$
$$W_4 - (W_1 x W_2) = 0$$
$$W_3 = 2$$

- Proof: π = (W, W x W)
- 3 linear queries, soundness error 2/|F|:
  - Consistency of two parts of π: $\langle W, R\rangle^2 = \langle W \times W, R \times R\rangle$
  - Consistency with gates: random linear combination of equations

Crypto Compilers for Linear PCPs

- First generation [IKO07,GI08,Gro09,SMBW12,…]
  - Standard assumptions
    - Linearly homomorphic encryption, discrete log
  - Interactive, one-way-succinct/somewhat succinct
  - Idea: use succinct vector-commitment with linear opening

- Second generation [Gro10b,Lip12,GGPR13,BCIOP13,…]
  - Strong "knowledge" or "targeted malleability" assumptions
  - Non-interactive using a (long, structured) CRS
  - Publicly verifiable via pairings
  - Idea: include "encrypted queries" in CRS

Crypto Compiler: First Attempt

Prover

$\pi =$ 4 3 1 2 8 3 1 2 1 9 3 1 6 1 2 1

$q_1 =$ 5 3 6 2 1 3 1 2 1 1 6 1 3 1 8 1

$q_2 =$ 7 3 1 2 4 3 1 2 7 1 3 1 7 1 2 1

$q_3 =$ 1 2 1 2 1 9 1 2 5 1 4 1 3 1 3 1

Verifier

a_1

a_2

a_3

x

ACC / REJ

Crypto Compiler: First Attempt

CRS

$q_1 =$	5	3	6	2	1	3	1	2	1	1	6	1	3	1	8	1
$q_2 =$	7	3	1	2	4	3	1	2	7	1	3	1	7	1	2	1
$q_3 =$	1	2	1	2	1	9	1	2	5	1	4	1	3	1	3	1

Prover

$\pi =$	4	3	1	2	8	3	1	2	1	9	3	1	6	1	2	1

Verifier

a_1

a_2

a_3

x

→

ACC / REJ

Crypto Compiler: First Attempt

CRS

$q_1 =$ [16 red boxes]

$q_2 =$ [16 red boxes] linearly homomorphic encryption

$q_3 =$ [16 red boxes]

Prover

$\pi =$ | 4 | 3 | 1 | 2 | 8 | 3 | 1 | 2 | 1 | 9 | 3 | 1 | 6 | 1 | 2 | 1 |

Verifier

a_1

a_2

a_3

x

→ ACC / REJ

Crypto Compiler: First Attempt

CRS

$q_1 =$

$q_2 =$ linearly homomorphic encryption

$q_3 =$

Prover

$\pi =$ 4 3 1 2 8 … 2 1 9 3 1 6 1 2 1

a_1

a_2

a_3

Problem 1: May allow more than just linear functions!

Solution 1: Assume it away: "linear-only encryption"

- A natural instance of targeted malleability [BSW12]
- Plausible for most natural public-key encryption schemes
  … including post-quantum ones [Reg05,BISW17]
- Win-win flavor

Crypto Compiler

CRS

$q_1 =$

$q_2 =$ linearly homomorphic encryption

$q_3 =$

Prover

$\pi =$ 4 3 1 … 3 1 2 1 9 3 1 6 1 2 1

Problem 2: Prover can apply different π_i to each q_i or even combine q_i

Solution 2: Compile LPCP into a proof system that resists this attack

- Linear Interactive Proof (LIP): 2-message IP with “linear-bounded” Prover
- IT compiler: LPCP → LIP via a random consistency check [BCIOP13]

