NON-BLACK-BOX ZK
(Barak's Protocol)

ALON ROSEN **IDC HERZLIYA**

FACT FOUNDATIONS & APPLICATIONS of CRYPTOGRAPHIC THEORY

BIU WINTER SCHOOL | February 2019

The Goal

Goal: construct CZK argument $\forall L \in \mathrm{NP}$

- with negligible soundness
- a constant number of rounds
- and public-coin

Need to address:

- How to use V^*'s code (BB impossibility)
- V^*'s running time is not a-priori bounded

Non-BB ZK Arguments for NP

- No $L \notin \mathrm{BPP}$ has a black-box ZK protocol that is:
    - constant-round
    - negligible-soundness
    - public-coin
- So for $L \notin \mathrm{BPP}$ must use a non-black box simulator
- On the one hand, $\forall V^* \, \exists S$ should be easier than $\exists S \, \forall V^*$
- On the other hand, where do we even begin?
    - Reverse engineering V^* is difficult!
    - Key insight: there is no need to reverse engineer
    - Enough for S to prove that he possesses V^*'s code

Non-BB ZK Arguments for NP

<u>Theorem [B'01]</u>: If CRH exist, every $L \in \mathrm{NP}$ has a constant-round, public-coin, negligible-soundness, ZK argument

- <u>Idea</u>: enable usage of verifier's code as a "fake" witness
- In the real proof, the code is V's random tape
- In simulation, the code is V^*'s "next-message function"
- Since P does not have access to V's random tape in real interactions, this will not harm soundness
- The simulator S, on the other hand, will be always able to make verifier accept since it obtains V^*'s code as input

Collision-Resistant Hash Functions

Definition: $H_k: \{0,1\}^* \rightarrow \{0,1\}^k$ **is** (t, ε)-**CRH if** $\forall$**time-**t A

$$Pr[A \text{ finds a collision in } h \in_R H_k] \leq \varepsilon$$

Collision: $x \neq x'$ **such that** $h(x) = h(x')$

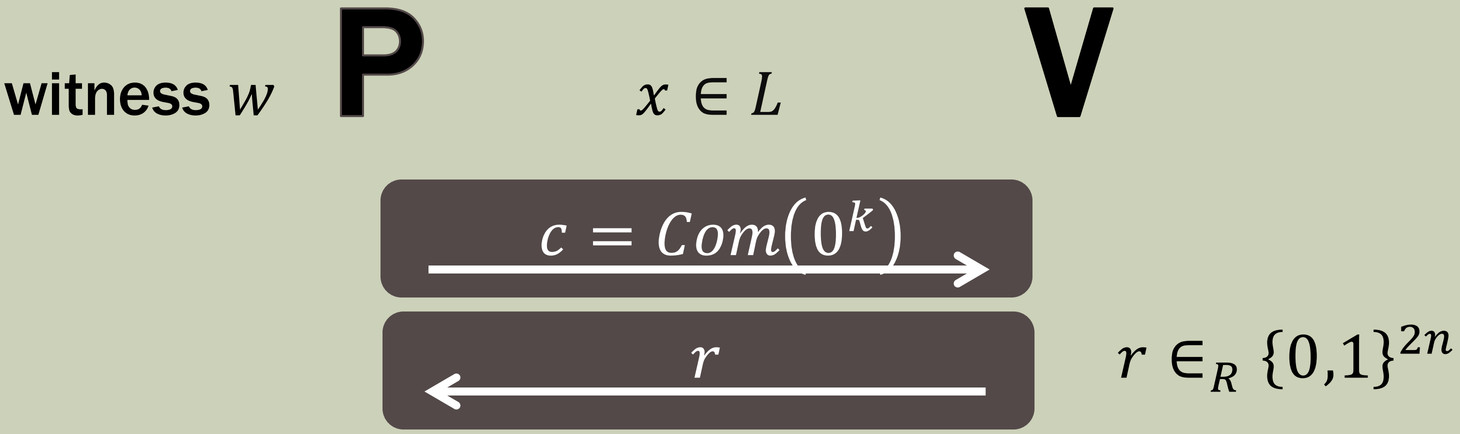
Candidate CRHs:

- **Discrete-log-based:** $g^{x_L} h^{x_R} \; mod \; P$
- **SIS:** $Ax \; mod \; q$
- **SHA:** $h(x_L, x_R)$

Later: $H_k: \{0,1\}^* \rightarrow \{0,1\}^k$ **from** $h: \{0,1\}^{2k} \rightarrow \{0,1\}^k$

Constant-Round $\mathcal{ZK}$ Arguments for $\mathcal{NP}$

The Basic Idea



witness w **P** $x \in L$ **V**

$c = Com(0^k)$ $\longrightarrow$

$\longleftarrow$ r

$r \in_R \{0,1\}^{2n}$

$\mathrm{NTIME}(t(n))$ statement

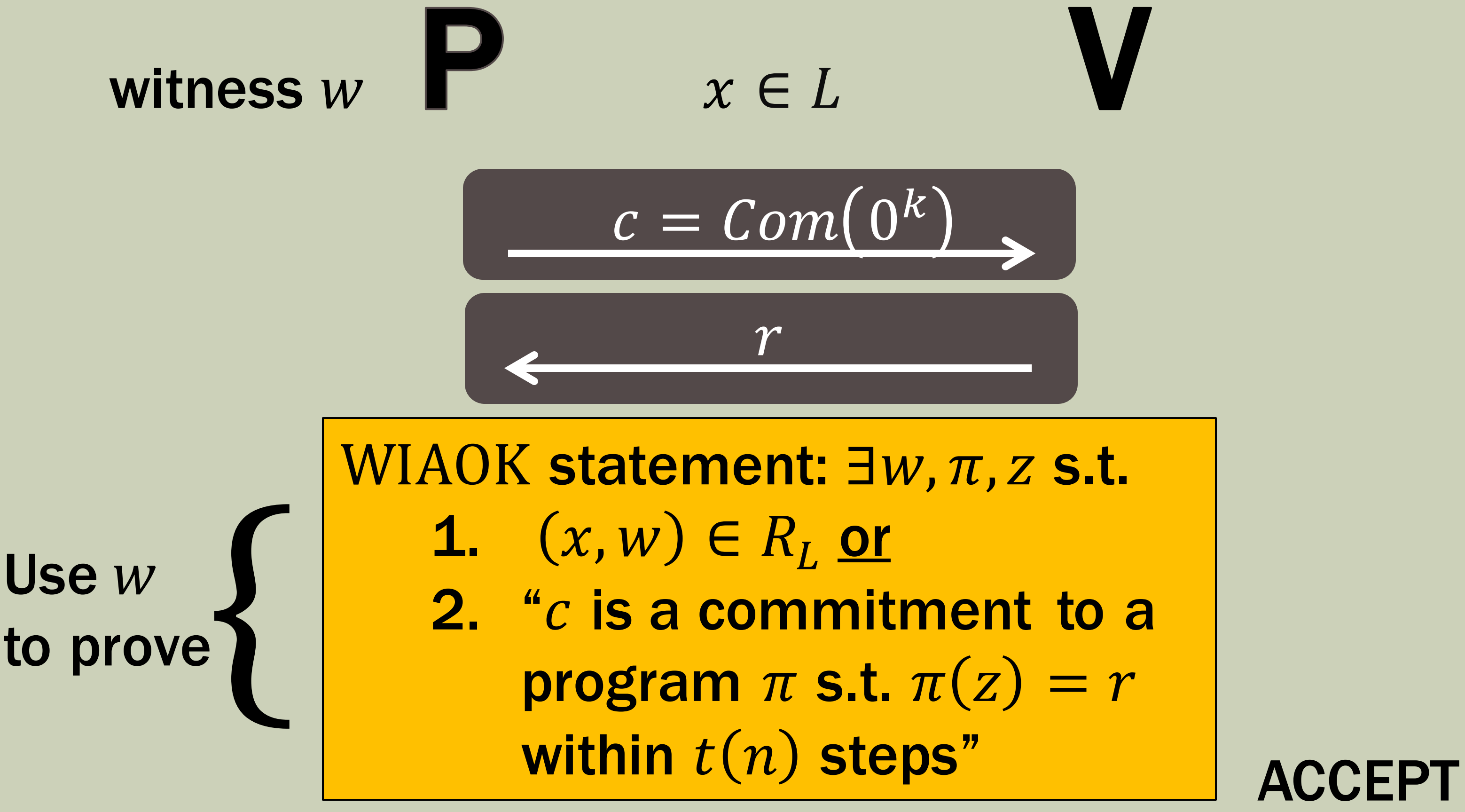
WIAOK **statement:** $\exists w, \pi, z$ **s.t.**

