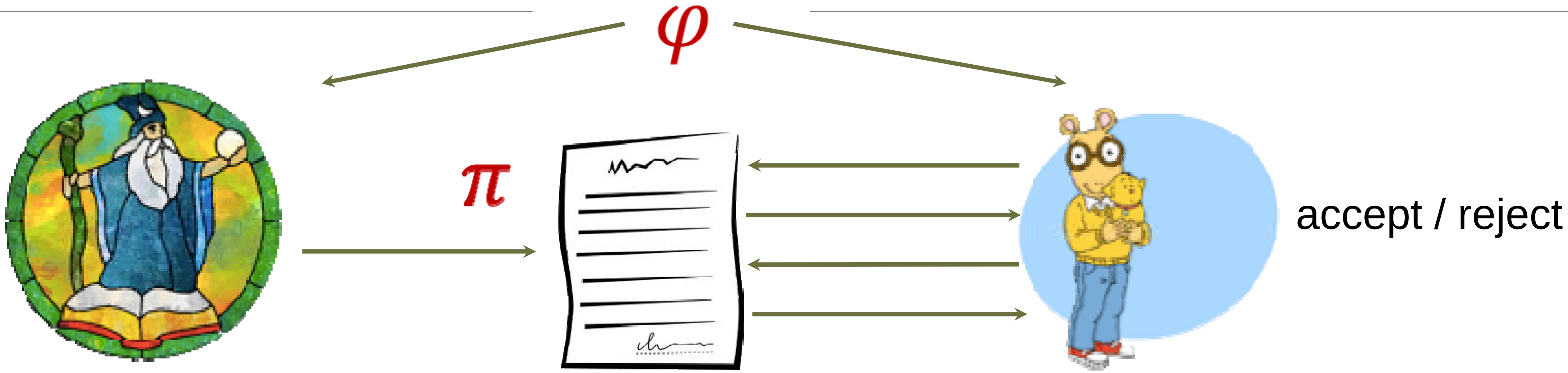
Languages with Efficient Zero-Knowledge **PC**
are in **SZK**

MOHAMMAD MAHMOODY (CORNELL)

DAVID XIAO (LIAFA)

Probabilistically Checkable Proofs (PCPs)

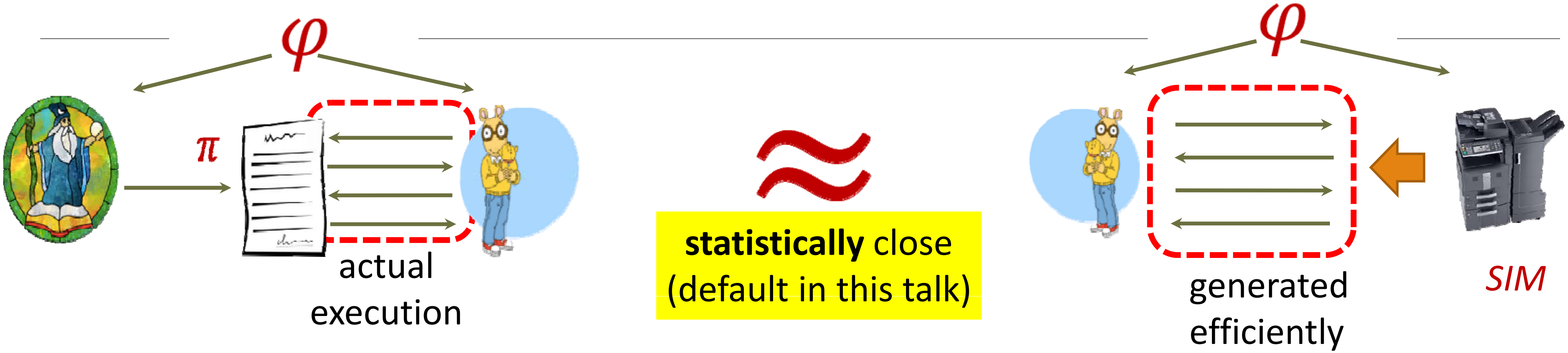


Hybrid of "traditional" and "interactive" proofs:

- Prover "writes" a "proof" π (of length $|\pi| = 2^{|\varphi|}$) + Verifier checks $poly(|\varphi|)$ bits of π
- **Completeness**: if $\varphi \in L$ there is some acceptable proof π
- **Soundness**: if $\varphi \notin L$ any proof rejected with high prob.

