Crypto for PRAM from iO (via Succinct Garbled PRAM)

Kai-Min Chung

Academia Sinica, Taiwan

Joint work with:

Yu-Chi Chen, Sherman S.M. Chow, Russell W.F. Lai,

Wei-Kai Lin, Hong-Sheng Zhou

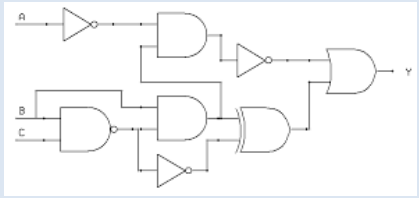
Computation in Cryptography

- Examples:
  - Multiparty Computation (MPC)
  - Non-interactive Zero Knowledge Proof (NIZK)
  - Fully Homomorphic Enc. (FHE)
  - Functional Encryption (FE)
  - Delegation with Persistent Database
  - Indistinguishability Obfuscation (iO)
- Traditionally, modeled as circuits
- Feasibility in more powerful computation model?

Models of Computation

- **Circuits**

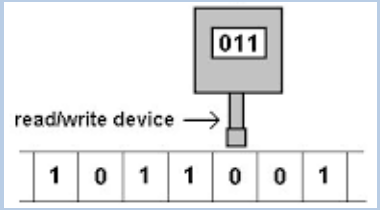
  Large description size
  Parallelizable

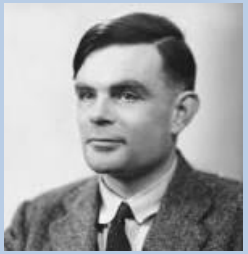


AND, OR, NOT gates

- **Turing Machines**

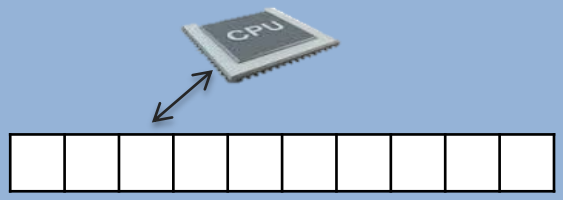
  Small description size





- **RAM Machines**

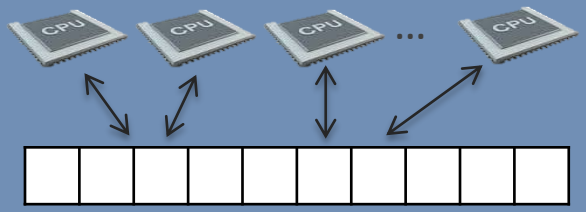
  Random data access



- **Parallel RAM**

  Random data access

  Parallelizable



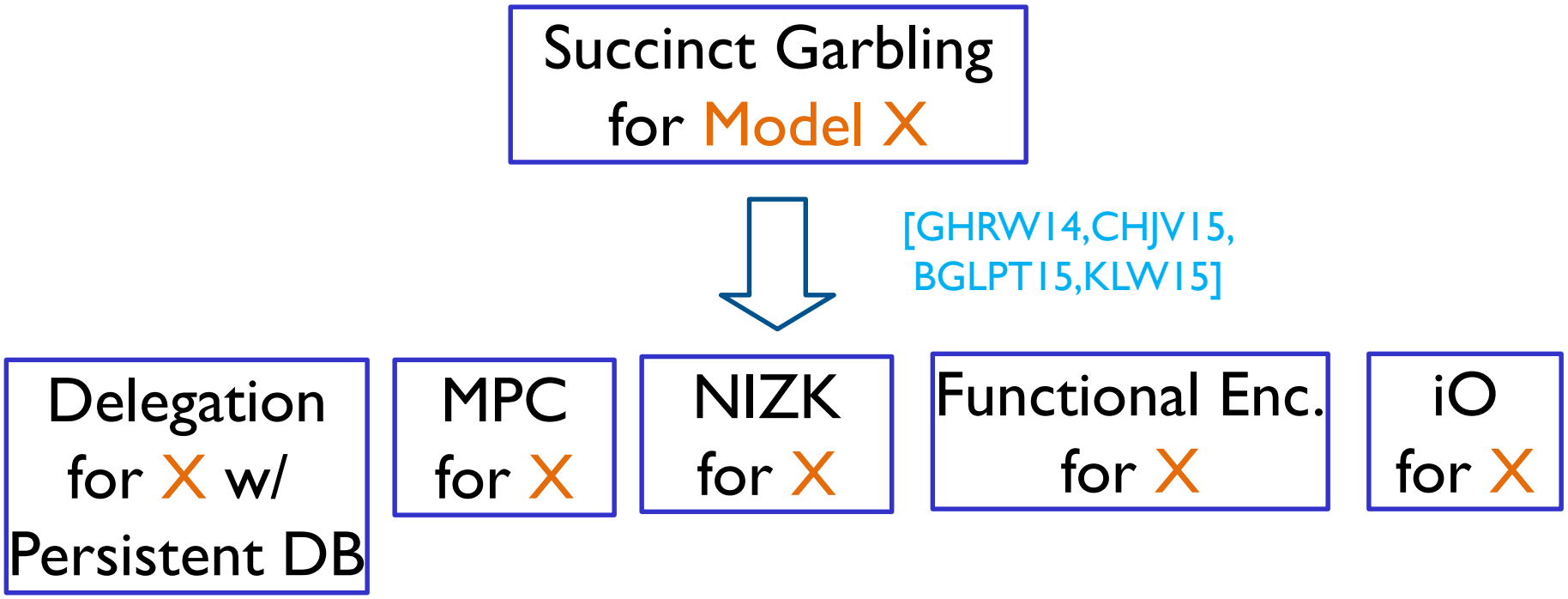
Efficiency Gap

Problem	Comp. Model	Total Time	Parallel Time
Binary search (input size n)	Circuit	Ω (n)	
	RAM	O(log n)	
Sorting	Circuit		O(log n)
	RAM		Ω (nlog n)
Keyword search/ Range query (output size m)	Circuit	Ω (n)	O(log n)
	RAM	O(mlog n)	Ω(mlog n)
	PRAM	O(mlog n)	O(log n)

