Concretely-Efficient Zero-Knowledge Arguments for Arithmetic Circuits and Their Application to Lattice-Based Cryptography

Ariel Nof

Bar-Ilan University

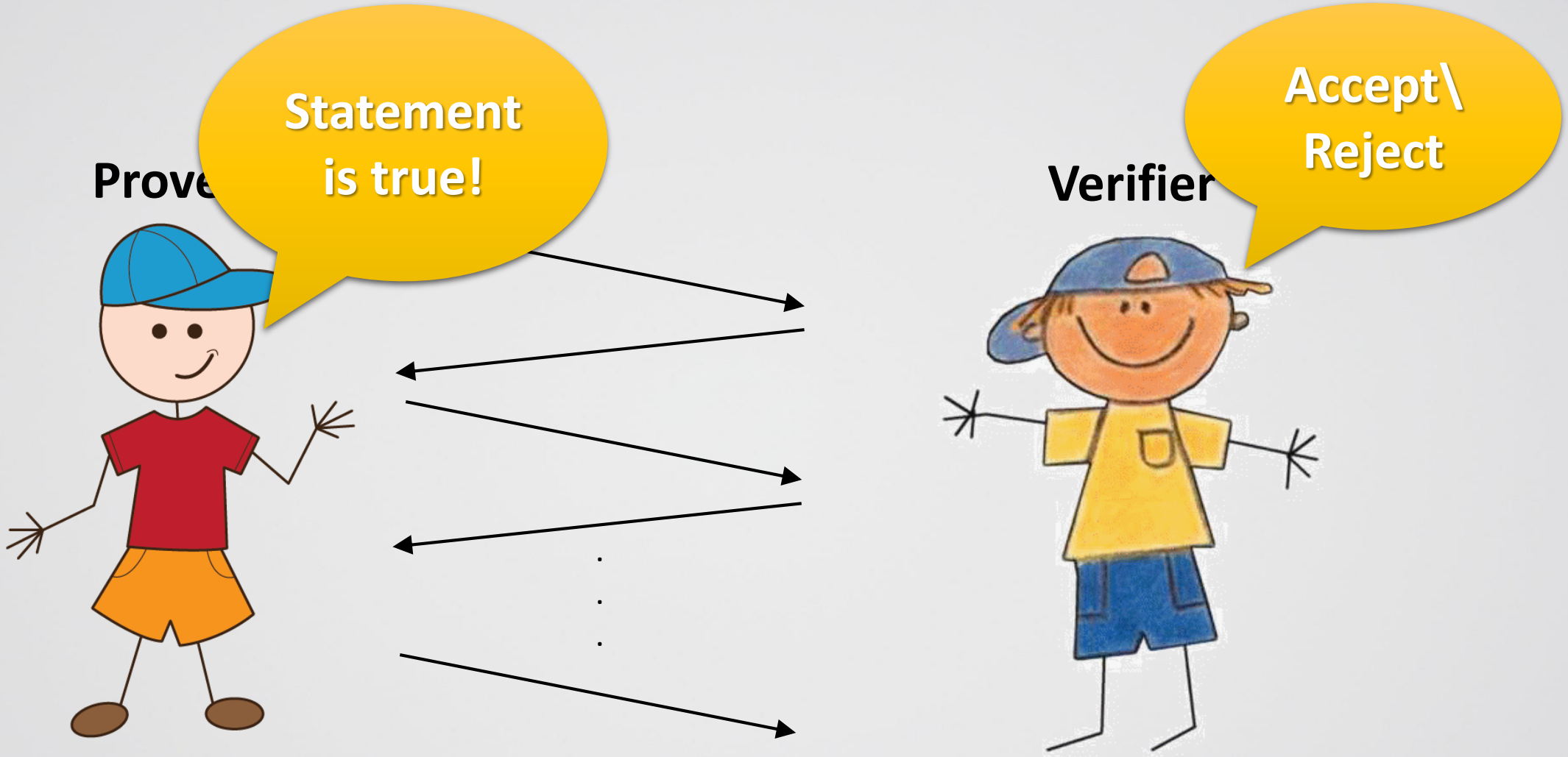
