NDSS Symposium 2020

MACAO: A Maliciously-Secure and Client-Efficient Active ORAM Framework

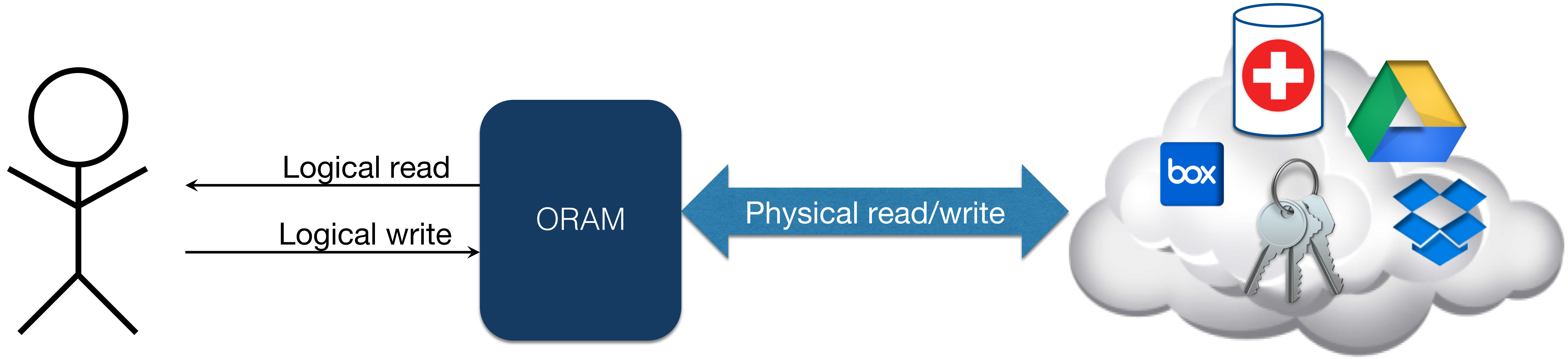
Thang Hoang[†], Jorge Guajardo[‡], Attila A. Yavuz[†]

[†]CSE, University of South Florida
hoangm@mail.usf.edu, attilaayavuz@usf.edu

[‡]Robert Bosch LLC – RTC
Jorge.GuajardoMerchan@us.bosch.com

Oblivious RAM

- Oblivious Random Access Machine (ORAM) allows a client to hide the access pattern when accessing data stored on untrusted memory.



ORAM applications: Cloud storage-as-a-service (personal data storage, health-record database, password management), searchable encryption, secure multiparty computation

Oblivious RAM – Timeline

- ORAM was first introduced by Goldreich for software protection
- Recent attempts focused on reducing ORAM communication overhead

1987
- **Square-root ORAM** (STOC)
  $O(\sqrt{N})$ comm.

1996
- **Hierarchical ORAM** (JACM)
  $O(\log^3 N)$ comm.
- **ORAM lower bound** (JACM)

2011
- **Partition ORAM** (NDSS)
  $O(\log N)$ comm.
  $O(\sqrt{N})$ client storage
- **Tree-ORAM** (AsiaCrypt)
  $O(1)$ client storage
  $O(\log^2 N)$ comm.

2013
- **Path-ORAM** (CCS)
  $O(\log N)$ comm.
  $O(\lambda)$ client storage
- **Multi-cloud ObliviStore** (CCS)
  $O(\sqrt{N})$ client storage
  $O(1)$ comm.

2014
- **Path-PIR** (NDSS)
  $O(\log N)$ comm.
- **Apon et al.** (PKC)
  FHE
  $O(1)$ comm.

