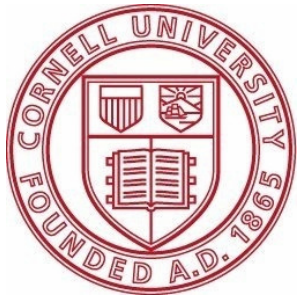
On the Composition of Public-Coin Zero-Knowledge Protocols

Rafael Pass (Cornell)

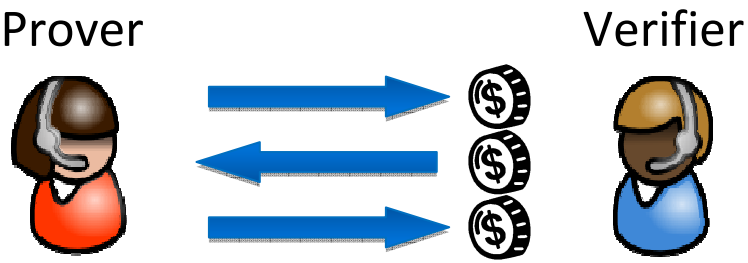
Wei-Lung Dustin Tseng (Cornell)

Douglas Wikström (KTH)



Zero Knowledge [GMR85]

- Interactive protocol between a Prover and a Verifier where the Verifier **learns nothing** except the proof statement



- Fundamental construct of cryptography
- Used in secure MPC, authentication, etc, etc

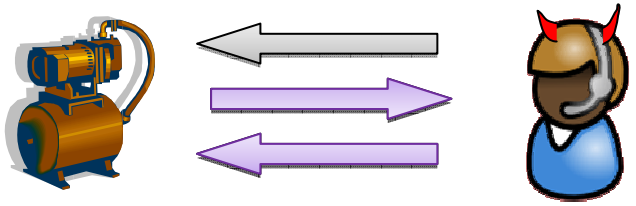
Zero Knowledge [GMR85]

- For every PPT V* (adversary) there is a PPT **simulator** S:

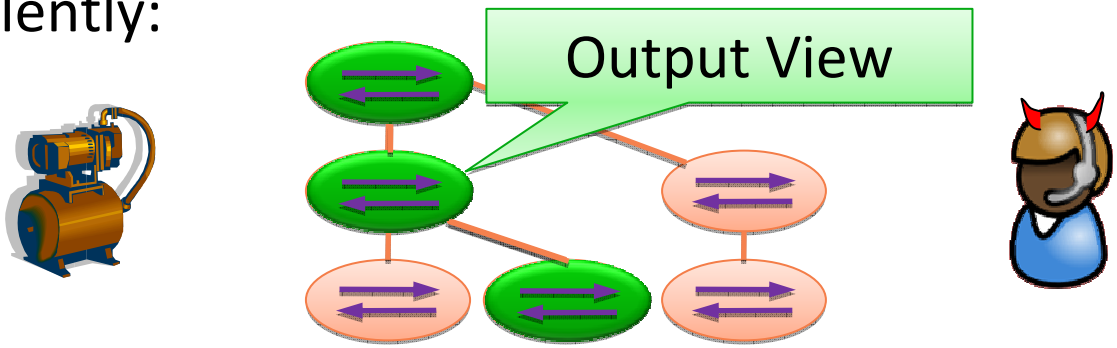
Indistinguishable

Black-Box Zero Knowledge [GO90]

- Universal S interacts with and **rewinds** V*

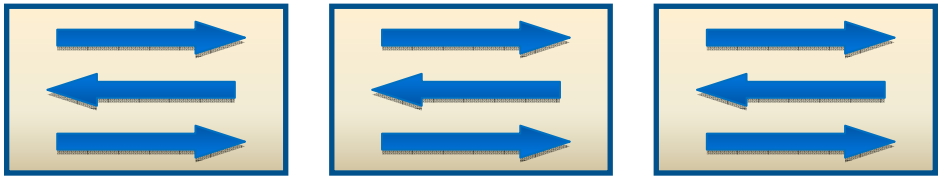


Equivalently:

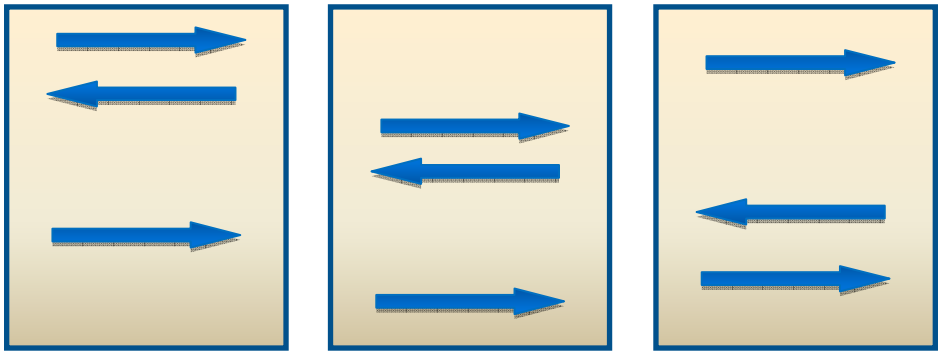


  - Most known and all practical ZK are BB
  - This talk: Focus on **BB ZK**

Composition of ZK [GKr90]



Parallel [FS90, GKr90]



Concurrent [FS90, DNS04]

- Do ZK protocols stay ZK when composed?

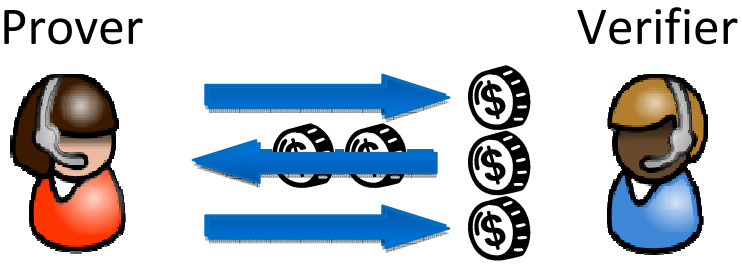
Composition of ZK [GKr90]

- **In general**: ZK breaks even under 2 parallel executions [FS90, GKr90]
- **Specific protocols**:
  - Secure under both parallel and concurrent composition (e.g., [GKa96, FS90, RK99, KP01, PRS02])
  - But these protocols use something new:

Private Coins

Public vs. Private Coins

- Public-coin:
- Private-coin:



- The original ZK protocols are all public-coin [GMR85,GMW91, Blum87]
- Why care about public-coin protocols?
  - Theory:
    - Understand original protocols
    - e.g. “IP(Poly) = AM(Poly)” [GS86]
  - Practice:
    - Simpler to implement
    - V resilient to leakage and side channel attacks

The Question:

*Are **private coins** **necessary** for composing ZK (even just) in parallel?*

- First studied by Goldreich-Krawczyk in 1990
- Partial result: No **constant round** public-coin BB ZK w/ neg. soundness error ($L \notin BPP$)
  - Known O(1) round public-coin BB ZK (with big soundness error) not secure in parallel