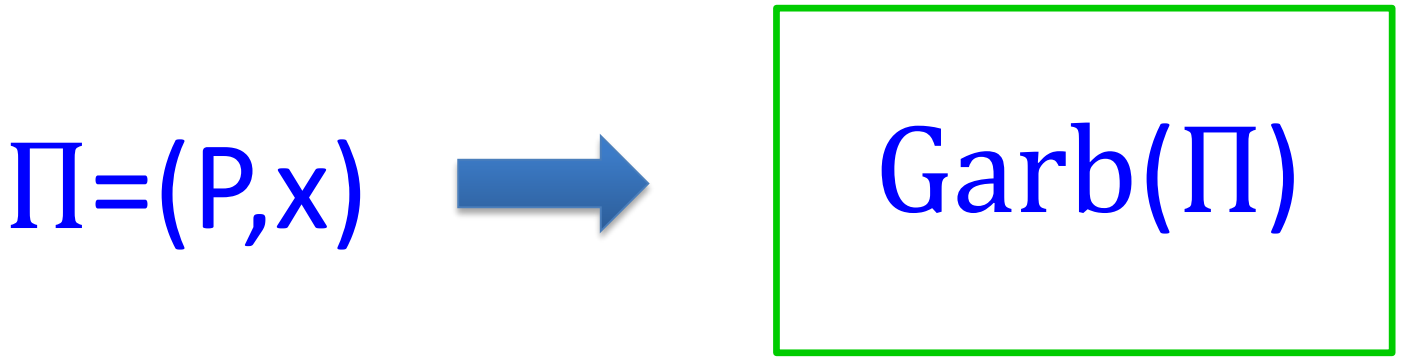
Parallel Model in Practice

- Emerging frameworks to handle big data
  - MapReduce, GraphLab, Spark, etc.
- Leverage *massive parallelism* & *random data access*
  - Circuit & RAM are not expressive enough
- PRAM: clean & expressive model to capture efficiency (total & parallel time & space) of these frameworks

Feasibility via Succinct Garbling

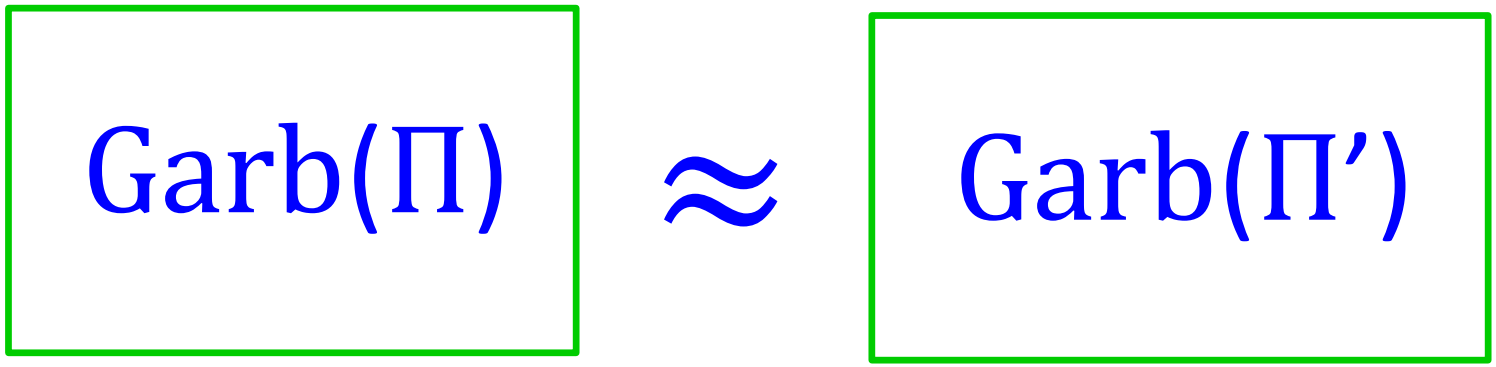


Succinct Garbling for Model X



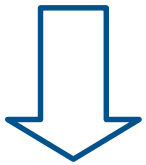
- Succinctness: Time(Garb(Π)) = poly(|Π|)
- Eval Efficiency: Complexity in Model X of

  Eval(Garb(Π)) ≈ Eval(Π) (up to polylog overhead)
- Security: Π, Π' same complexity & output ⟹



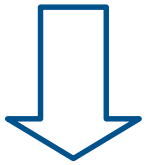
Feasibility via Succinct Garbling

iO for circuit
+ OWF



[KLW15]

Succinct Garbling
for X = TM



[GHRW14,CHJV15,
BGLPT15,KLW15]

Delegation for X w/ Persistent DB

MPC for X

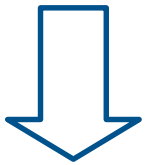
NIZK for X

Functional Enc. for X

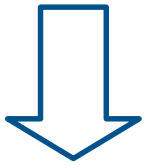
iO for X

Our Contribution

iO for circuit
+ OWF



Succinct Garbling
for X = PRAM



[GHRW14,CHJV15,
BGLPT15,KLW15]