[Babai-Fortnow-Lund'90] : **NEXP** provable using PCPs

Zero-Knowledge PCPs



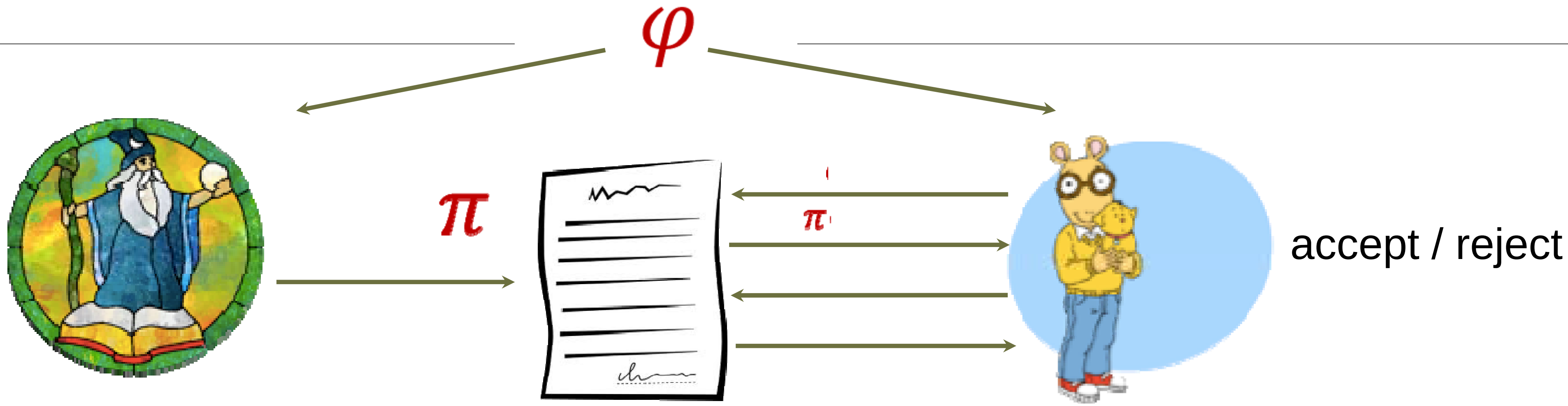
Def: View of any efficient verifier can be efficiently “simulated” (similar to ZK interactive proofs)

- **Harder** to achieve zero-knowledge PCPs (than provers) verifier can read **any** PCP answers.
- **Easier** to achieve sound PCPs (than provers).

[Kilian-Petrank-Tardos’97] ***NEXP*** has (statistical) zero-knowledge PCPs

- Inherently of super-polynomial length (even for **NP**)

Efficient Zero-Knowledge PCPs



Efficient PCP:

- Given any query q, the answer $\pi(q)$ can be computed in time $poly(|\varphi|)$
- Meaningful even if PCP has super-polynomial length

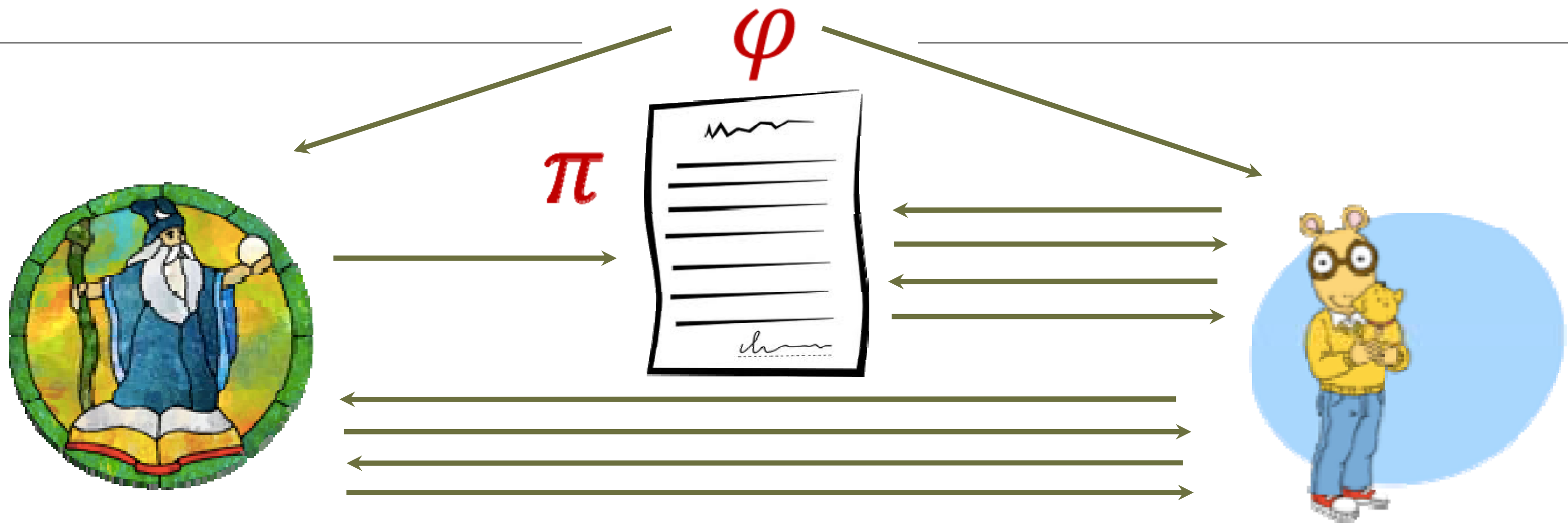
The zero-knowledge PCP of [KPT'97] is not efficient even for **NP**

Main Question: Are there **efficient** (statistical) ZK PCPs for **NP** ?