Our Results

1. Any public-coin protocol is **not BBZK** if repeated sufficiently in parallel ($L \notin BPP$).

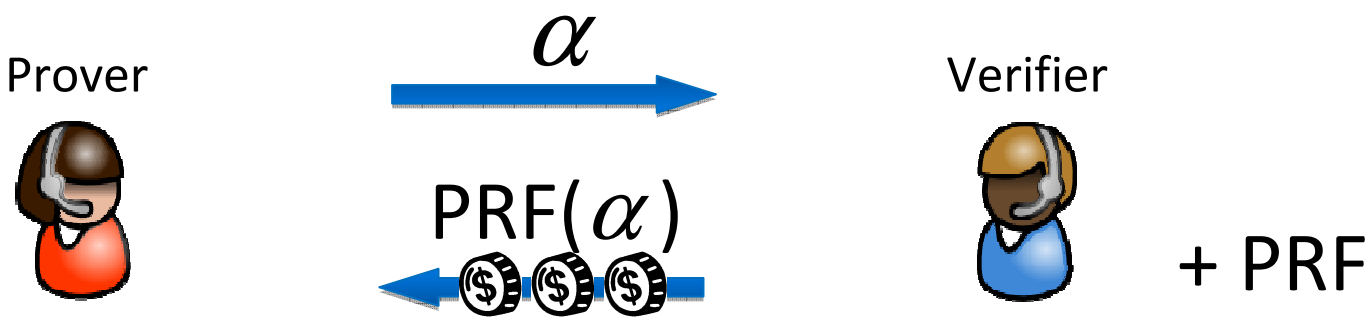
2. For every ***m***, there is a public-coin proof for NP that is BBZK **up to *m* concurrent sessions**, assuming OWF.

[Bar01]: Public-coin constant round bounded-concurrent **non-BB** ZK **argument** assuming **CRH**.

The Goldreich-Krawczyk framework

[GKr90]: If the verifier uses PRF to generate its messages in a constant round public-coin protocol

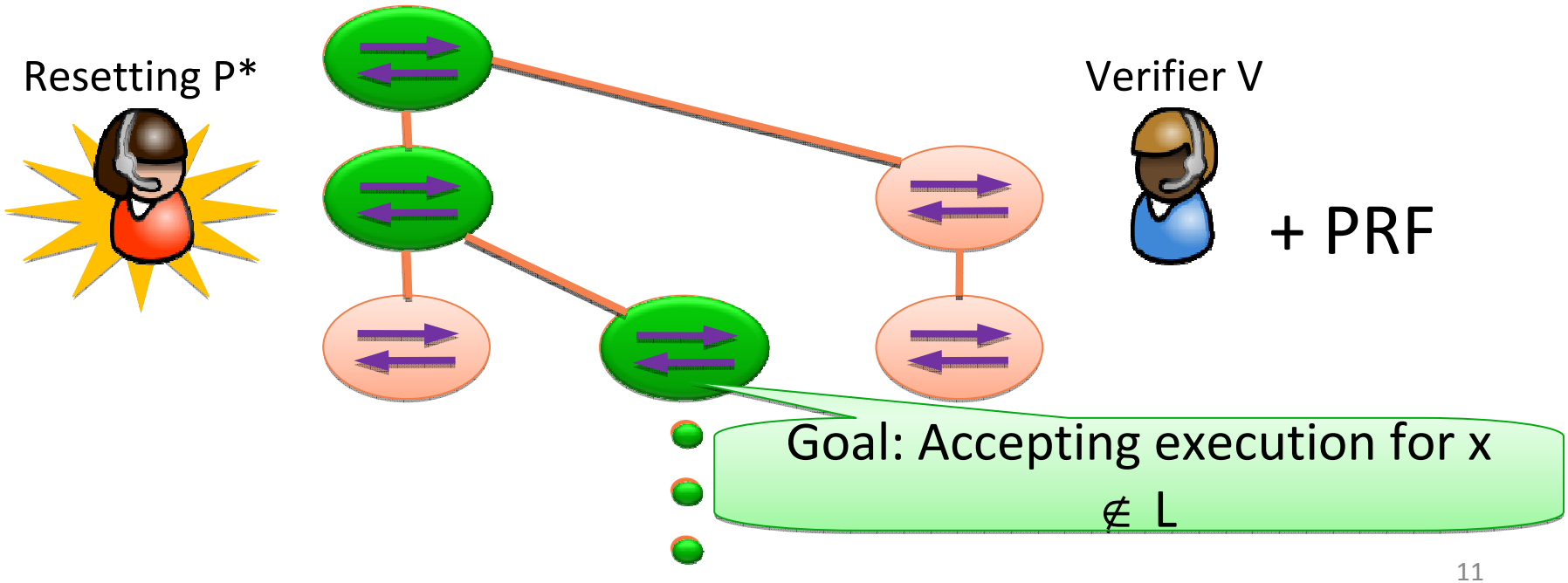
$\rightarrow$ Protocol is **resettably-sound** [BGGL01]