Delegation
for X w/
Persistent DB

MPC
for X

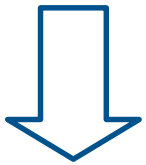
NIZK
for X

Functional Enc.
for X

iO
for X

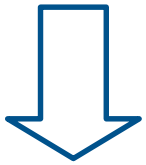
Concurrent Work [CH16]

iO for circuit
+ OWF



Modular Proof

Succinct Garbling
for X = RAM



[GHRW14,CHJV15,
BGLPT15,KLW15]

Delegation
for X w/
Persistent DB

MPC
for X

NIZK
for X

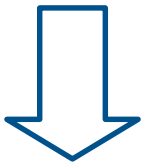
Functional Enc.
for X

iO
for X

Succinct Garbling for TM [KLW15]

Abstraction of [KLW15]

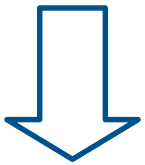
iO for circuit
+ OWF



Authentication Step

ST-Garbling for TM

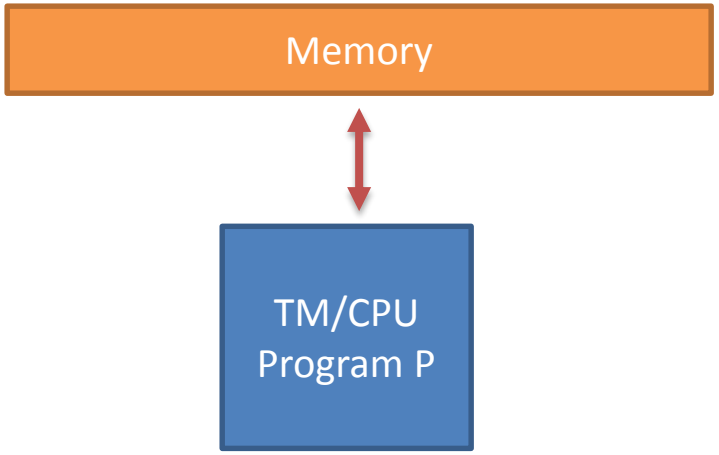
Same-Trace Garbling



Hiding Step

Succinct Garbling for TM

Same-Trace Garbling for TM/RAM



Computation Trace =
(initial-value),
$(st_1, addr_1, val_1)$,
$(st_2, addr_2, val_2)$,
$(st_3, addr_3, val_3)$,
…
$(st_{T-1}, addr_{T-1}, val_{T-1})$,
$(st_T, addr_T, val_T)$

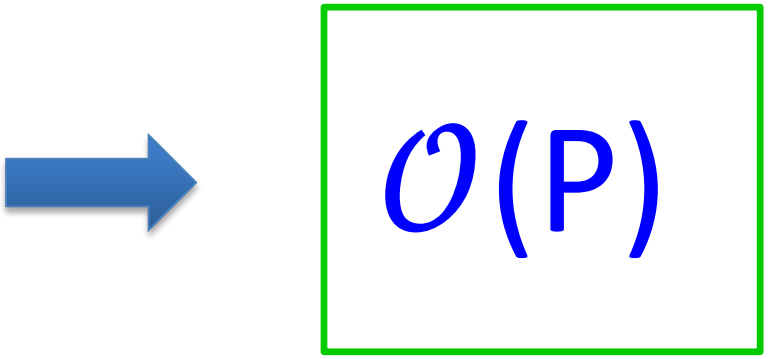
- Security: Π, Π' *same trace* (so same inp/out, complexity) $\Longrightarrow$

$$\text{Garb}(\Pi) \approx \text{Garb}(\Pi')$$

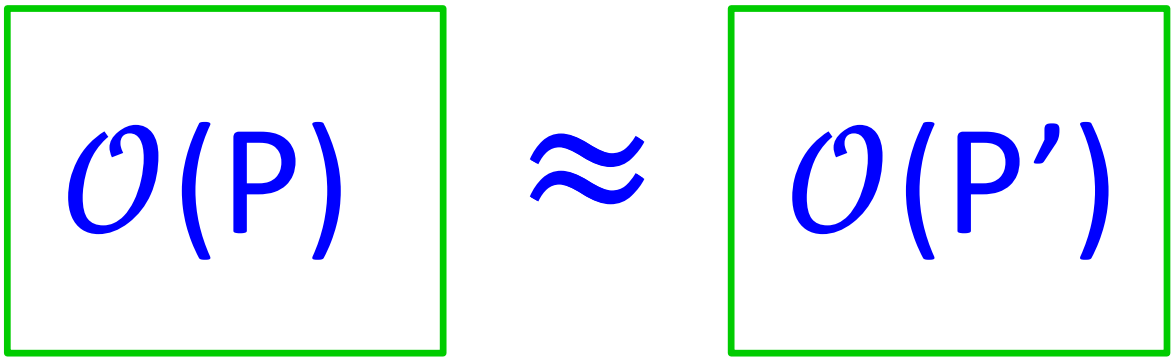
Indistinguishability Obfuscation (iO)

[BGI+12,GGH+13]

- Scramble program to make it "unintelligible"

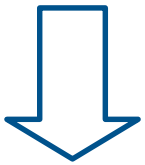


- Maintain functionality: $\mathcal{O}(P)(x) = P(x) \quad \forall\, x$
- Security: If $P(x) = P'(x) \quad \forall\, x$ & same size $\Longrightarrow$