Carsten Baum

Aarhus University

TPMPC 2019

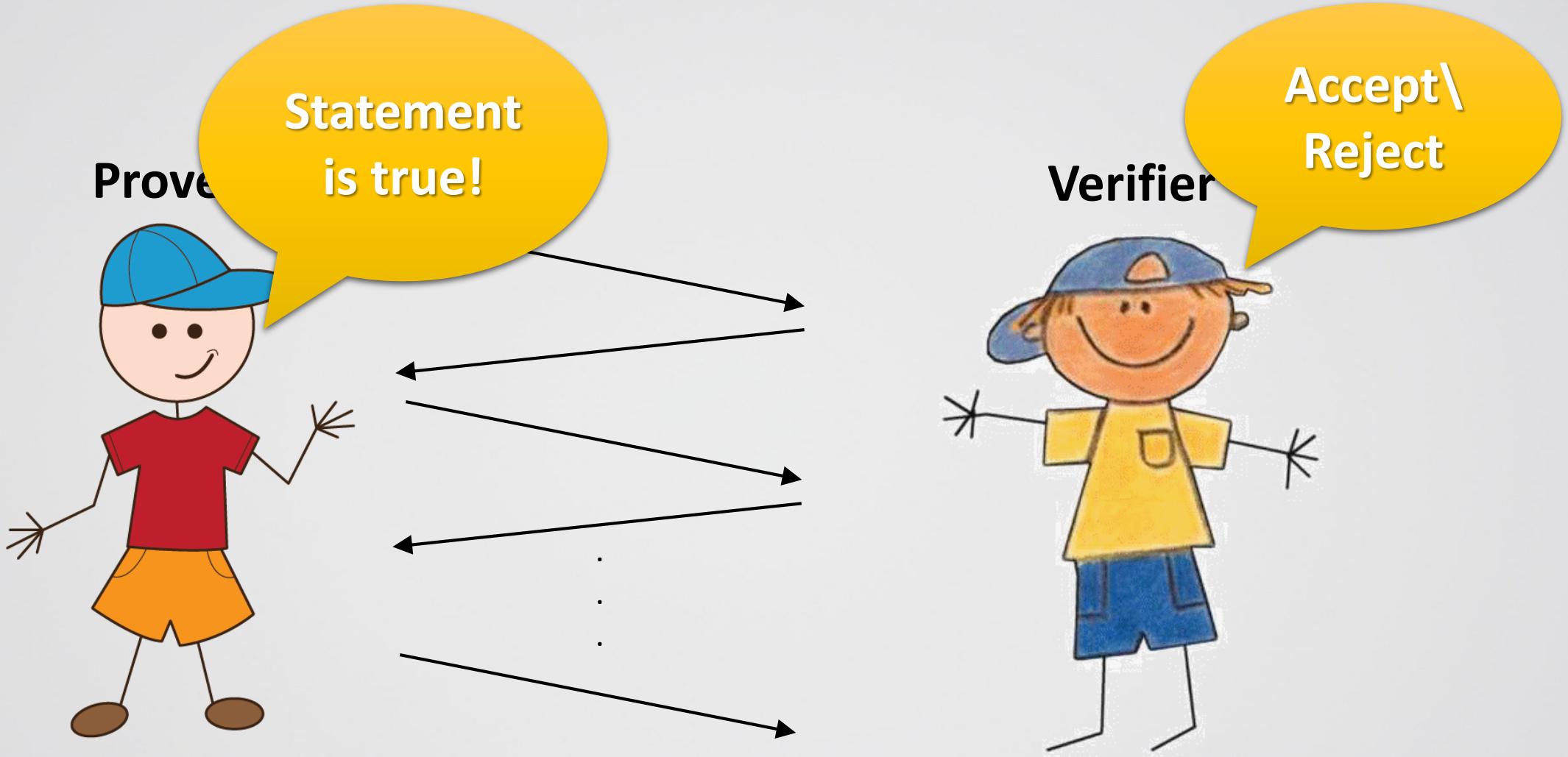
BIU Center for Research in Applied Cryptography and Cyber Security

Zero-Knowledge Argument of Knowledge (ZKAoK)