Main Question: Are there **efficient** (statistical) ZK PCPs for **NP** ?

Motivation: Basing Crypto on Tamper Proof Hardware

[Kat07, MS08, CGS08, GKR08, GISVW10, Kol10, GIMS10, ...]

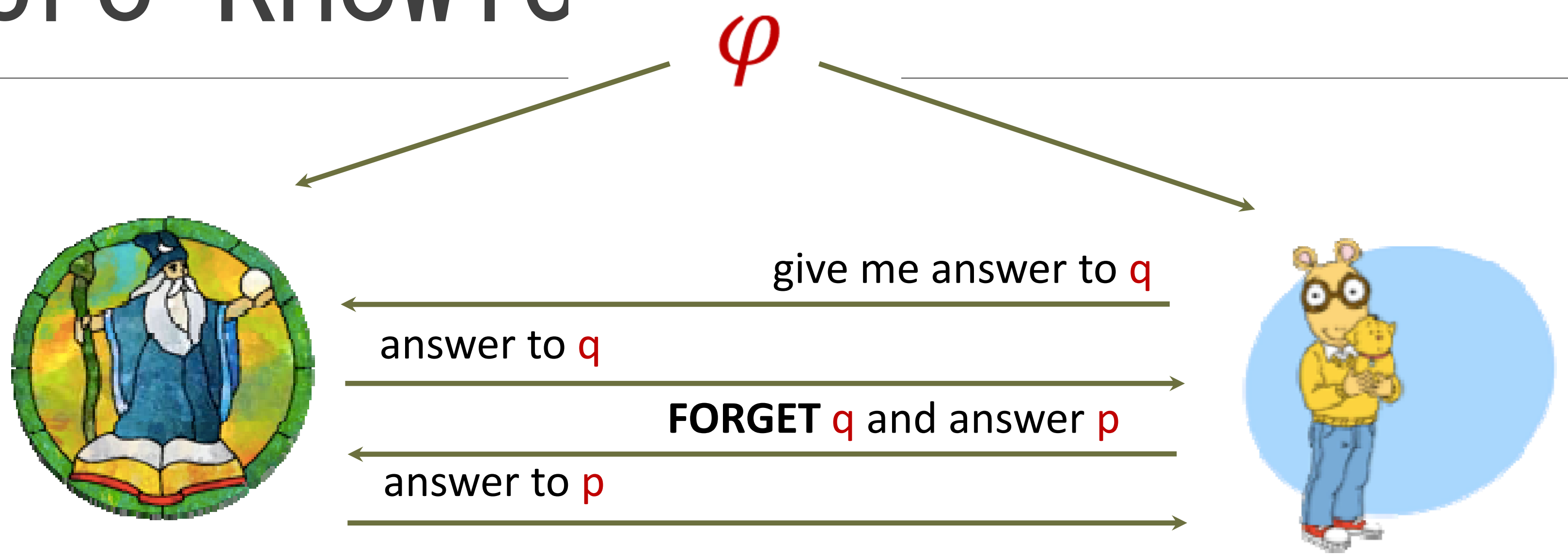


[Goyal-Ishai-M-Sahai'10] Unconditional (statistical) zero-knowledge for **NP** in Interactive PCP model [KR'08] : Prover generates an efficient PCP π + answers 2 challe

Implies zero-knowledge for **NP** using 1 stateless **hardware** and 4 messages

Left open: Using only one stateless hardware ?
Equivalent to our main question: Are there efficient ZK PCPs for **NP** ?

Motivation: Resettable Statistical Zero-Knowledge



Resettable ZK [CGGM'00] : Verifier can "reset" (rewind) prover to previous state

[Garg-Ostrovsky-Visconti-Wadia'12] Quad-Resid. has an **efficient** single prover **resettable** statistical **zero-knowledge.**

Corollary: Efficient statistical zero-knowledge PCPs exist for Quad-Resid

Proof: Let $\pi(q_1, q_2, \ldots, q_t)$ returns the sequence of answers to $a_1, a_2, \ldots, a_t$

Limits of Efficient (statistical) Zero-Knowledge PCPs?