Abstraction of [KLW15]

iO for circuit
+ OWF

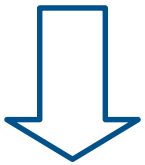


Authentication Step

$ST\text{-}Garb(P, x) = (iO(P_{auth}), x_{auth})$

Only generate comp. trace of P(x)

ST-Garbling for TM



Hiding Step

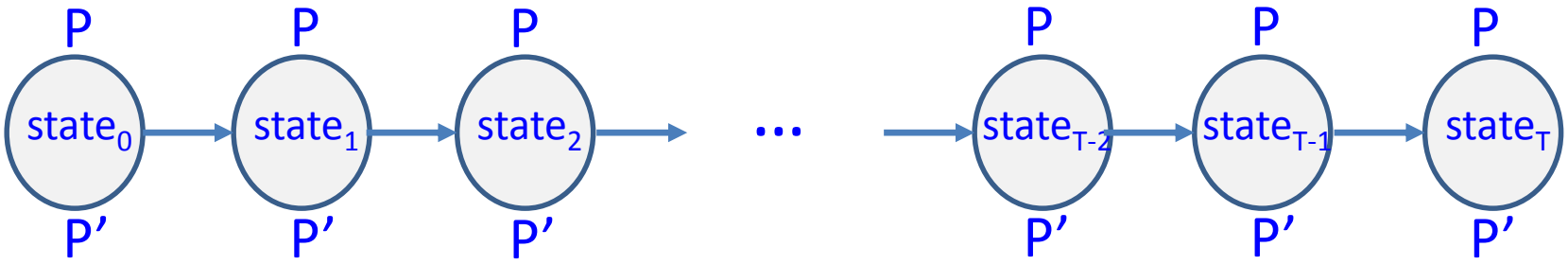
$Garb(P, x) = (ST\text{-}Garb(P_{hide}, x_{hide}))$

Hide memory/CPU state content & memory access pattern

Succinct Garbling for TM

Authentication & Hiding in [KLW15]

- Authentication step: ST-Garb(P, x) = (iO(P_{auth}), x_{auth})
  - iO-friendly authentication primitives
  - Enable program switching step by step in hybrids



Authentication & Hiding in [KLW15]

- Authentication step: ST-Garb(P, x) = (iO(P_{auth}), x_{auth})
  - iO-friendly authentication primitives
  - Enable program switching step by step in hybrids
- Hiding step: Garb(P, x) = (ST-Garb(P_{hide}, x_{hide}))
  - Hide content by encryption
  - Hide access pattern by Oblivious TM [PF79]
  - Allow erasing computation step by step in hybrids

Succinct Garbling for RAM

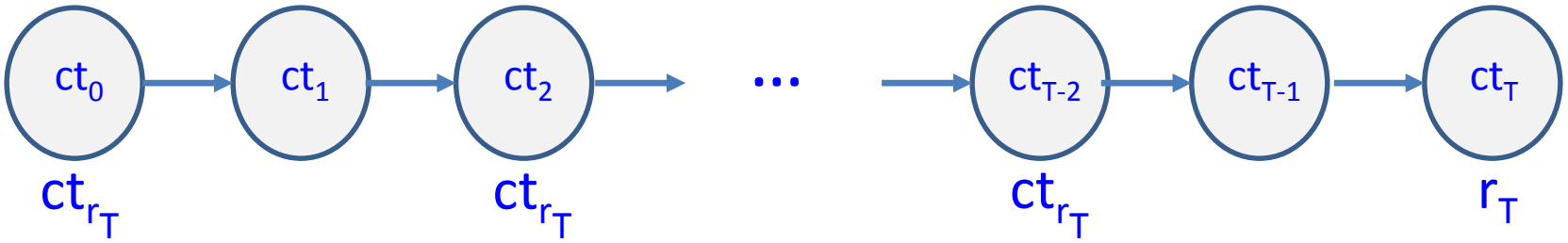
Challenge: Hiding Access Pattern

$\text{Garb}(P, x) = (\text{ST-Garb}(P_{\text{hide}}, x_{\text{hide}}))$

- Replace Oblivious TM by Oblivious RAM [GO96]
- Issue: Cannot use ORAM security
  - ORAM is inherently randomized, security hold only when ORAM randomness is hidden
- Idea: “Puncturing” ORAM

Puncturing ORAM

- Use tree-based ORAM [SLSC11], which is "puncturable"
  - t-th step access pattern is determined by single randomness r_t
  - if r_t is punctured/erased from program, t-th step access pattern can be simulated by random
- Puncturing r_t
  - r_t may appear multiple times (encrypted) in history
  - Carefully erase r_t backward in time step by step
    - Modify program: "erase r_t after step s" for $s = t, t-1, \ldots, 0$



Puncturing ORAM

- Use tree-based ORAM [SLSC11], which is "puncturable"
  - t-th step access pattern is determined by single randomness r_t
  - if r_t is punctured/erased from program, t-th step access pattern can be simulated by random
- Puncturing r_t
  - r_t may appear multiple times (encrypted) in history
  - Carefully erase r_t backward in time step by step
    - Modify program: "erase r_t after step s"