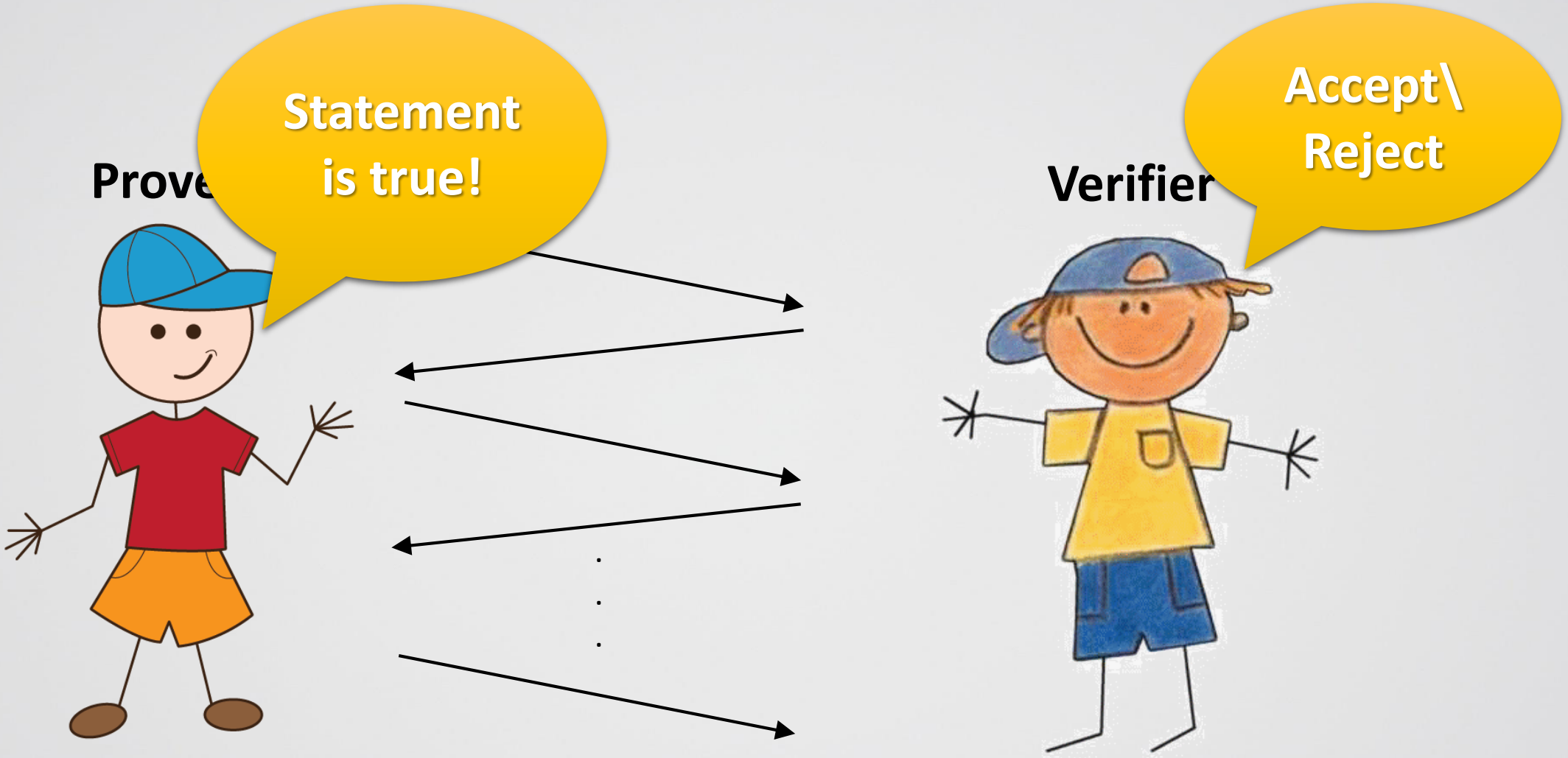
Requirements: Correctness, (knowledge) soundness, privacy (Zero-knowledge)

Zero-Knowledge Argument of Knowledge (ZKAoK)