2015
- **Circuit-ORAM** (CCS)
  $O(\log N)$ comm.
- **C-ORAM** (CCS)
  Insecure
- **Bucket-ORAM**
  FHE

2016
- **Onion-ORAM** (TCC)
  $O(1)$ comm.
  AHE

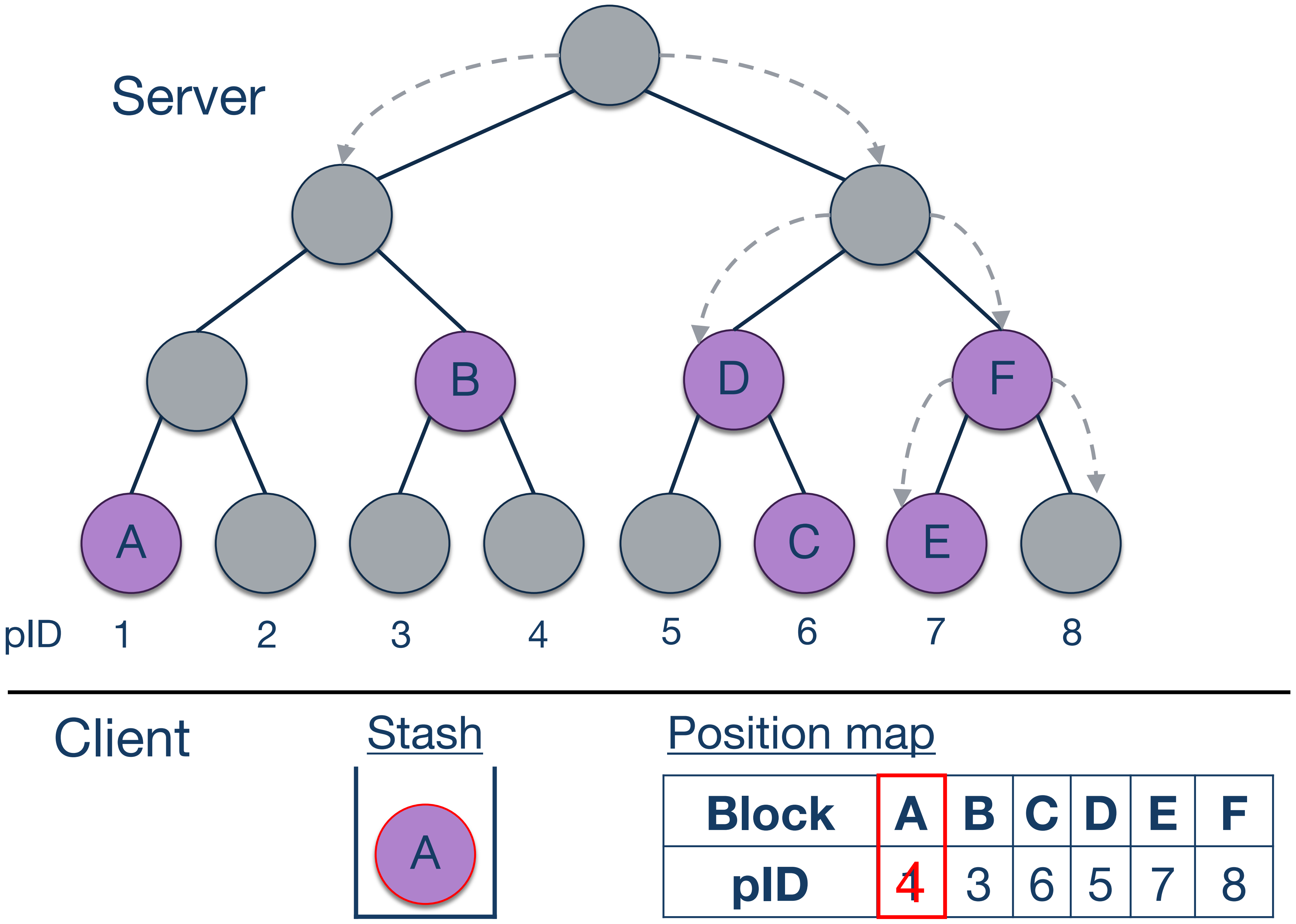
2017
- **S³ORAM** (CCS)
  $O(1)$ comm.
  $O(1)$ client storage
  semi-honest security

2018
- **Passive ORAM lower bound** (Crypto)
- **2-server ORAM** (Asiacrypt)
  $O(N)$ computation