1. $(x, w) \in R_L$ **or**
2. **"c is a commitment to a program π s.t. $\pi(z) = r$ within $t(n)$ steps"**

Intuition:

- In the real interaction P cannot predict the random string r
- In simulation, $r = V^*(c)$ so S can set $\pi = V^*$ and $z = c$

Completeness



witness w **P**

$x \in L$

V

$c = Com(0^k)$ $\longrightarrow$

$\longleftarrow$ r

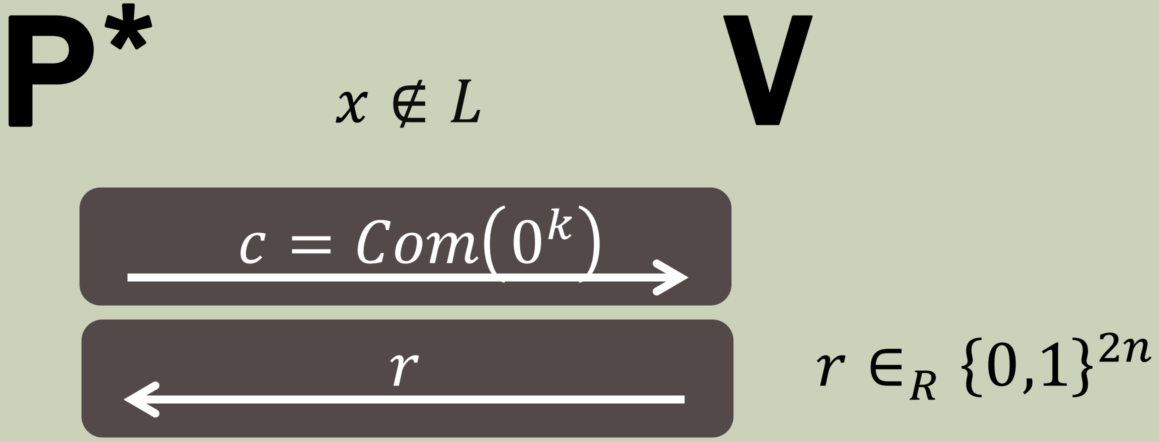
Use w to prove

WIAOK statement: $\exists w, \pi, z$ **s.t.**

1. $(x, w) \in R_L$ **<u>or</u>**
2. "c **is a commitment to a program** π **s.t.** $\pi(z) = r$ **within** $t(n)$ **steps**"

ACCEPT

Soundness



$x \notin L$

$c = Com(0^k)$

r

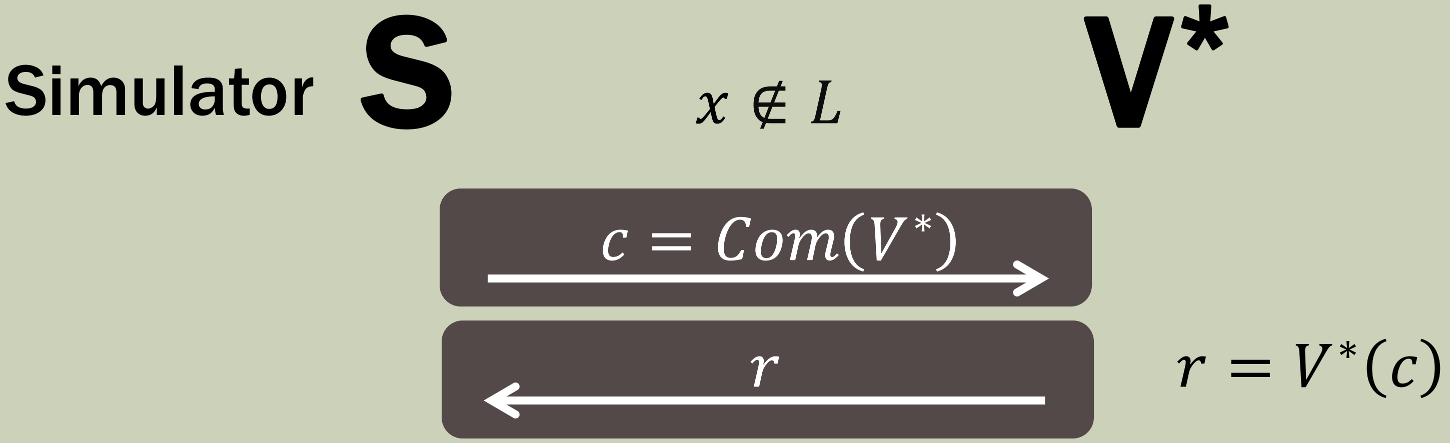
$r \in_R \{0,1\}^{2n}$

WIAOK statement: $\exists w, \pi, z$ s.t.

1. ~~$(x, w) \in R_L$~~ or
2. "c is a commitment to a program π s.t. $\pi(z) = r$ within $t(n)$ steps"

$$\forall \pi,\ Pr_r[\exists z \in \{0,1\}^n, \pi(z) = r] \le 2^n \cdot 2^{-2n}$$
$$= 2^{-n}$$

Zero-Knowledge



Use $\pi = V^*$ $z = c$ to prove

WIAOK statement: $\exists w, \pi, z$ s.t.

1. $(x, w) \in R_L$ **or**
2. "c is a commitment to a program π s.t. $\pi(z) = r$ within $t(n)$ steps"

Cannot distinguish if 1 or 2

By definition, $\pi(z) = V^*(c) = r$

Observations and Technical Issues

- **Simulator runs in strict polynomial time**
- **Possession of V^* is sufficient. No reverse engineering!**

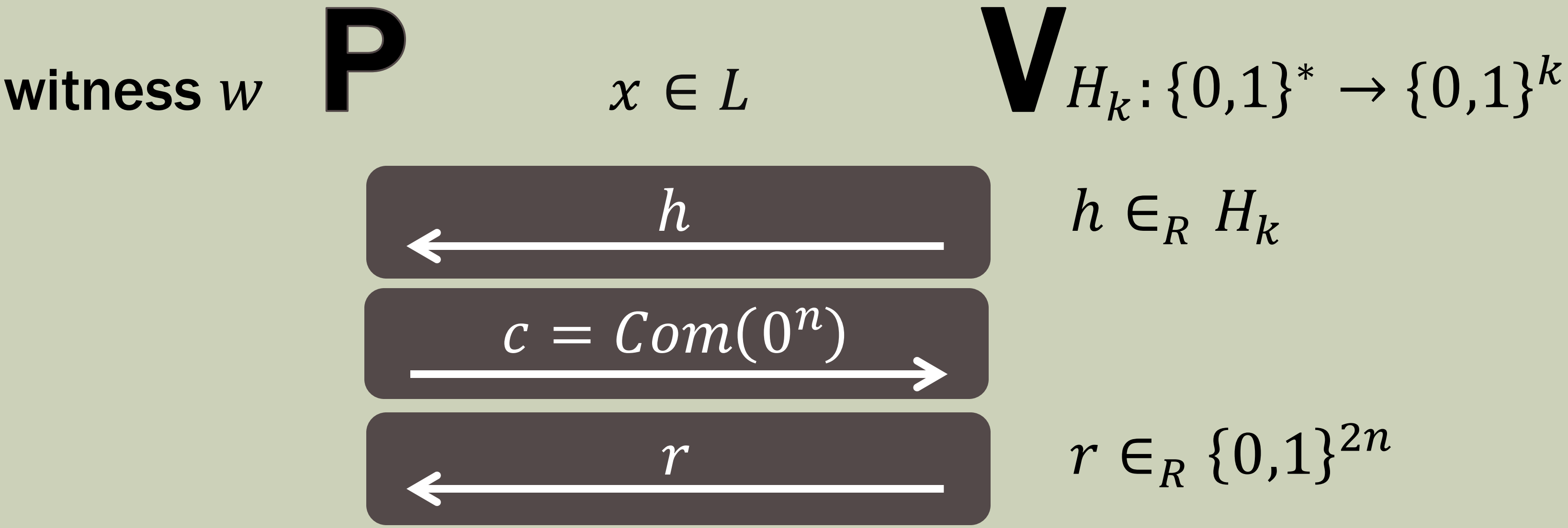
First technical issue:

- V^***'s size is** $poly(n)$**, but not a-priori bounded**
- **In particular, how can** $c = Com(V^*)$ **accommodate** V^***?**
- **Solution: use** $h: \{0,1\}^* \to \{0,1\}^k$ **to compute** $Com(h(V^*))$

Second technical issue:

- **Running time** $t(n)$ **of** V^* **not bounded by any fixed** $poly(n)$
- **So** $\mathrm{NTIME}(t(n))$ **relation in** WIAOK **is not an** NP**-relation**
- **Solution:** WIAOK **that handles** $\mathrm{NTIME}(n^{\omega(1)})$ **relations**

A constant-round ZK Argument



witness w **P** $x \in L$ **V** $H_k:\{0,1\}^* \rightarrow \{0,1\}^k$

h $h \in_R H_k$

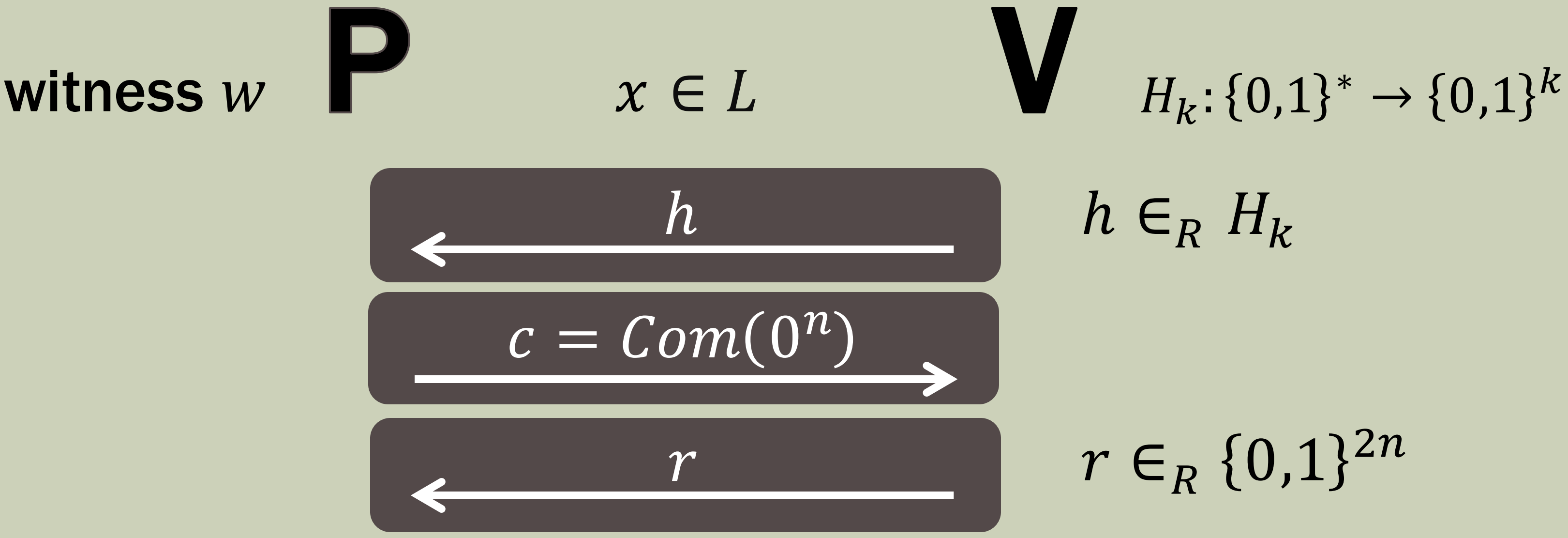
$c = Com(0^n)$

r $r \in_R \{0,1\}^{2n}$

WIAOK **statement:** $\exists w, \pi, z$ **s.t.**

1. $(x, w) \in R_L$ **or**
2. **"**c **is a commitment to** $h(\pi)$ **where** π **is a program s.t.** $\pi(z) = r$ **within** $t(n)$ **steps"**

The Relation R_{SIM}



witness w **P** $x \in L$ **V** $H_k: \{0,1\}^* \to \{0,1\}^k$

h $h \in_R H_k$

$c = Com(0^n)$

r $r \in_R \{0,1\}^{2n}$

$\mathrm{NTIME}(t(n))$
statement

WIAOK **statement:** $\exists w, \langle \pi, s, z \rangle$ **s.t.**

1. $(x, w) \in R_L$ **<u>or</u>**
2. $(\langle h, c, r \rangle, \langle \pi, s, z \rangle) \in R_{SIM}$

$(\langle h, c, r \rangle, \langle \pi, s, z \rangle) \in R_{SIM}$:

1. $|z| \leq |r| - n$
2. $c = Com(h(\pi), s)$ **<u>and</u>**
3. $\pi(z) = r$ **within** $t(n)$ **steps**

The Universal Language L_U

Goal: handling $\mathrm{NTIME}(t(n))$ statements for $t(n) = n^{\omega(1)}$

Consider the universal language L_U:

$$y = (M, x, t) \in L_U$$

$$\Updownarrow$$

$$\exists w, M(x, w) = \textbf{ACCEPT within } t \textbf{ steps}$$

- **Every $L \in \mathrm{NP}$ is linear-time reducible to L_U**
- **A proof system for L_U enables to handle all NP -statements**
- **More importantly, a proof system for L_U enables to handle $\mathrm{NTIME}(n^{\omega(1)})$ statements and even beyond (NEXP)**

Universal Arguments

Universal Argument Systems

$y = (M, x, t) \in L_U \iff \exists w, M(x, w) =$ **ACCEPT in t steps**

Definition [K'91, M'91, BG'02]: A universal argument system for L_U is a pair (P, V) such that $\forall y = (M, x, t)$:

Efficient verification: V runs in $poly(|y|)$ time

Completeness: If $y \in L_U$, then $Pr[(P, V)$ accepts $y] = 1$
Moreover, P runs in time $poly(t)$

Computational soundness: If $y \notin L_U$, then $\forall PPT\ P^*$
$Pr[(P^*, V)$ accepts $x] \leq neg(n)$

Theorem: If CRH exist, L_U has a universal argument

Building block: PCP Proof System

Makes use of a $\mathrm{PCP}[O(log), poly]$ system for L_U

What is a $\mathrm{PCP}[O(log), poly]$ proof system?

- **It is a $PPT\ V_{\mathrm{PCP}}$ with access to an oracle π_y that represents a proof for $y \in L_U$ in redundant form**
- **V_{PCP} (non-adaptively) queries q oracle bits of π_y where**