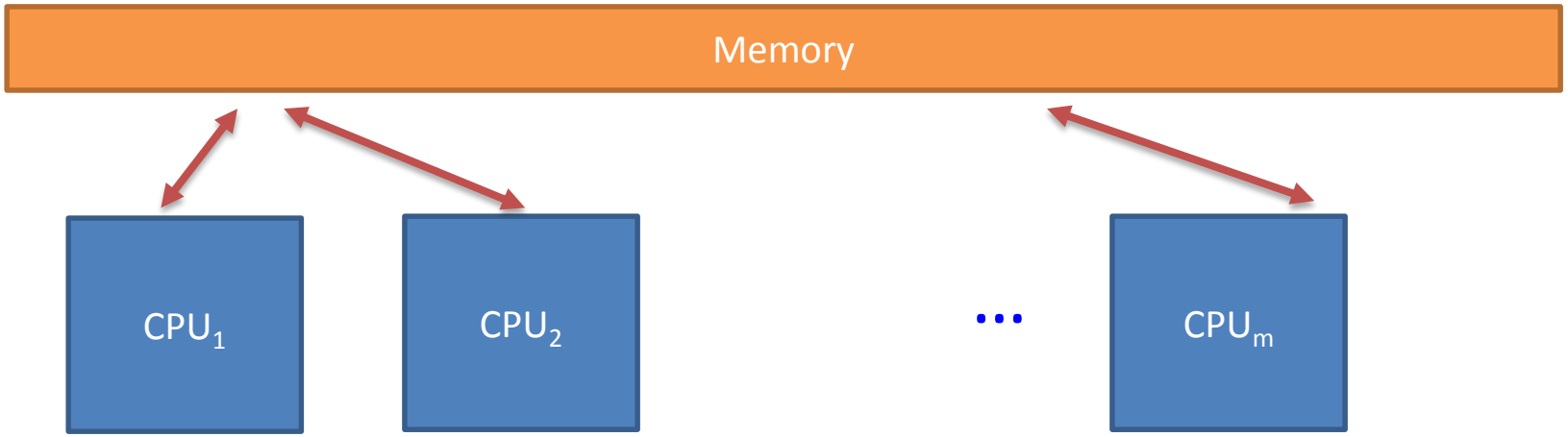
[CH16]: "2 tracks trick" w/ modular & simpler proof

Succinct Garbling for PRAM

Challenge: Authenticate Memory

$\text{ST-Garb}(P, x) = (\text{iO}(P_{\text{auth}}), x_{\text{auth}})$

- Memory authenticated by "Merkle tree"
  - root stored in CPU state
  - Locally updatable by given augment path
- Issue: Parallel CPU $\Longrightarrow$ Parallel Update
  - Require CPU-to-CPU communication