2019
- **Ring-Onion ORAM** (CCS)
  $O(1)$ comm.
  $O(\log N)$ client storage
  semi-honest security

Tree-ORAM Paradigm [SCSL11]

- Binary tree data structure
- Block data located somewhere in the tree path
- Empty nodes are filled with dummy data



Position map

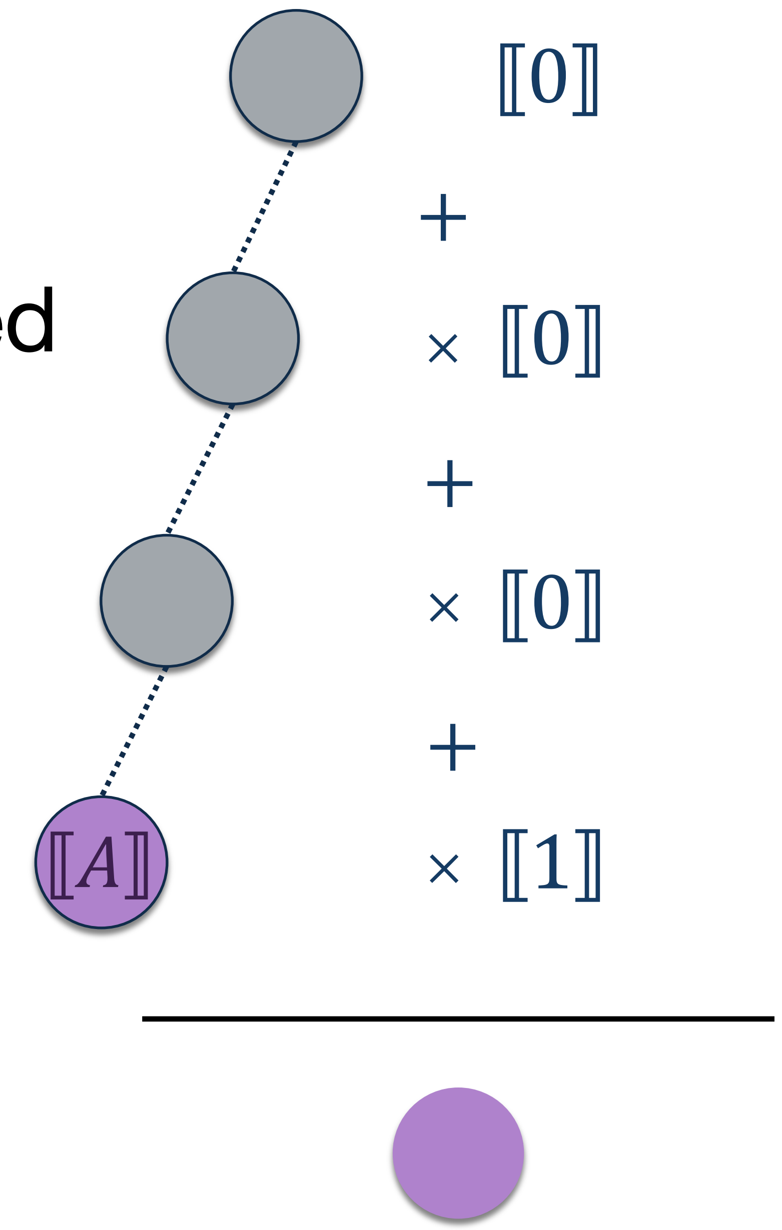
Block	A	B	C	D	E	F
pID	4	3	6	5	7	8

General Access Protocol

1. Get pID of A: 1
2. **Retrieve** path of A
3. Update A (if needed)
4. Randomly select new path for A: 4
5. **Evict**

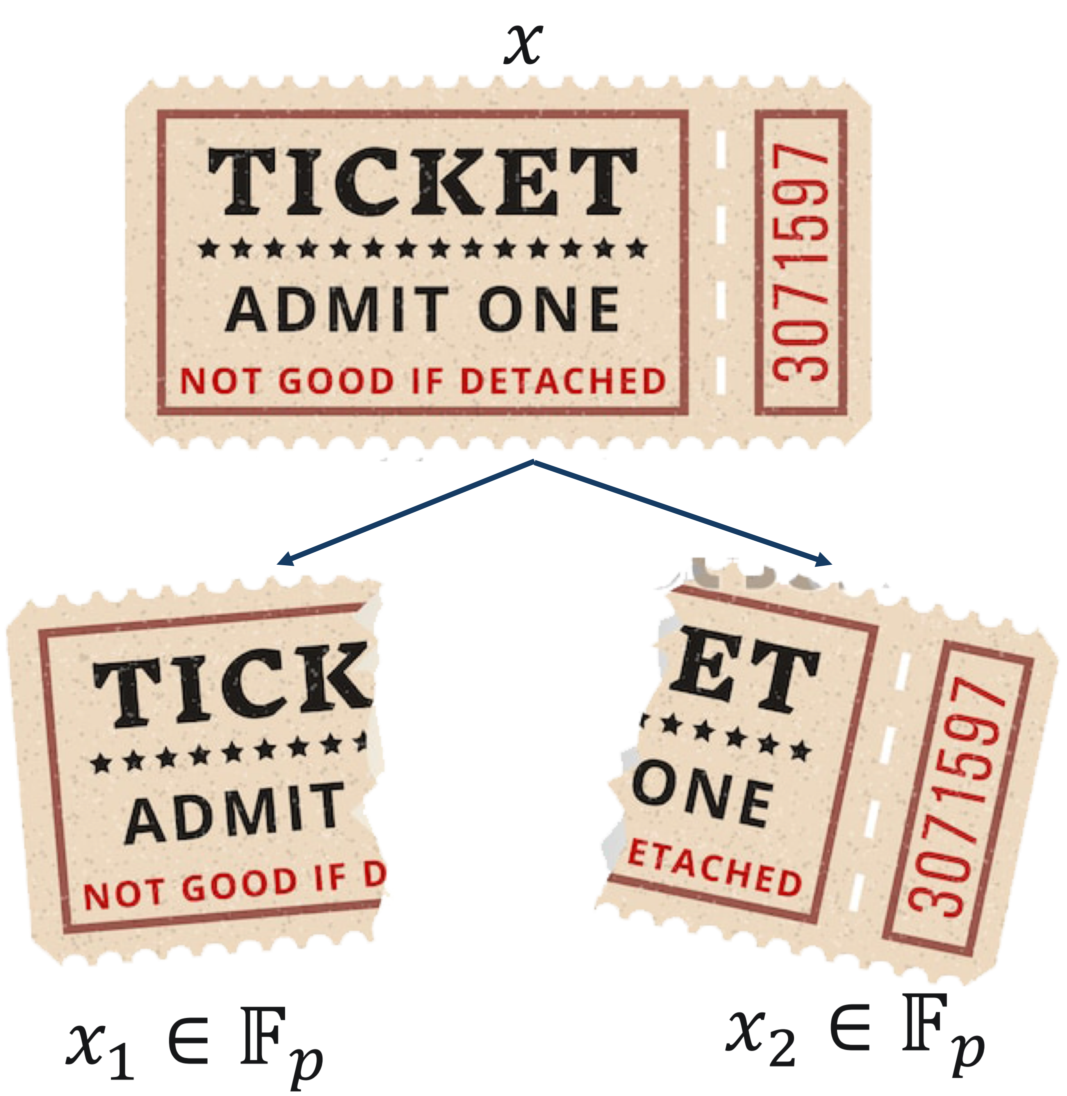
PIR-based ORAM: Malicious Security Concern

- Due to unit vectors created in retrieval phase
  - Contain only one element 1, while others are 0
  - Malicious adversary can tamper with the blocks corresponding to elements “0”
    - Computation result is still correct ➔ cannot be detected by client
    - Learn real block positions
    - Access pattern leakage



MACAO Framework

- Based on (authenticated) additive secret sharing [DPSZ11]



s.t. $x = x_1 + x_2$

- $x \in F_p$ is authenticated shared if each party P_i has random values $x_i, \alpha_i, m_i \in F_p$ s.t.