The Goldreich-Krawczyk framework

[GKr90]: If the verifier uses PRF to generates it messages in a constant round public-coin protocol

$\rightarrow$ Protocol is **resettably-sound** [BGGL01]



The Goldreich-Krawczyk framework

[GKr90]: If the verifier uses PRF to generates it messages in a constant round public-coin protocol

$\rightarrow$ Protocol is **resettably-sound** [BGGL01]

- If protocol is resettably-sound and BB ZK for L
  - $\rightarrow$ $L \in$ BPP (decided by S) [GK90, BGGL01]:
  - $x \in L \rightarrow S(x)$ gives accepting view (ZK)
  - $x \notin L \rightarrow S(x)$ gives rejecting view (resettable-sound)

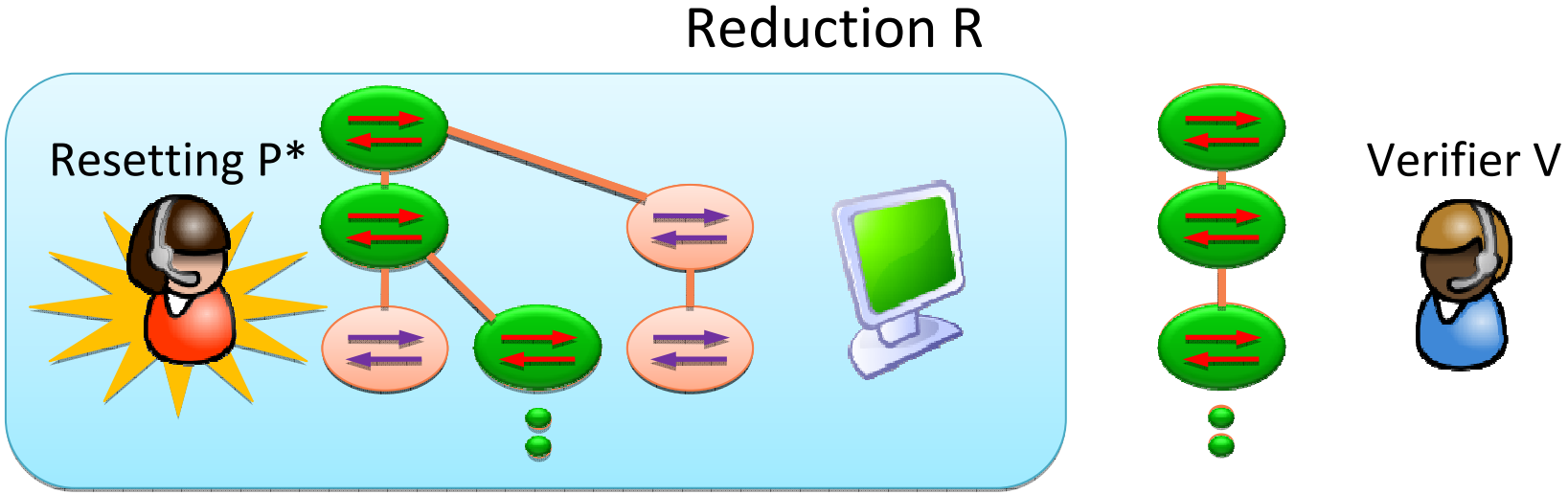
Main Lemma

Any public-coin protocol (where V uses PRF for its messages) is **resettably-sound** when repeated sufficiently in parallel.