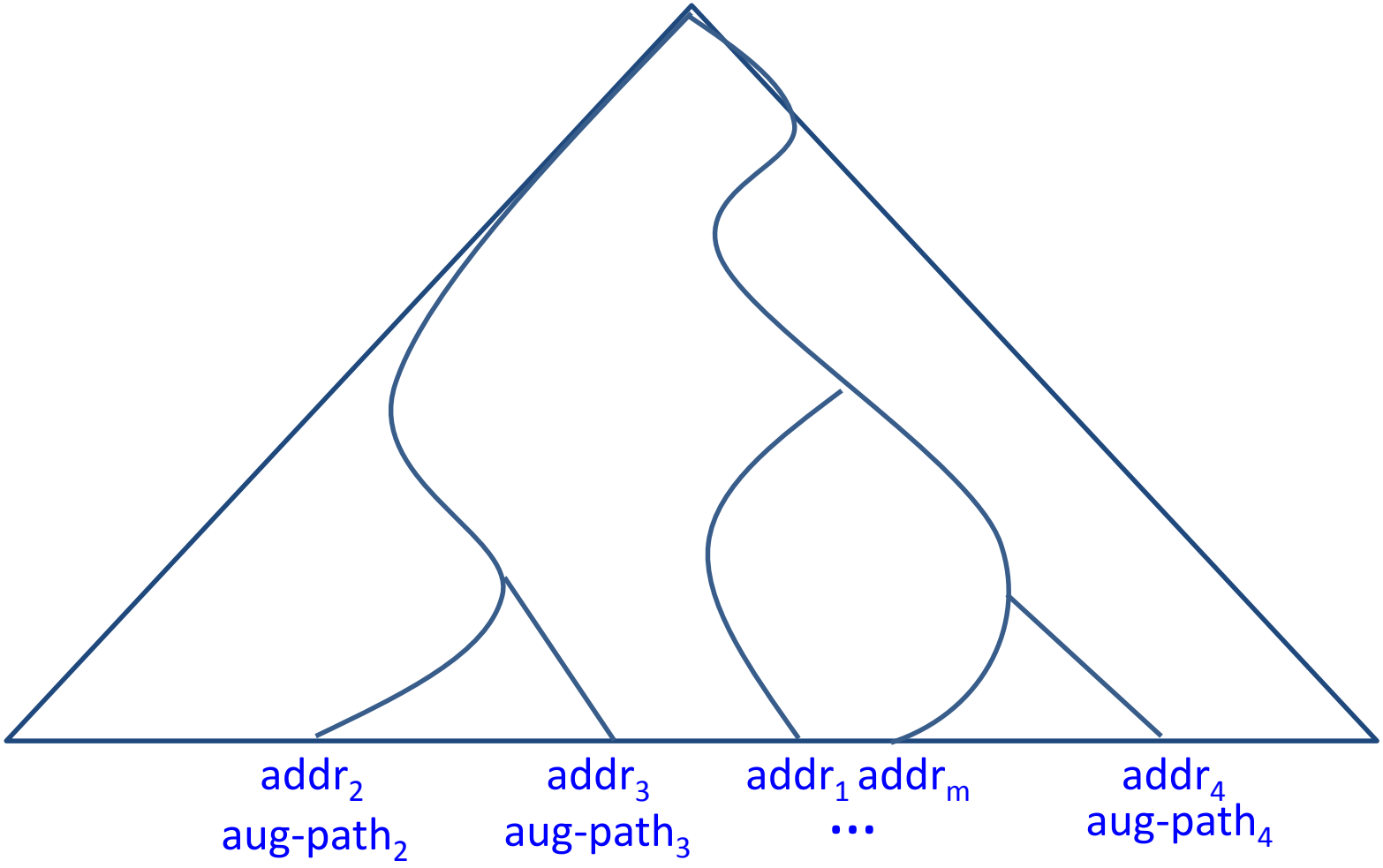


Challenge: Authenticate Memory

ST-Garb(P, x) = (iO(P_{auth}), x_{auth})

- Memory authenticated by "Merkle tree"
  - root stored in CPU state
  - Locally updatable by given augment path
- Issue: Parallel CPU $\Longrightarrow$ Parallel Update
  - Require CPU-to-CPU communication
- Issue: Cannot afford $\Omega(m)$ overhead in parallel time
  - Otherwise, void the gain of parallelism

Parallel Update Problem



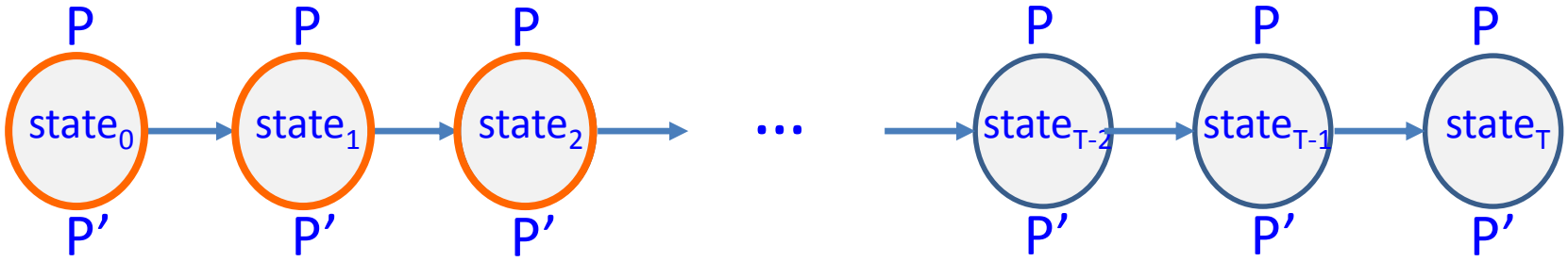
Challenge: Authenticate Memory

ST-Garb(P, x) = (iO(P_{auth}), x_{auth})