$$x = \sum_i x_i$$

Random global MAC key →

$$\alpha = \sum_i \alpha_i$$

$$\alpha x = \sum_i m_i$$

- Authenticated share of x is denoted as $\langle x \rangle = (\llbracket x \rrbracket, \llbracket \alpha x \rrbracket)$

Any **linear** function of shared values can be computed locally

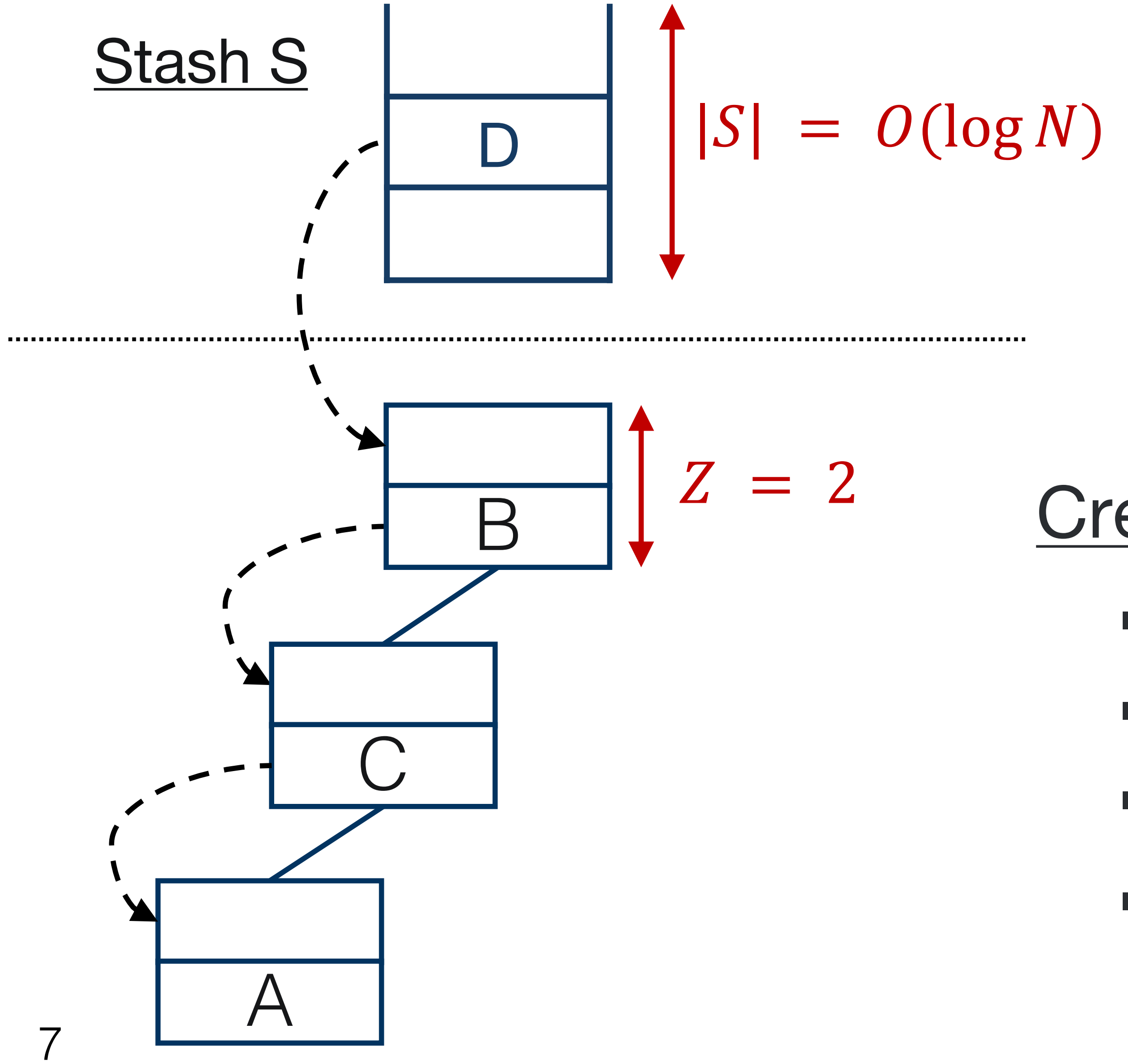
- Given constants v_1, v_2 and shared values $\llbracket x \rrbracket, \llbracket y \rrbracket$