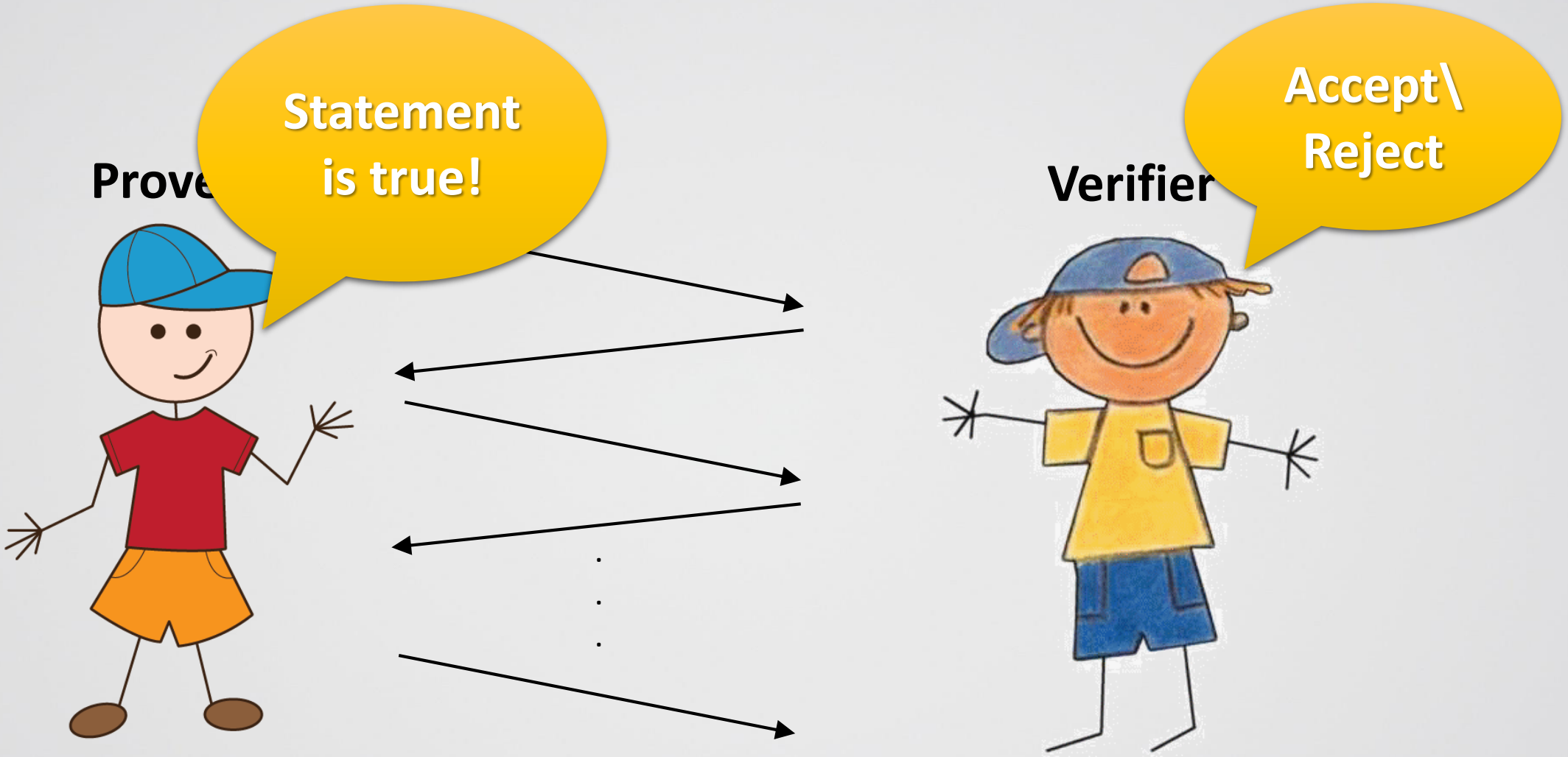
Requirements: **Correctness**, (knowledge) soundness, privacy (Zero-knowledge)

Zero-Knowledge Argument of Knowledge (ZKAoK)