$$q = poly(|y|)$$

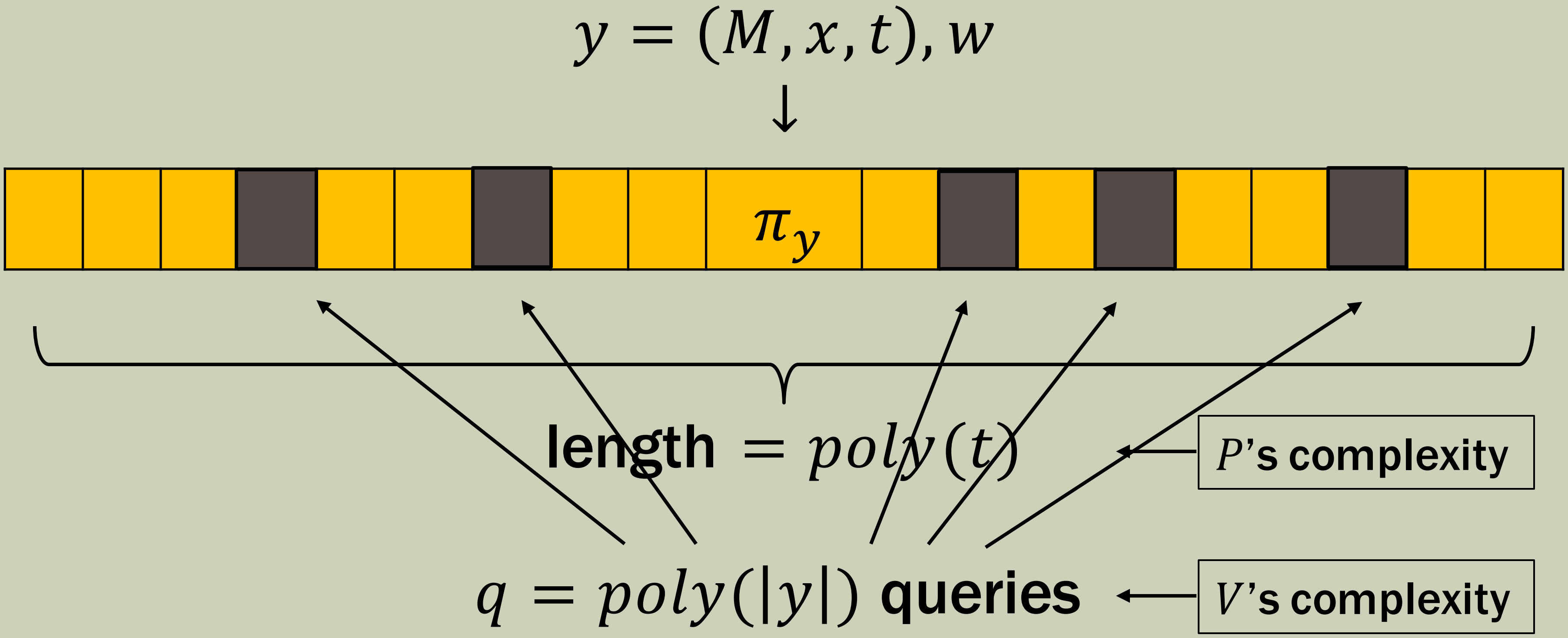
V's complexity

- **the bit positions are determined by V_{PCP}'s coin tosses**
- **the number of coins tossed by V_{PCP} is $\mathrm{O}(log\, t)$**
- **and the length of π_y is**

$$exp\big(\mathrm{O}(log\, t)\big) = poly(t)$$

P's complexity

PCP Reduction



$$y = (M, x, t), w$$

↓

π_y

length $= poly(t)$

P**'s complexity**

$q = poly(|y|)$ **queries**

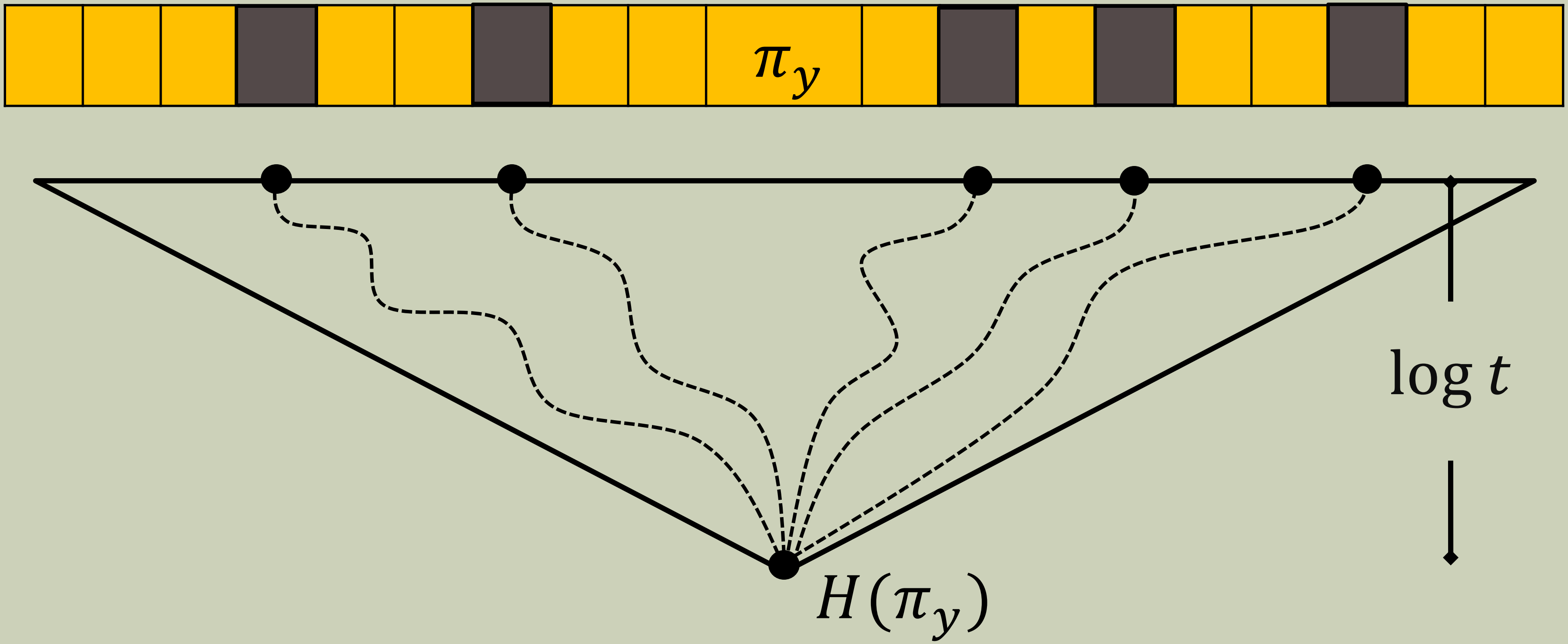
V**'s complexity**

the q **queries are determined by** $V_{\text{PCP}}(r)$ **where** $r \in \{0,1\}^{O(log\ t)}$

Commitment with Local Decommitment

Problem: the PCP is too long to be sent to V in its entirety

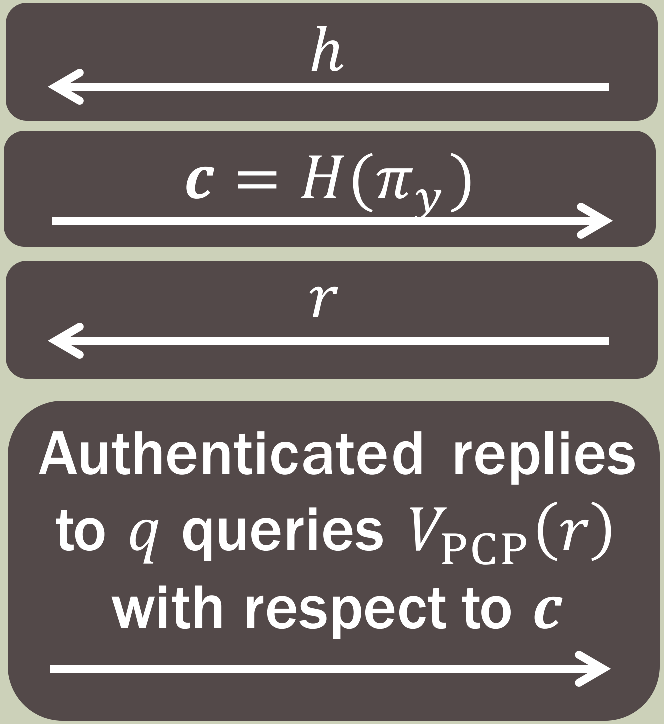
Solution: commit to π_y and allow "local decommitment"