$$v_1 \cdot \llbracket x \rrbracket + v_2 \cdot \llbracket y \rrbracket = \llbracket v_1 x + v_2 y \rrbracket = \llbracket z \rrbracket$$

MACAO Framework

Harness Circuit-ORAM eviction [WCS15] and permutation matrix [HOY+17] principles

- $O(1)$ client bandwidth overhead
- Bucket size $Z = O(1)$
- Each eviction takes a block from the stash and writes it back to the tree



Circuit-ORAM Eviction Principle:

- Only scan once from root to leaf
- For each level, pick or drop (at most) 1 block
- At any time, can only hold (at most) 1 block

Create $(H+1)$ permutation matrices $\mathbf{I}_h$ sized $(Z+1)\times(Z+1)$ s.t.

- $\mathbf{I}_h[j, Z+1] = 1$: Pick the block at index j
- $\mathbf{I}_h[1, j] = 1$: Drop the holding block to index j
- $\mathbf{I}_h[1, Z+1] = 1$: Move the holding block to next level $h+1$
- $\mathbf{I}_h[j+1, j] = 1$: Keep the block at index j remain

MACAO Framework

Two main schemes

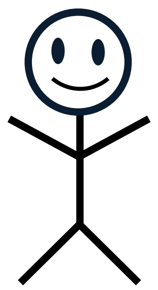
- $\boldsymbol{\Pi_{\mathrm{rss}}}$
  - Replicated secret sharing (RSS)
  - 3-server setting with honest majority
- $\boldsymbol{\Pi_{\mathrm{spdz}}}$
  - SPDZ secret sharing
  - General ℓ-server setting with dishonest majority

MACAO Framework - Π_{rss} scheme

<u>Retrieval</u>

- Select query $\mathbf{q} = (0, \ldots, 1, \ldots, 0)^{H+1}$

1. **XOR-PIR**: a pair of PIR queries $\left(\mathbf{q}_i^{(1)}, \mathbf{q}_i^{(2)}\right)$ <u>per authenticated share</u> $\langle \mathbf{T} \rangle_i$
   - $\mathbf{q}_i^{(1)} \leftarrow_\$ \{0,1\}^{H+1}$, $\mathbf{q}_i^{(2)} \leftarrow \mathbf{q} \oplus \mathbf{q}_i^{(1)}$