Requirements: Correctness, **(knowledge) soundness**, privacy (Zero-knowledge)

Zero-Knowledge Argument of Knowledge (ZKAoK)



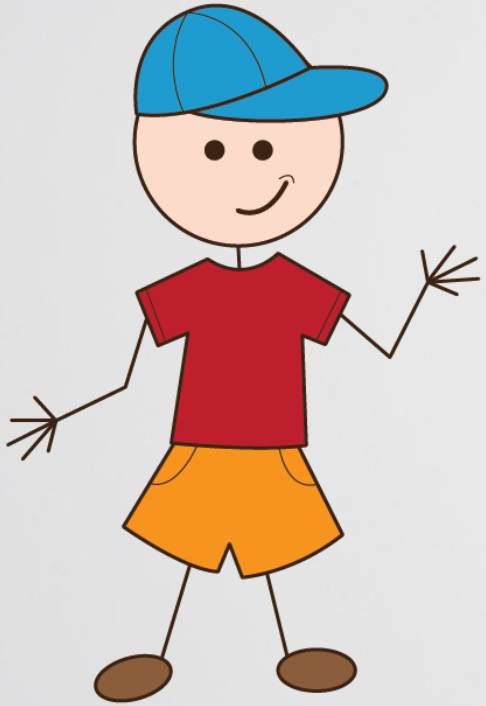
Requirements: Correctness, (knowledge) soundness, **privacy (Zero-knowledge)**

Zero-Knowledge Argument of Knowledge (ZKAoK)

Computationally bounded

C,w

Prover



$$C(w) \stackrel{?}{=} 0$$

C

Verifier

Different Solutions with Different Falvours…

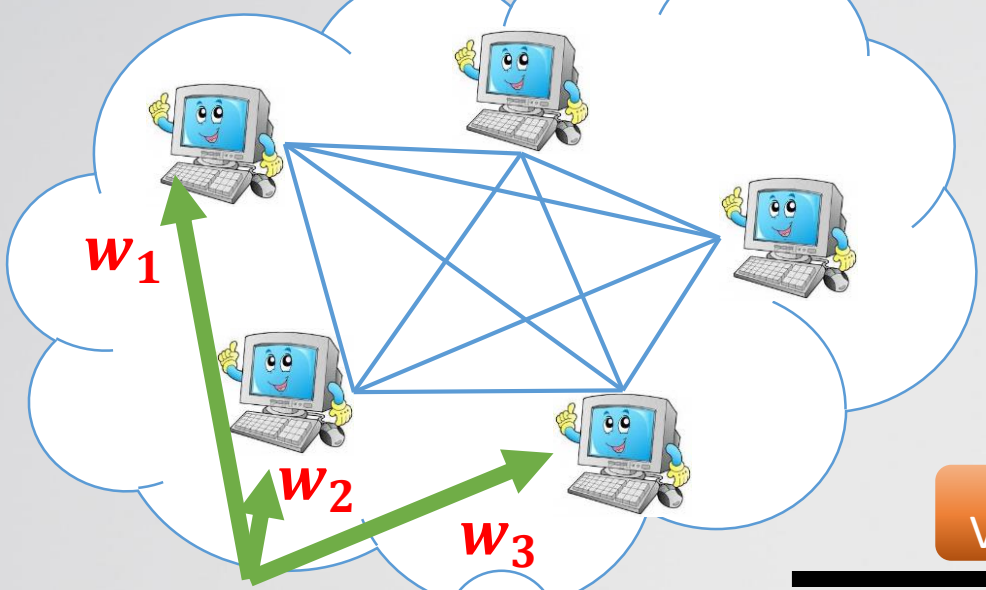
- Asymptotic VS. concrete efficiency
- Communication VS. Computation efficiency
- Trusted Setup?
- Assumptions?
- ….

Our Results

- **New** ZKAoK protocol
  - Based on the **MPC-in-the-head** technique
  - Both communication and computation efficient
  - Communication is **not** sub-linear, but has good constants
  - Symmetric-key primitives ➔ **post-quantum** secure
  - Can be made **non-interactive** via Fiat-Shamir

- Application to Lattice-based Cryptography
  - Proving knowledge for the Short Integer Solution (SIS) problem with **low prover time** and **small proof size**

MPC-in-the-Head (IKOS 07)



w

View ar… correct?