- Memory authenticated by "Merkle tree"
  - root stored in CPU state
  - Locally updatable by given augment path
- Issue: Parallel CPU $\Longrightarrow$ Parallel Update
  - Require CPU-to-CPU communication
- Issue: cannot afford $\Omega(m)$ overhead in parallel time
  - Otherwise, void the gain of parallelism
- $O(\log^2 m)$ -round parallel algorithm
  - Parallel update level-by-level from leaves to root

Security Issue: High Pebble Complexity

Put "pebble" on node to switch program



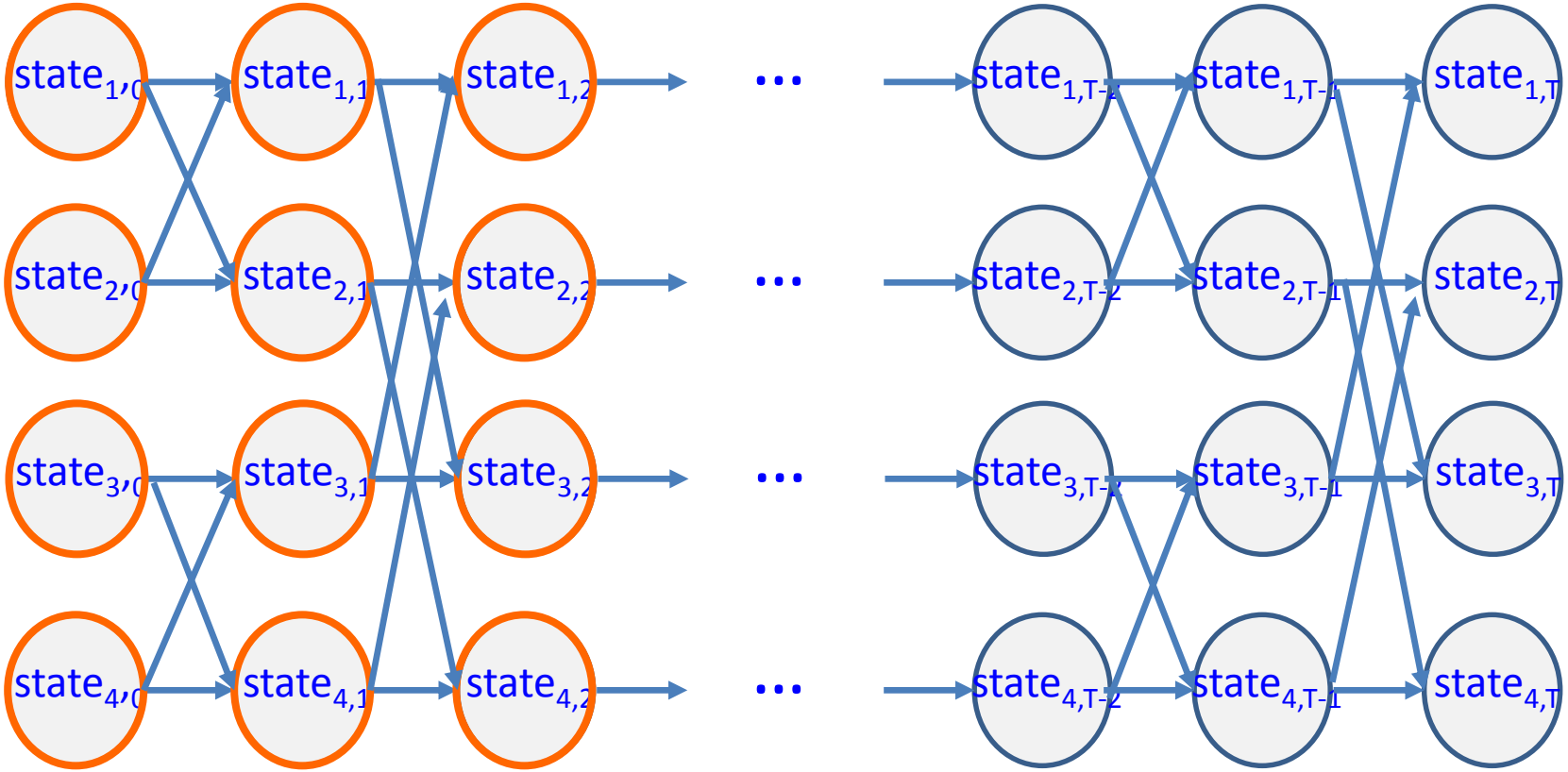
Put pebble on node require to hardwire input/output