Crypto Compiler

CRS

$q_1 =$

$q_2 =$ linearly homomorphic encryption

$q_3 =$

Prover

$\pi =$ 4 3 1 … 8 3 1 2 1 9 3 1 6 1 2 1

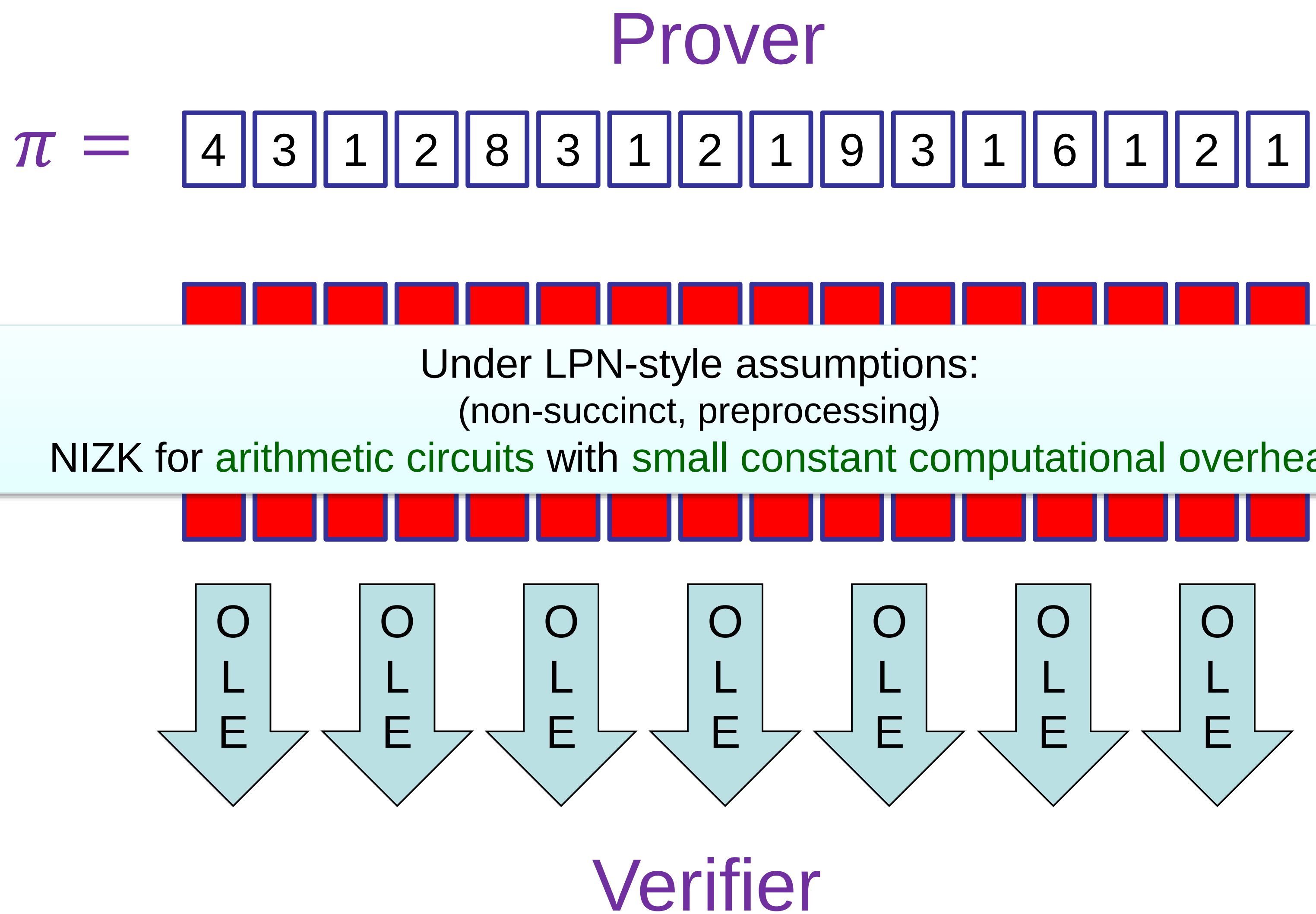
Problem 3: Only works in a designated-verifier setting

Solutions 3:

- Look for designated verifiers around your neighborhood
- LPCP with deg-2 decision + “bilinear groups” → public verification [Gro00,BCIOP03]

Alternative OLE-Based Compiler

[BCGI18,CDIKLOV19]



Under LPN-style assumptions:

(non-succinct, preprocessing)

NIZK for arithmetic circuits with small constant computational overhead

Combining the Two Relaxations: Linear IOP

Variant: ILC
[BCGGHJ17]

Prover

Variant: polynomial IOP
Crypto compilers via polynomial commitments
[ZGKPP17,WTsTW18,Set19,XZZPS19,BFZ19]

$\pi_1 =$ 1 3 1 2 1

$q_1 =$ 5 3 6 2 1 3 1 2 1 1 6 1 3 1 8 1

Verifier

Challenge

$\pi_2 =$ 1 3 1 2 1 3 1 2 1 1 3 1 3 1 2 1

$q_2 =$ 7 3 1 2 4 3 1 2 7 1 3 1 7 1 2 1

Challenge

Captures interactive proofs for P
[GKR08,RRR16]

Fully Linear PCP/IOP
[BBCGI19]

- Suppose statement x is known to prover but is
  - Secret-shared between two or more verifiers
  - Distributed between two or more verifiers
  - Encrypted or committed
- Tool: fully linear proof systems
  - Only allow linear access to x: q_i applies jointly to (x, π)
  - Meaningful even for "simple" languages and even if P=NP
  - Strong ZK: statement x remains hidden from verifiers
- Standard LPCPs are fully linear, but long proofs
  - Talk next week by Niv:
    Short ZK-FLPCPs for simple languages + applications

Fully Linear PCP/IOP
[BBCGI19]

- Sup... is
  - S...
  - Distributed between two or more verifiers
  - Encrypte... committed
- Too...
  - Only allow linear access to x: q_i applies jointly to (x,π)
  - Meaningful even for "simple" languages and even if P=NP
  - Strong ZK: statement x remains hidden from verifiers
- Standard LPCPs are fully linear, but long proofs
  - Talk next week by Niv:
    Short ZK-FLPCPs for simple languages + applications

Section 2 of **ePrint 2019/188:**
High-level overview of PCP types + crypto compilers

Also studied over general graphs in a distributed computing context [KKP10,KOS18,NPY18]

Conclusions

- Modular approach to efficient ZK/SNARG design
  - – Information-theoretic ZK-PCP + crypto compiler
    - point queries vs. linear queries
    - non-interactive vs. interactive
- rature
  - 09]
  - 4]

Constant computational overhead w/negligible error?
- Known for arithmetic computations with linear queries
- Open for Boolean circuits *or* with point queries
- Applies both to low-query PCPs and (arbitrary) ZK-PCPs

- Lots of ... for further progress
  - – Better PCPs (and lower bounds)

Better 1-query Linear PCP?
- Avoid PCP theorem
- Achieve strong soundness

Start with 1-query Fully Linear PCP?

Conclusions

- Modular approach to efficient ZK/SNARG design
  - Information-theoretic ZK-PCP + crypto compiler
    - point queries vs. linear queries
    - non-interactive vs. interactive
- Applies to most efficient protocols from the literature
  - ...
  - ...
- Lo...
  - Bette...
  - Better crypto compilers
  - Better IT compilers

- Better tools: subvector commitments, polynomial commitments,…
- Better compilers for general (Interactive) Linear PCP?
- Eliminate generic models and “non-falsifiable” assumptions

- ... queries in Linear PCPs?

The research leading to these results has received funding from the European Union's Horizon 2020 Research and Innovation Program under grant agreement

no. 742754 – ERC – NTSC

European Research Council