- Compare with soundness amplification
  - Recent work: Parallel repetition amplifies soundness of public-coin arguments [PV07, HPPW08]:
    - From $\varepsilon \rightarrow \varepsilon^{poly(n)}$
  - Our work: **"Quality"** of soundness also improves
    - From "standard sound" $\rightarrow$ "resettably sound"
  - Can use soundness amplification techniques

Proof Idea

- Reduction R: Resettable P* $\rightarrow$ normal P



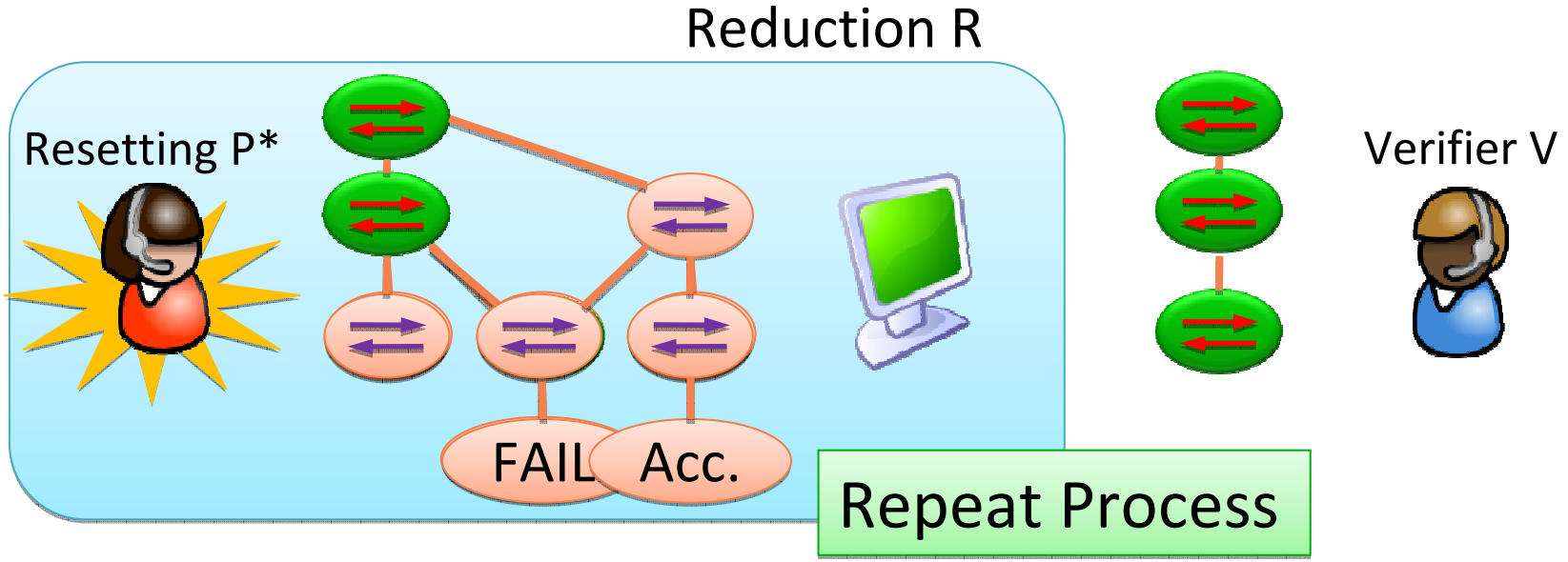
- R tries to forward messages that P* utilize for an accepting execution
  - Possible to continue simulation due to public-coin

Which Message to Forward?