H is computationally binding - built using CRH h

The Protocol

witness w **P** $y = (M, x, t) \in L_U$ **V** $H_k : \{0,1\}^* \to \{0,1\}^k$



$$\xleftarrow{h}$$

$h \in_R H_k$

time $poly(t)$

↓

π_y

$$\xrightarrow{\boldsymbol{c} = H(\pi_y)}$$

$$\xleftarrow{r}$$

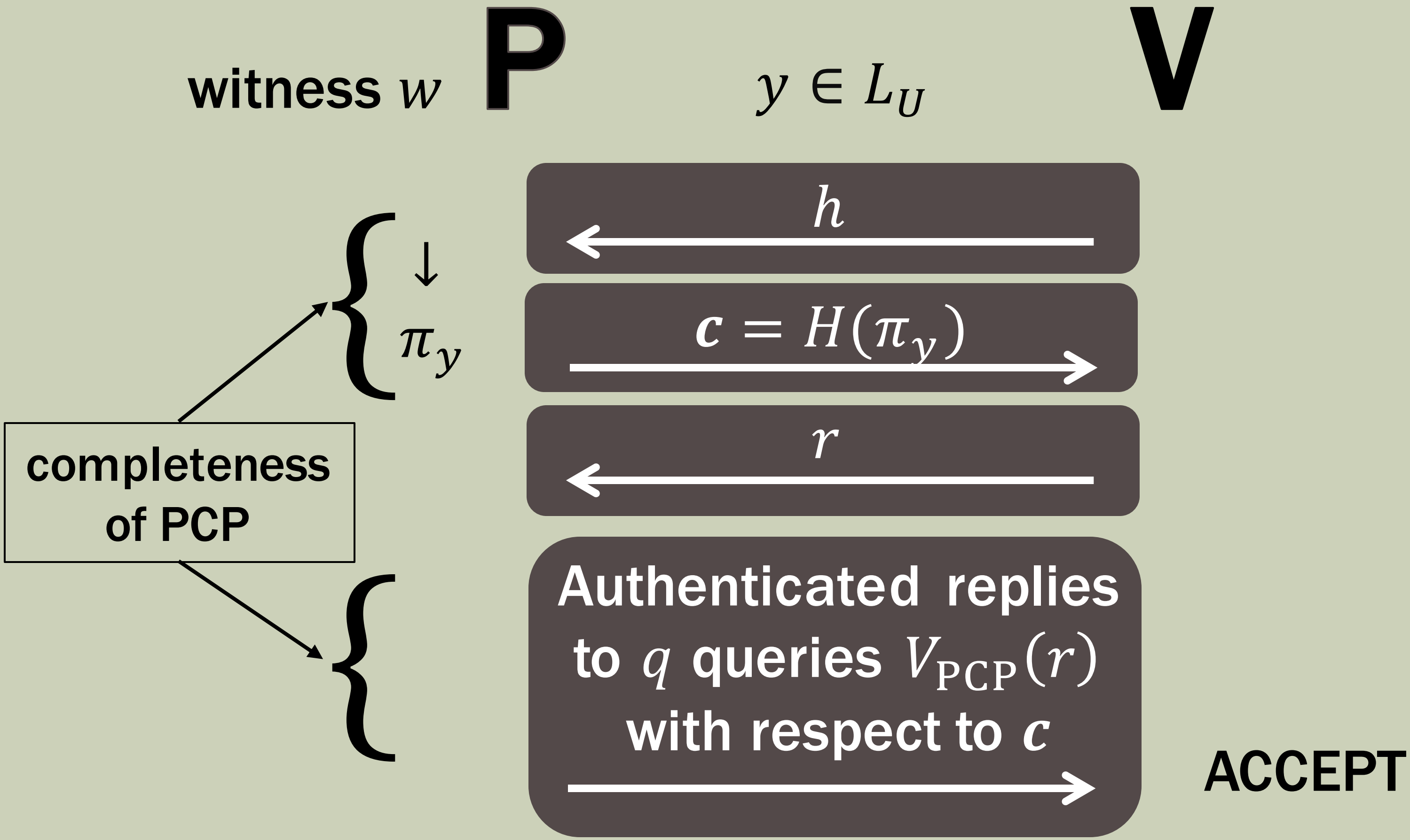
$r \in_R \{0,1\}^{O(log\ t)}$

Authenticated replies to q queries $V_{\text{PCP}}(r)$ with respect to $\boldsymbol{c}$

$$\longrightarrow$$

Time
$poly(q) = poly(|y|)$

Completeness



witness w **P**

$y \in L_U$

V

$\downarrow$

π_y

completeness of PCP

$\xleftarrow{h}$

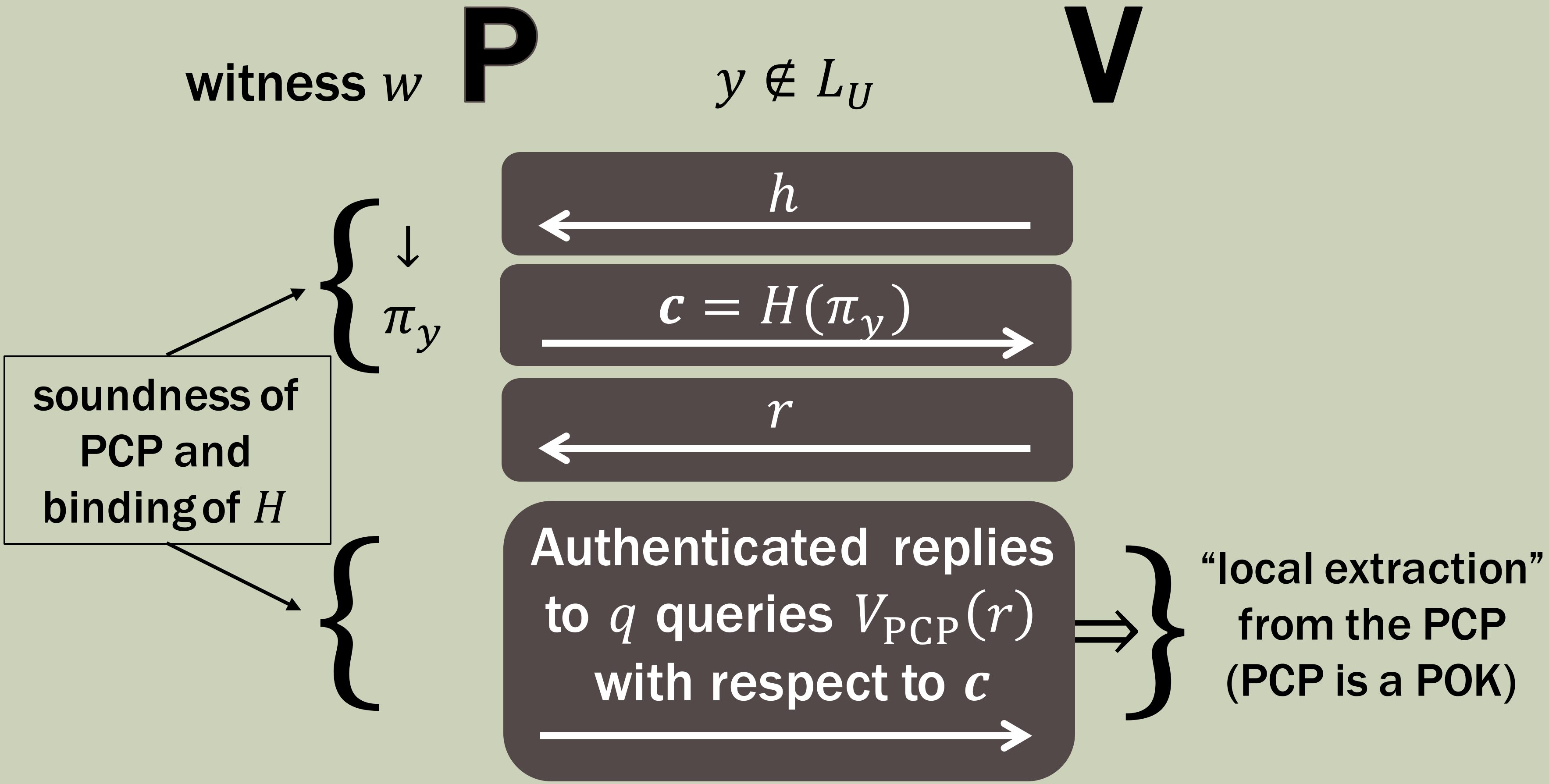
$\xrightarrow{\boldsymbol{c} = H(\pi_y)}$

$\xleftarrow{r}$

Authenticated replies to q queries $V_{\text{PCP}}(r)$ with respect to $\boldsymbol{c}$ $\longrightarrow$