Accept\Reject?

View_1 • • • View_n

Challenge: open a random subset of t views

$View_{i_1}$ … $View_{i_t}$

Prover

Verifier

MPC-in-the-Head (IKOS 07)

- **zero-knowledge:**
  - Security against t corrupted parties ➔privacy
  - Semi-honest security suffices

- **Soundness**:
  - Follows from the fact that the prover commits to the views *before* the challenge is known

MPC-in-the-Head

- Introduced in IKOS[07]
- Implemented and optimized in ZKBoo[GMO16], ZKB++[CDG+17]
- Ligero [AHIV17] – **sub-linear communication** complexity!
- [KKW18] – MPC-in-the-Head in the **pre-processing model**
- **This work**- a MPC-in-the-Head in the **pre-processing model**

  which is more efficient for Arithmetic circuits

MPC-in-the-Head with Pre-Processing [KKW18]

- Pre-processing = generating input-independent randomness
  - Beaver triples that are used to multiply shared values
- Not straight forward in the context of MPC-in-the-head
  - Gives the prover more opportunities to cheat!

Pre-processing in MPC

- $[v]$ - sharing of v
- $([a],[b],[c])$ is a random multiplication triple if a, b are random and $c = a \cdot b$.

1st approach: Use random multiplication triple to multiply two shared values

- Multiply $[x],[y]$ using $([a],[b],[c])$
- Always succeeds!
- Requires $([a],[b],[c])$ to be correct
- Deterministic process

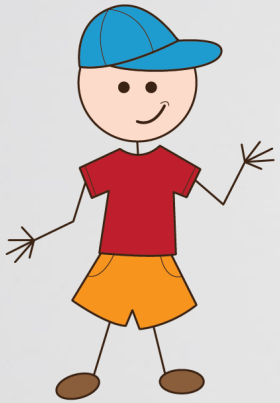
2nd approach: Verify multiplication triple correctness using a second triple

- Verify $[x],[y],[z]$ using $([a],[b],[c])$
- fails w.p. $1/|F|$
- $([a],[b],[c])$ **not** need to be correct!
- Randomized process

MPC-in-the-head: 1st approach [KKW18]

Cut & Choose

Prover



Verifier

Commit to random Multiplication triples

Challenge: subset of triples to open

1. De-commit the chosen subset
2. Simulate the on-line computation using the unopened triples
3. Commit to the views of the parties

Challenge: subset of views to open

De-commit the chosen views

1. Triples:
2. ...stent?
   ...put: correct?

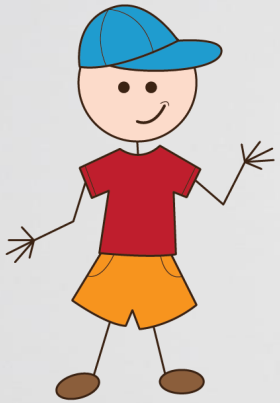
Accept\Reject?

MPC-in-the-head: 2nd approach [Our]

Sacrificing

Commit to:
1. random Multiplication triples
2. Sharings over all wires of the circuit

Prover



Verifier

Challenge: randomness for the verification protocol

Simulate the verification protocol.
Send commitments for the parties' views

Challenge: subset of views to open

De-commit the chosen views

1. Views: consisten
2. Verificati
   O
   output:
   correct?

Accept\Reject?

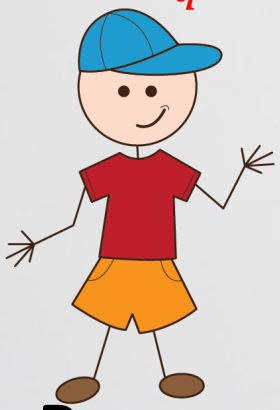
MPC-in-the-Head in the pre-processing Model: Two Approaches

- **Cut-and-Choose** VS. **Sacrificing**
- Both have **similar communication cost** per multiplication gate
  - Addition and multiplication-by-a-constant gates are **for free**!
- Who is better?
  - Circuits over Rings: Sacrificing doesn't work…
  - Circuits over large fields:
    - **Soundness of the "Sacrificing" approach is better** ➔
      Less repetitions of the MPC simulation is required ➔
      Less overall communication!