Security Issue: High Pebble Complexity

Can use 2m pebbles to traverse graph, but not better

⇒ Need to hardwire $\Omega(m)$ information in P_{auth}

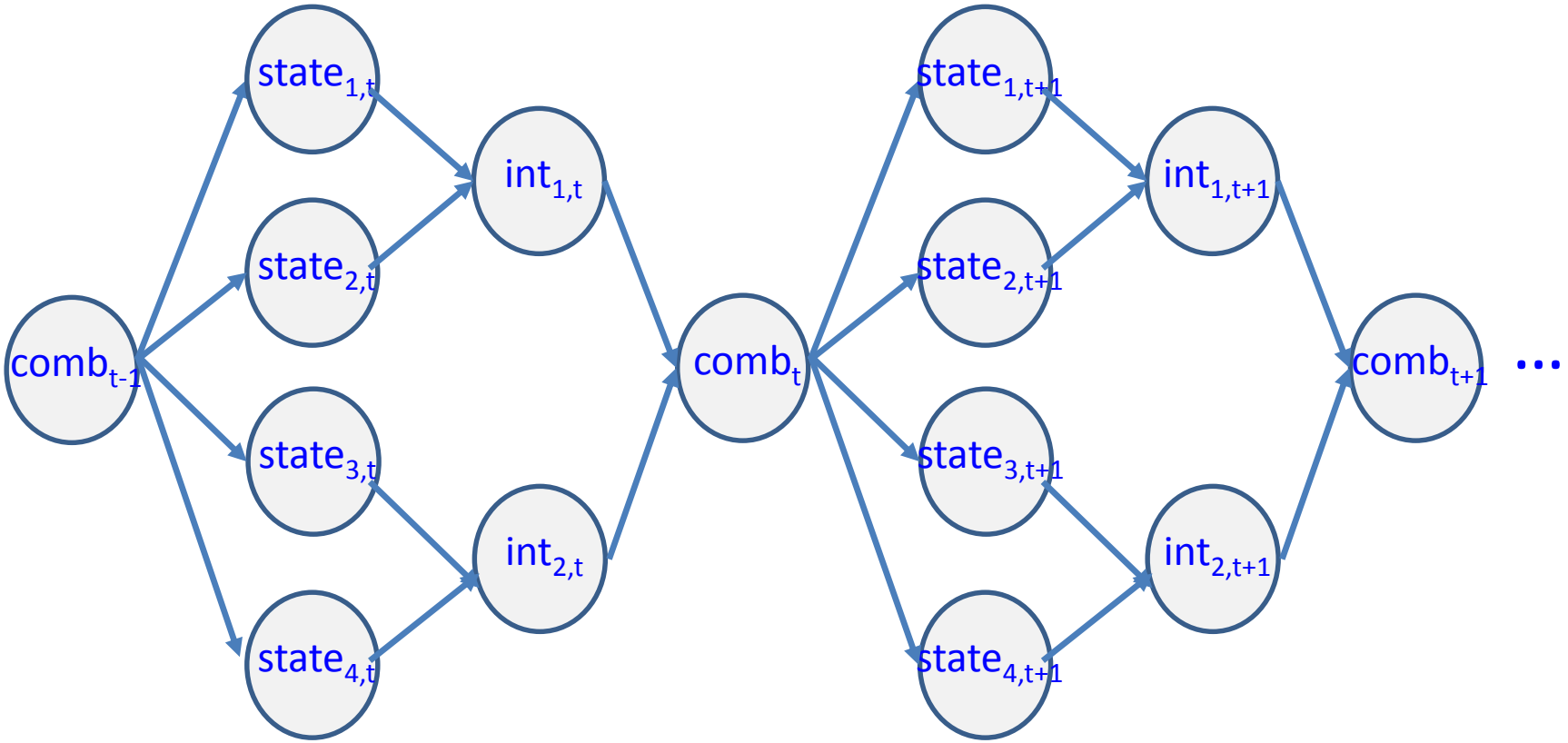
⇒ poly(m) overhead



Branch & Combine Emulation

Change topology to reduce pebble complexity

- Combine m CPU states to 1 combined state
- Branch one step computation from it

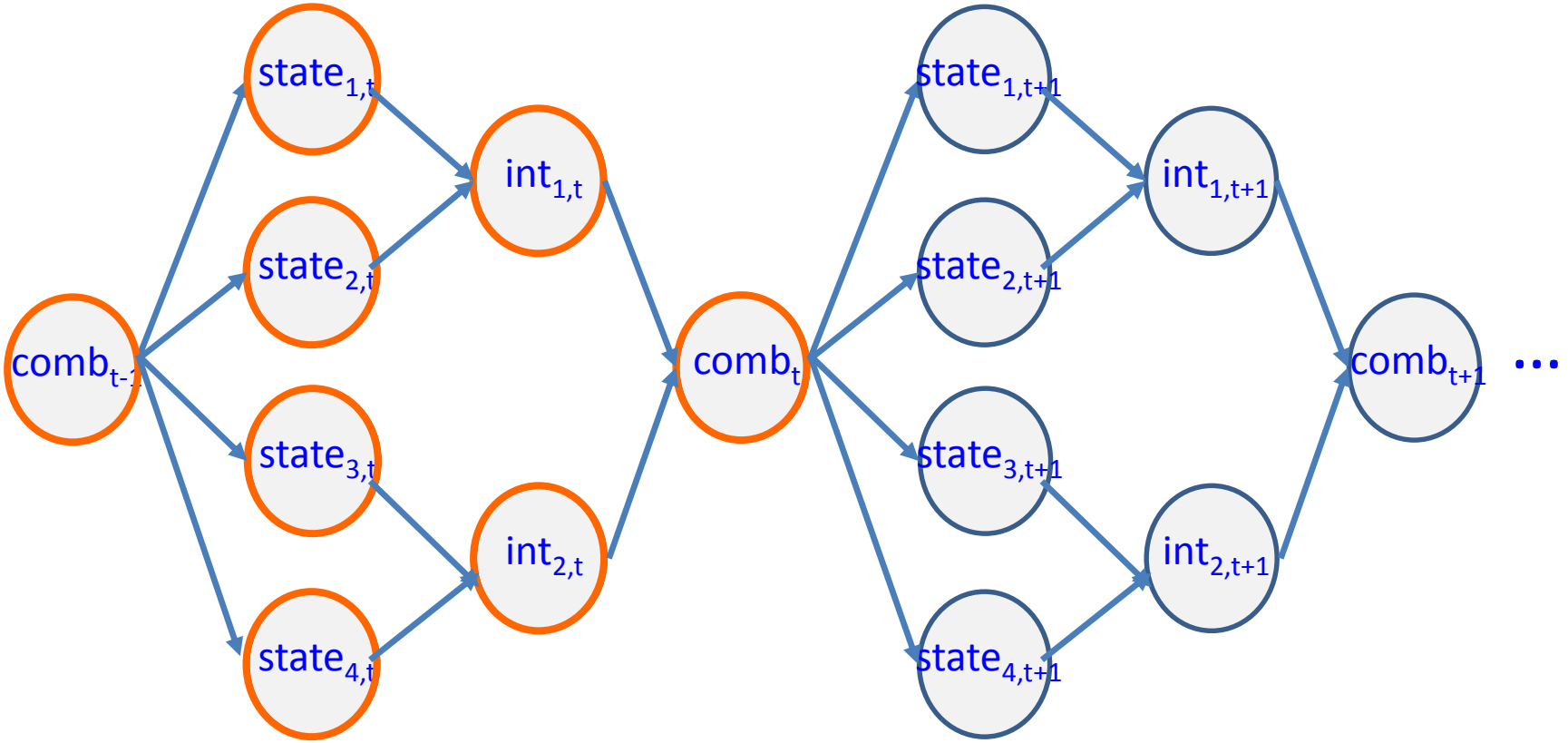


Branch & Combine Emulation

Change topology to reduce pebble complexity

- Combine m CPU states to 1 combined state
- Branch one step computation from it

Claim: pebble complexity = O(log m)



Branch & Combine Emulation

Change topology to reduce pebble complexity

- Combine m CPU states to 1 combined state
- Branch one step computation from it

Claim: pebble complexity = O(log m)

- Combine step
  - Build “Merkle tree” on CPU states
  - Combined state = root
- Branch step
  - Authentication & one step computation

Hiding Step for PRAM

$\mathsf{Garb}(P, x) = (\mathsf{ST\text{-}Garb}(P_{\mathsf{hide}}, x_{\mathsf{hide}}))$

- Replace ORAM by Oblivious PRAM [BCP16]
  - also puncturable

Summary and Open Problems

- Feasibility of crypto for PRAM based on iO via succinct garbled PRAM
- Adaptive succinct garbled (Paralle) RAM with persistent memory (next talk) [ACC+15,CCHR15]
- Open: FHE for RAM/PRAM?
- Open: Crypto for PRAM without iO
  - ABE for RAM/PRAM based on LWE?
- Other parallel model?

Thank you! Questions?