ACCEPT

Computational Soundness



witness w **P**

$y \notin L_U$

V

h

$\left\{ \begin{array}{l} \downarrow \\ \pi_y \end{array} \right.$

$\boldsymbol{c} = H(\pi_y)$

r

soundness of PCP and binding of H

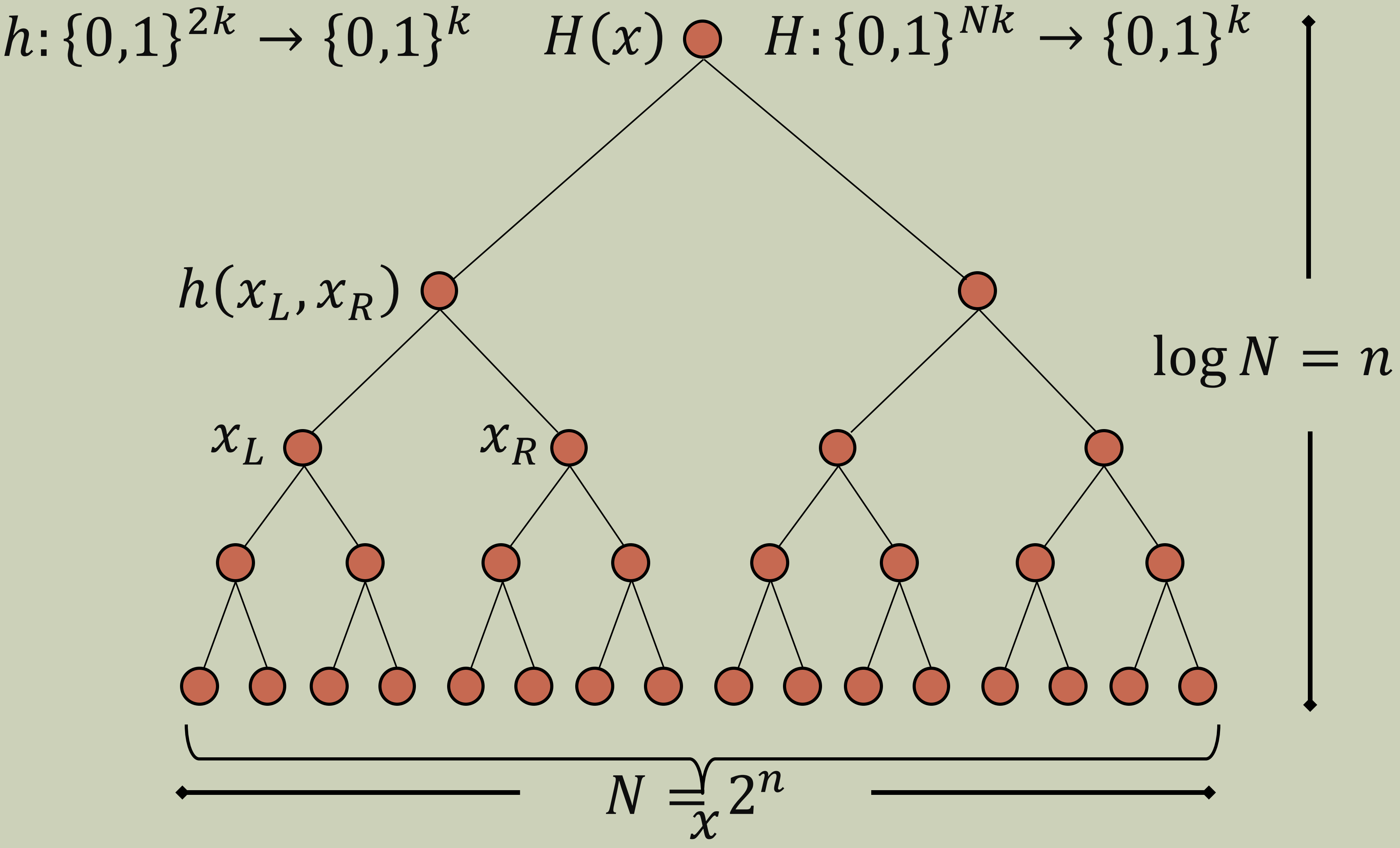
Authenticated replies to q queries $V_{\text{PCP}}(r)$ with respect to $\boldsymbol{c}$

$\Longrightarrow$ “local extraction” from the PCP (PCP is a POK)

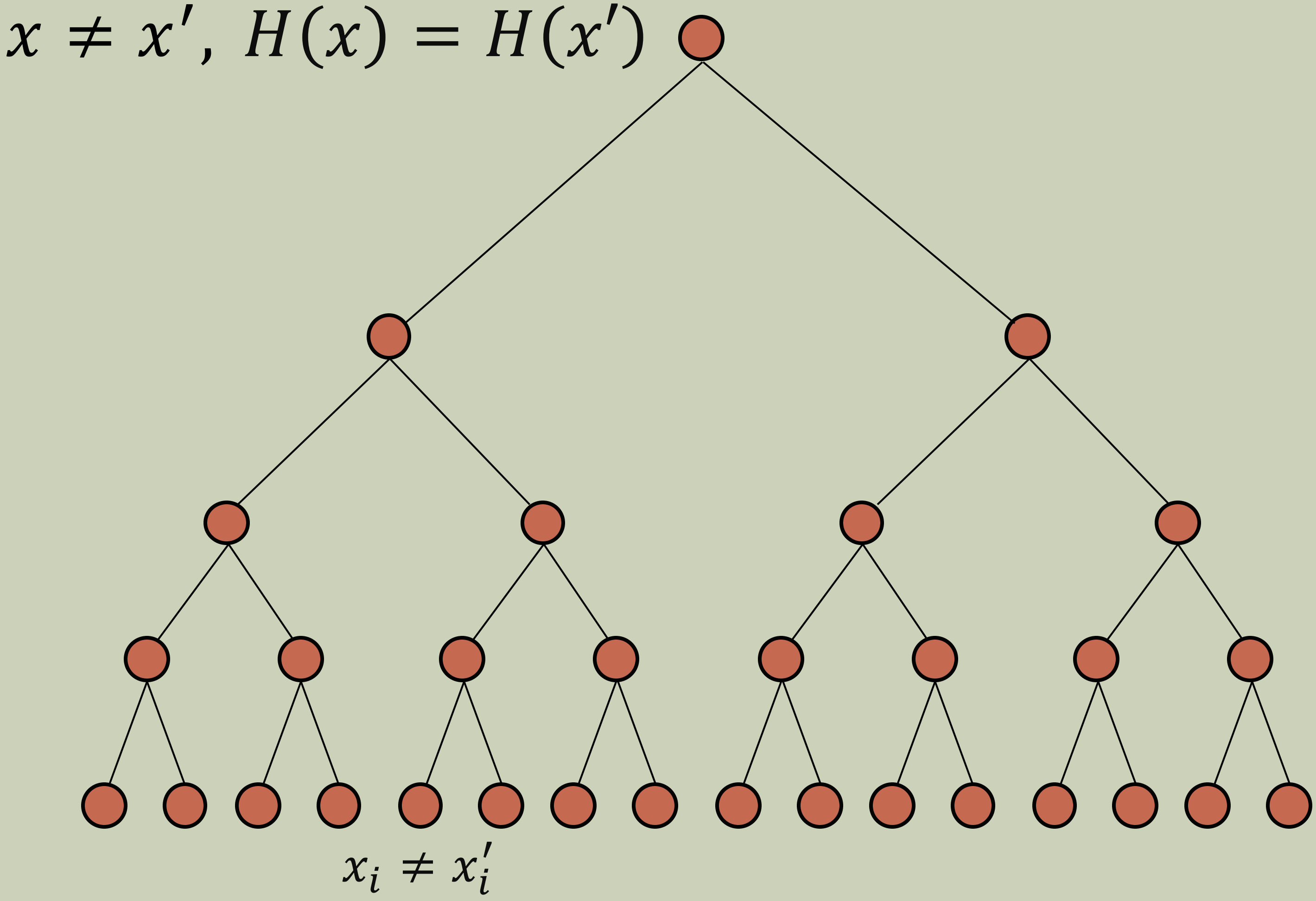
<u>Recall</u>: binding of H is computational - built using CRH h