- [GKr90] For constant round protocols, choose random messages to forward
  - Guess correctly w.p. 1/poly each round
  - Doesn't work when there are more rounds
- Our approach:
  - Do a **test run** to see which msg "should've been" forwarded. Forward it and continue simulation
  - If P* doesn't use forwarded msg, **rewind** P* until it does

Example

Start: Two rounds are already forwarded



Case: Forwarded message is accepting view

→ Rewind next message to forward

The Reduction Again

1. In a test run of P*, find the msg used by P* to form an accepting view.
2. Forward the msg to V and receive a fixed reply.
3. Keep rewinding P* until the forwarded msg is used in an accepting view
   - The next msg in view gets forwarded. Repeat.

Reduction idea analogous to [HPPW08]

Reduction always works! **Is it poly time?**

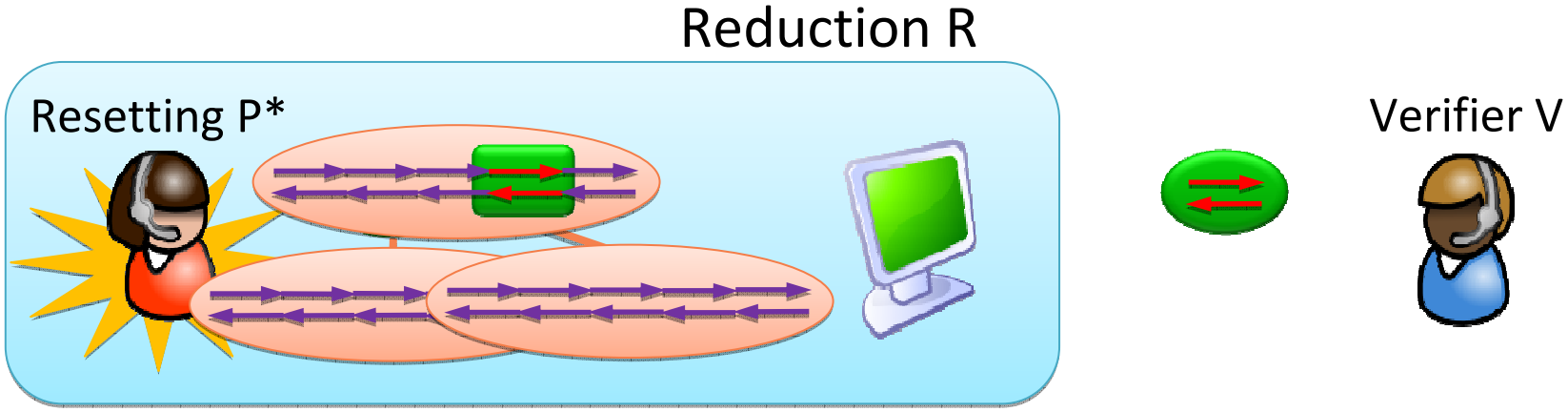
Analysis Sketch

- If we can rewind external V:
  - Case: P* chooses which branch to use in view randomly.
    $\rightarrow$ Then poly rewinds are enough
  - This is actually the **worst case**

- But we can't rewind external V:
  - Forwarded messages are **fixed**. Might fix a **BAD** message
  - Reduction: Resettable**standalone** P*$\rightarrow$normal **standalone** P
  - New picture!

Analysis Sketch

- Can **almost rewind the Verifier**
- Results in a statistically close distribution!
  - Technically shown by relying on Raz's Lemma
  - Technique used in soundness amplification of 2-prover games [Raz98] and public-coin arguments [HPPW08]



Conclusion

- Any public-coin protocol, with enough parallel repetitions, is **resettably-sound**

  $\rightarrow$ so **not BB ZK** unless L $\in$ BPP

- Elucidate connection between **hardness amplification** and **BB ZK lower bounds**

  – **New set of techniques for BB lower bounds**

Corollary

- Bare Public-Key setup
  - More efficient (private-coin) concurrent ZK
  - Model studied in the soundness amplification literature [IW97, BIN97, HPPW08]
- Using [BIN97, HPPW08] techniques, we can extend our impossibility result to BPK too

Thank You!