Application: The SIS problem

- A crucial building block for post-quantum lattice-based cryptography

$\boldsymbol{A} \in Z_q^{n \times m}$
$\boldsymbol{t} \in Z_q^n$
$\beta \in \mathbb{N}$
$s \in Z_q^m$



Prover

1. $\mathbf{A} \cdot \mathrm{s} = \mathrm{t}$
2. $\left|\left|\mathrm{s}\right|\right|_\infty \leq \beta$

$\boldsymbol{A} \in Z_q^{n \times m}$
$\boldsymbol{t} \in Z_q^n$
$\beta \in \mathbb{N}$

Verifier

The Binary SIS problem

1. $\mathbf{A} \cdot \mathrm{s} = \mathrm{t}$
2. $s \in \{0,1\}^{\mathrm{m}}$

- The first condition consists of multiplying a secret vector with a public matrix
  ➔ using our scheme this parts is done **for free**!
- The second condition:

$$s_i \in \{0,1\} \Leftrightarrow s_i(s_i - 1) = 0$$

- **We obtain a circuit of m multiplication gates**

The General SIS problem

1. $\mathbf{A} \cdot \mathrm{s} = \mathrm{t}$
2. $\left|\left|\mathrm{s}\right|\right|_{\infty} \leq \beta$

- The first condition consists of multiplying a secret vector with a public matrix
  ➔ using our scheme this parts is done **for free**!
- The second condition:

$$s_i \in \{-\beta, \beta\} \Leftrightarrow s_i + \beta \in \{0, 2\beta\}$$

- Set $s_i = \sum_{i=1}^{\log(2\beta)} 2^i \cdot s_{i,j}$ and prove that $s_{i,j} \in \{0,1\}$
- **We obtain a circuit of $\boldsymbol{m \cdot log(\beta)}$ multiplication gates**

Can we do better?

- Can we reduce communication furthermore?
  - i.e., design a circuit with less multiplication gates
- Yes, we can!
  - For Binary-SIS problem: a circuit with just **one multiplication gate**
  - For General SIS problem: a circuit with just **one multiplication gate** but the proof is only approximate (it has a slack which depends on the size of the input)
- How?
  - Using "**Circuit Sampling**"

Circuit Sampling on the Fly

- The prover and the verifier “negotiate’’ on the final structure of the circuit
- High-level of the process:
  1. The prover commits to the input and the randomness used in each MPC instance
  2. The prover and the verifier **sample together a circuit**
  3. Continue with the proof as before
- Soundness:
  - The prover cannot ``tailor’’ a witness into the chosen circuit
  - Once the circuit was picked, soundness is the same as before

- More details - in the paper….

Implementation & Experimental Results

1. $\mathbf{A} \cdot s = t$
2. $s \in \{0,1\}^m$

$\boldsymbol{A} \in Z_q^{n \times m}$
$\boldsymbol{t} \in Z_q^n$

- We ran experiments for the Binary-SIS problem only
- The experiments was ran on Amazon C5.9xlarge instances using two standard servers. The network bandwidth between the nodes is 10Gpbs.
- We tested our protocol for various SIS parameters:
  - $\log_2(q) = 15, 31, 59, 61$
  - $n = 256, 512, 512$
  - $m = 1024, 2048, 4096$
- To the best of our knowledge this is the first implementation of ZKAoK for this problem.

Implementation & Experimental Results

- For the ``toughest'' SIS instance (large field and matrix **A**):
  - Our protocol run in **1.2 second** when utilizing one thread and in **250ms** when utilizing 32 threads
- Choosing the number of parties in the MPC instances
  - More parties ➔ less communication
  - Less parties ➔ low running time
- Matrix multiplication becomes a bottle-neck
  - We introduce an optimization to batch this on the verifier side
  - Using structured matrices is a possible improvement direction

Questions?

https://eprint.iacr.org/2019/532