Interlude:
Merkle Trees

Merkle Tree



Merkle Tree: Collision Resistance

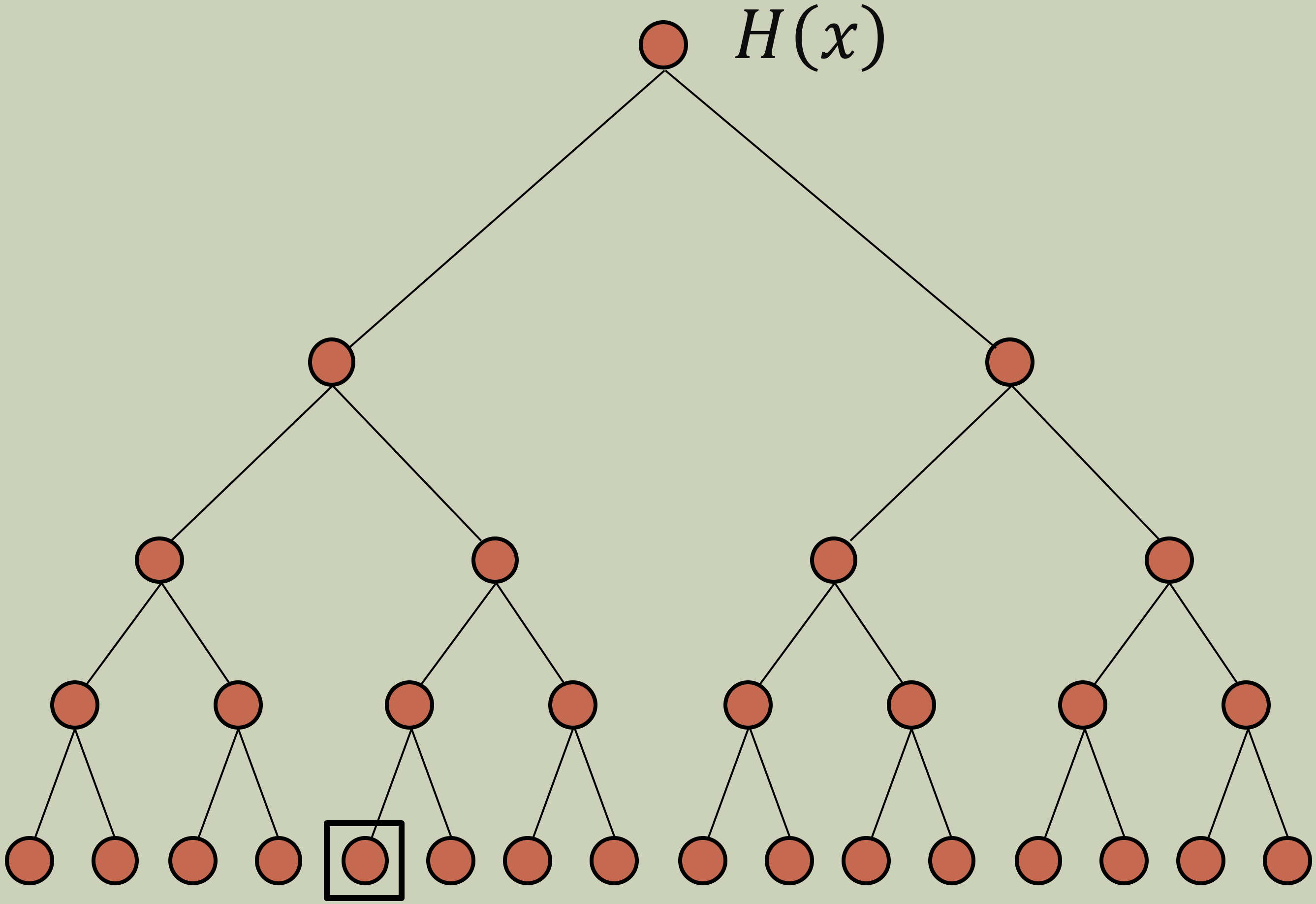


$x \neq x',\ H(x) = H(x')$

$x_i \neq x_i'$

Computationally (globally) binding

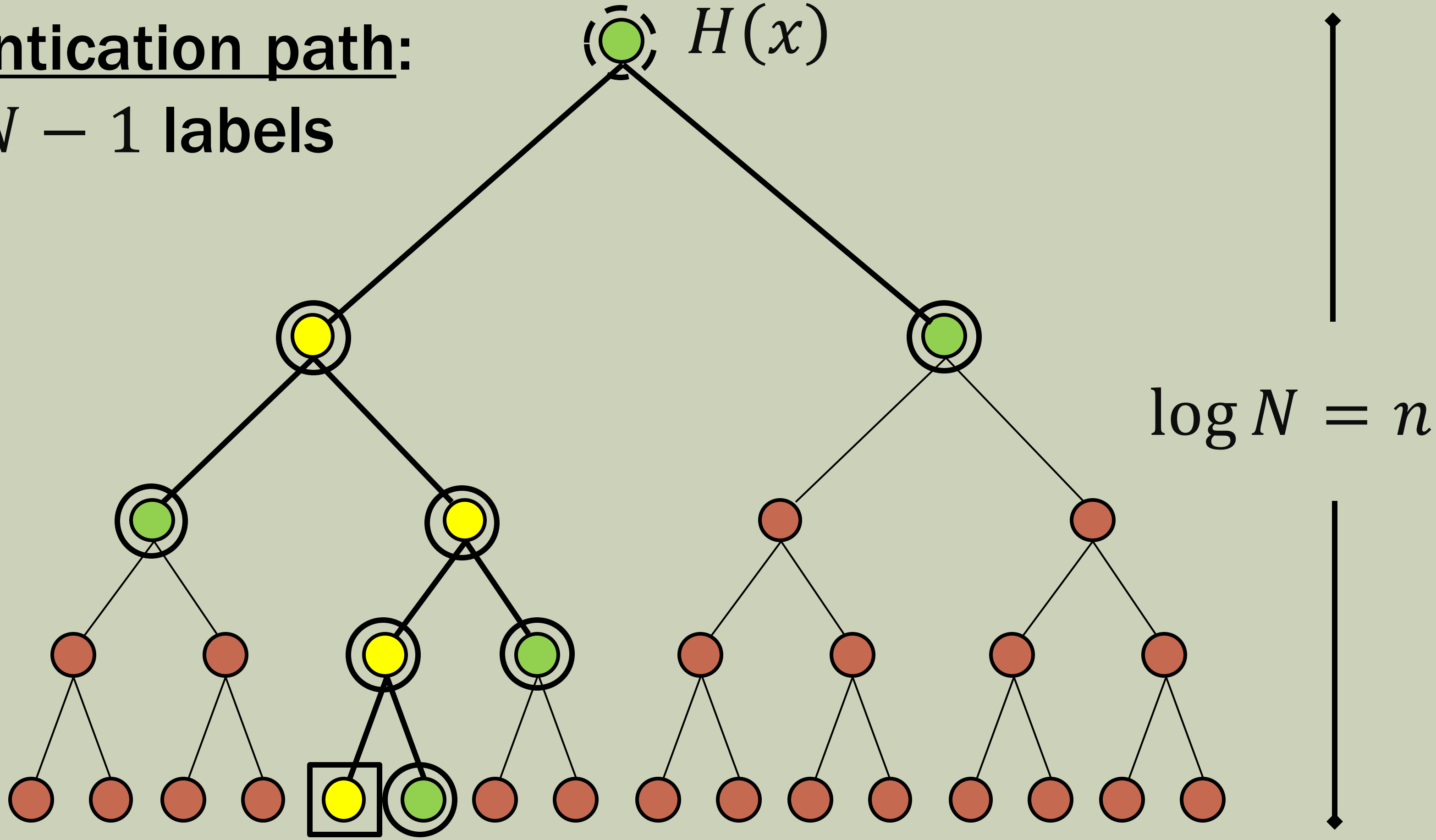
Merkle Tree: Local Decommitment



Merkle Tree: Local Decommitment

Authentication path:

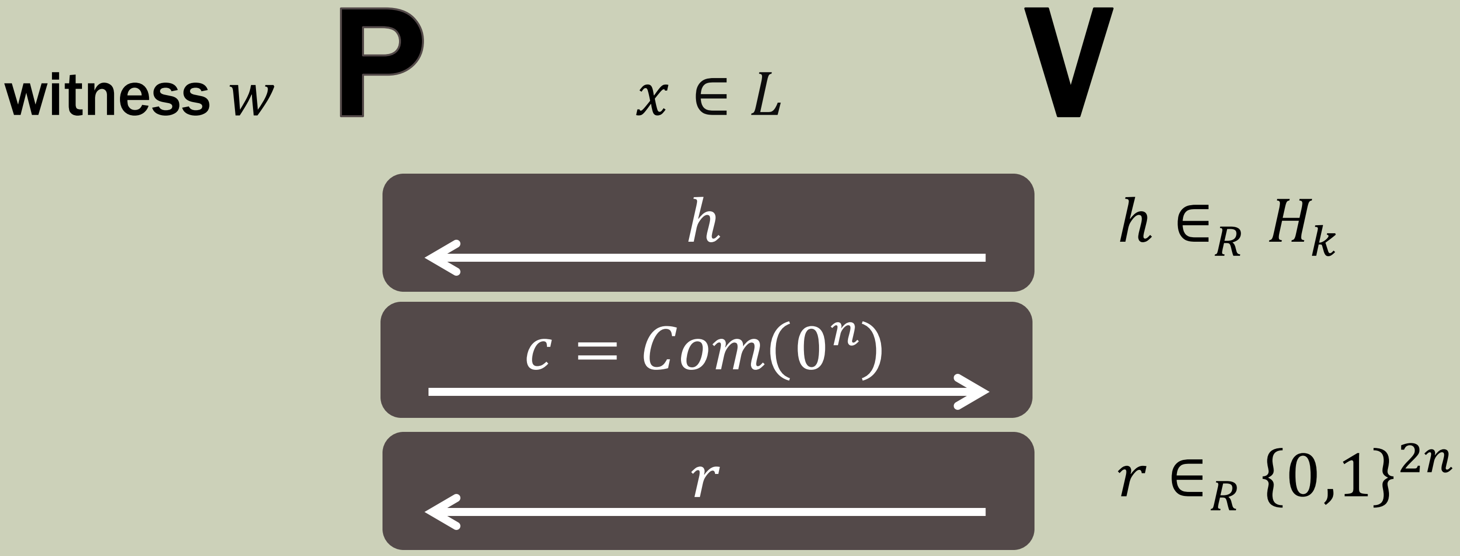
$2 \log N - 1$ **labels**



Computationally (locally) binding

Back to $\mathcal{ZK}$ Arguments for $\mathcal{NP}$

Recall: Barak's Protocol

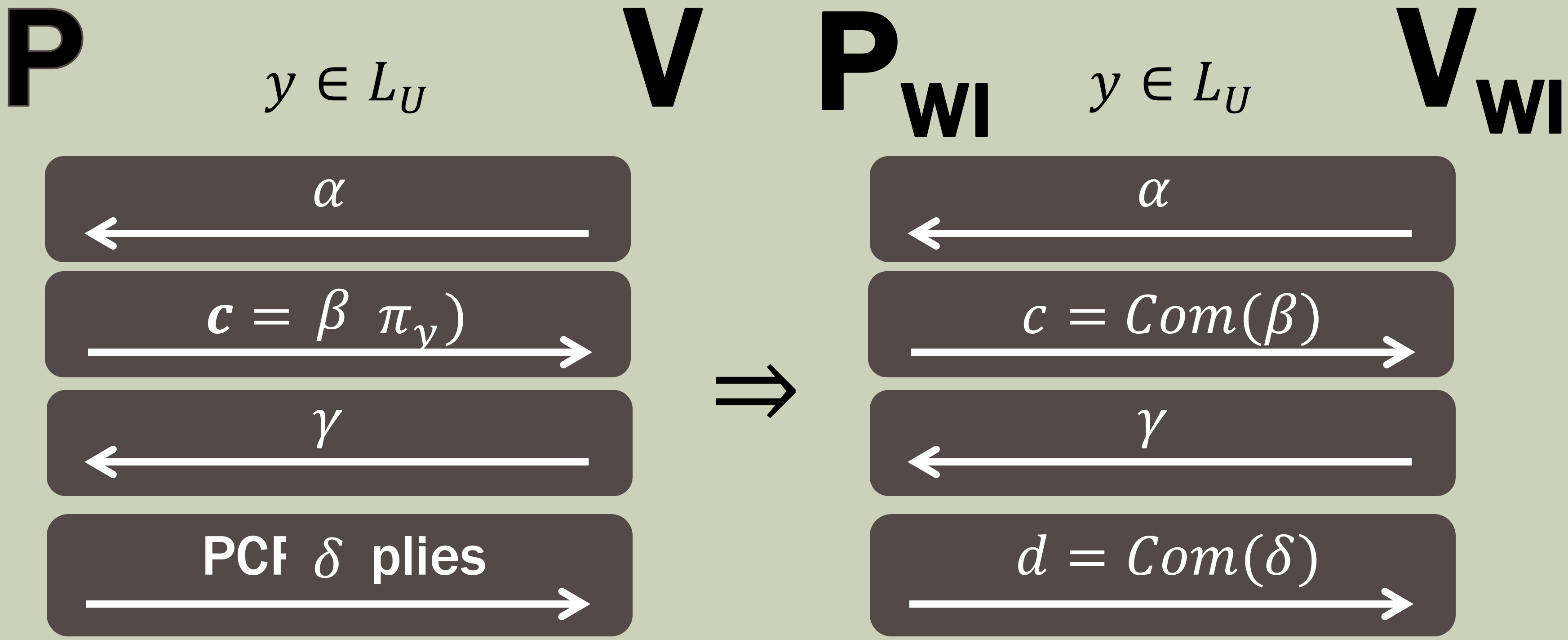


WIUAOK **statement:** $\exists w, \pi, z$ **s.t.**

1. $(x, w) \in R_L$ **<u>or</u>**
2. **"**c **is a commitment to** $h(\pi)$ **where** π **is a program s.t.** $\pi(z) = r$ **within** $t(n)$ **steps"**

<u>So far</u>: we only saw how to build UAOK**. What about** WI**?**

WI Universal Arguments



P $y \in L_U$ **V**

α

$\boldsymbol{c} = \beta \ \pi_y)$

γ

PCF δ **plies**

$\Rightarrow$

$\mathbf{P_{WI}}$ $y \in L_U$ $\mathbf{V_{WI}}$

α

$c = Com(\beta)$

γ

$d = Com(\delta)$

<u>**Subtle point**</u>**: actually run** k **parallel copies of** ZKPOK **with constant soundness error**

WIAOK **statement:** $\exists \beta, \delta$ **s.t.**

1. $c = Com(\beta)$
2. $d = Com(\delta)$
3. $V(\alpha, \beta, \gamma, \delta) =$ **ACCEPT**

Summary

Saw:

- CZK **argument** $\forall L \in \mathrm{NP}$
- **with negligible soundness**
- **a constant number of rounds**
- **and public-coin**

Tools:

- **Non-black-box simulation**
- WI **universal arguments**

Follow-up Work (2001-2012)

- Resettably-sound ZK [BGGL’01,CPS’13,COPVV’13]
- Constant-round bounded-conc. ZK and MPC [B’01,PR’03]
- Constant-round ZK with strict poly-time sim. [BL’02]
- Simultaneously resettable ZK and MPC [DGS’09,GM’11]
- Constant-round covert MPC [GJ’10]
- Constant-round public-coin parallel ZK [PRT’11]
- Simultaneously resettable WI-POK [COSV’12]
- Constant-round conc. ZK from iO [CLP’13, PPS’13, CLP’15]
- Concurrent secure computation [GGS’15]

New non-BB Techniques

[BP'12]:

- Impossibility for obfuscation → non BB simulation
- In particular, no use of PCP

[BKP'15]:

- Homomorphic trapdoors
- Enables to break all Black-Box barriers for e.g. WH

Food for Thought

Efficiency Optimizations

Efficiency of universal arguments depends on:

- **Number q of oracle queries made by V_{PCP} to π_y**

$$q = poly(|y|)$$

- **Length of π_y - depends on number of coins tossed by V_{PCP}**

$$exp\big(\mathrm{O}(log\, t)\big) = poly(t)$$

- **Optimizing params:**
  - **Larger alphabet size**
  - **Trading off prover/verifier time**
- **Less modular design and/or other models:**
  - **Interactive PCPs/oracle IPs**
  - **Using homomorphism of commitments**

Merkle Trees: Other Considerations

- **Can turn Merkle-tree into statistically hiding:**
  - **Generically**
  - **Assuming h is a random oracle**

<u>Open questions</u>:

- **Is $O(qk \log N)$ optimal?**
- **In practice N can be quite large**
- **Bulletproofs is $O(q + k \log N)$ but verifier space is N**
- **Lattices/amortization gets $O\left(q + k\sqrt{N}\right)$**

- **Ideally $O(q + k \log N)$ size <u>and</u> verification time**

Modern Crypto

- Define what it means to be secure
- Build a protocol/scheme
- Prove that protocol/scheme satisfies definition

- First *feasibility* then *efficiency*

- Relax definitions

History

Boaz Barak

Joe Kilian

Ralph Merkle

History


Rafael Pass


Nir Bitansky


Dakshita Khurana


Omer Paneth


Rachel Lin


Kai-Min Chung


Dustin Tseng


Muthuramakrishnan Venkitasubramaniam


Vipul Goyal


Abhishek Jain


Ivan Visconti

The End

Questions?