Main Question: Are there **efficient** ZK PCPs for **NP** ?

[Ishai-M-Sahai'12] Any language with an efficient ZK PCP using a **non-adaptive** verifier is in co-**AM**

Corollary: No efficient ZK for **NP** using a **non-adaptive** verifier unless the polynomial-time hierarchy collapses [BHZ'87]

Our Result

Theorem: Any language L with an efficient statistical ZK PCP has a statistical zero-knowledge **single-prover** proof system (i.e. $L \in \mathbf{SZK}$)

- Removes the non-adaptivity constraint of [IMS'12]
- Improves the co$\mathbf{AM}$ bound to $\mathbf{SZK} \subseteq \text{co}\mathbf{AM}$

Corollary: No **NP**-complete language has an efficient statistical ZK PCP unless the polynomial-time hierarchy collapses.

Ideas behind the proof

Approach of [IMS' 12]

[IMS'12]: If language L has an efficient statistical ZK PCP with a **non-adaptive** verifier ⇒ L ∈ **AM** ∩ coAM.

Goal: Decide both L and $\overline{L}$ with help of an untrusted prover in $O(1)$ rour
Have: Statistical ZK Simulator SIM for PCP

Protocol: for both L and $\overline{L}$:

1. "Extract" a PCP π from $SIM(\varphi)$
2. Run PCP verifier V_{PCP} over π

With help of **untrusted** prover
[GS'89, GVW'01, HMX'10]
Rely on non-adaptivity of V_{PCP}

Naïve Approach

Theorem: Any language L with an efficient statistical ZK PCP has a statistical zero-knowledge **single-prover** proof system (i.e. $L \in \mathbf{SZK}$)

Given φ : want to find out $\varphi \in L$ or $\varphi \notin L$

- Run $SIM(\varphi)$ to generate $view$ of Verifier
- Decide based on $view$

✓ : If $\varphi \in L$ we will get $view = accept$ by ZK

➢ : If $\varphi \notin L$ might also generate $view = accept$

Our Approach

Theorem: Any language L with an efficient statistical ZK PCP has a statistical zero-knowledge **single-prover** proof system (i.e. $L \in \mathbf{SZK}$)

Idea: Naïve approach with a few more "checks" in **SZK**

- Run $SIM(\varphi)$ to generate $view$ of Verifier
- $view = (r, q_1, a_1, \dots, q_n, a_n)$
- r = randomness, q_i = i'th query , a_i = i'th answer

Our Approach

Idea: Naïve approach with a few more "checks" in **SZK**

- Run $SIM(\varphi)$ to generate $view$ of Verifier
- $view = (r, q_1, a_1, \ldots, q_n, a_n)$
- r = randomness, $q_i = i$'th query , $a_i = i$'th answer

Our Approach

Idea: Naïve approach with a few more "checks" in **SZK**

- Run $SIM(\varphi)$ to generate $view$ of Verifier
- $view = (r, q_1, a_1, \ldots, q_n, a_n)$
- $r =$ [...] h answer
- Let [...] nd a its answer

Conditional Entropy Approximation: in **SZK** [Vadhan'04]

Main Obs[...]**vation:**
If $H(\, a \mid q\,) \approx$ small $\Rightarrow$ answers in $view$ close to some PCP $\Rightarrow$ **Soundness**

Completeness ? By running k copies of V_{PCP} make $H(\, a \mid q\,) \approx \frac{poly(n)}{k}$

$poly(n)$: # random bits of **efficient** algorithm computing PCP π

Putting Things Together

Let **malicious** V^k be k executions of honest verifier V_{PCP}

- Run $SIM(\varphi)$ to generate $(view_1, \dots, view_k)$ of Verifier
- $view_k = (r, q_1, a_1, \dots, q_n, a_n)$
- r = randomness, q_i = i'th query , a_i = i'th answer
- Let $q = q_i$ for **unknown** random $i \leftarrow [n]$ and a its answer

Check that:

1. $view_k$ is accept.
2. $H(a \mid view_1, \dots view_{k-1}, q) \approx$ small
3. $H(r \mid view_1, \dots view_{k-1}) \approx$ large

$O(1)$ checks in **SZK** can a
be done in **SZK** [Vadhan'

Summary

Theorem: No efficient statistical ZK PCP for **NP** unless polynomial-ti
hierarchy collapses -- removing the non-adaptivity barrier of [IMS'1

Open: Characterize languages with efficient ZK PCPs.
Conjecture: All of **SZK** (sufficient to make compiler of [GOVW] effici

Open: Number of messages (2 or 3 or 4) needed in addition to an efficient PCP (hardware token) to get statistical zero-knowledge for

Thank You !