$\mathbf{q}_1^{(2)} \mathbf{q}_0^{(2)}$

$\mathbf{q}_1^{(1)} \mathbf{q}_0^{(1)}$

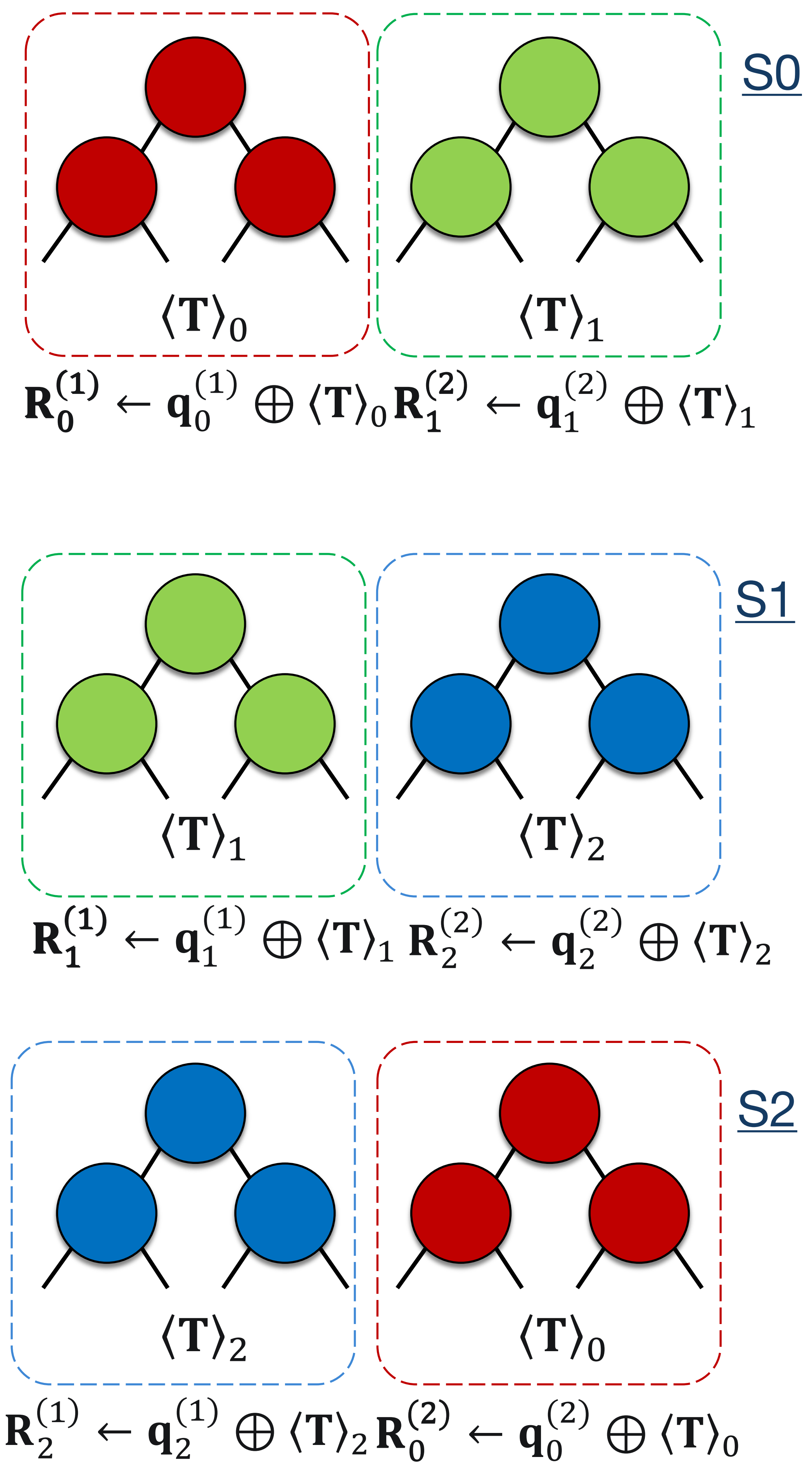
$\langle \mathbf{B} \rangle_1 \leftarrow \mathbf{R}_1^{(1)} \oplus \mathbf{R}_1^{(2)}$

$\langle \mathbf{B} \rangle_0 \leftarrow \mathbf{R}_0^{(1)} \oplus \mathbf{R}_0^{(2)}$

$\langle \mathbf{B} \rangle_2 \leftarrow \mathbf{R}_2^{(1)} \oplus \mathbf{R}_2^{(2)}$

$(\mathbf{X}, \mathbf{Y}) \leftarrow \langle \mathbf{B} \rangle_0 + \langle \mathbf{B} \rangle_1 + \langle \mathbf{B} \rangle_2$

Check if $\alpha \mathbf{X} =? \mathbf{Y}$



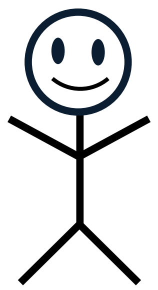
MACAO Framework - Π_{rss} scheme

Retrieval

- Select query $\mathbf{q} = (0, \dots, 1, \dots, 0)^{H+1}$

2. RSS-PIR: two RSS queries $(\mathbf{q}_i, \boldsymbol{q}_{i+1})$ per server S_i

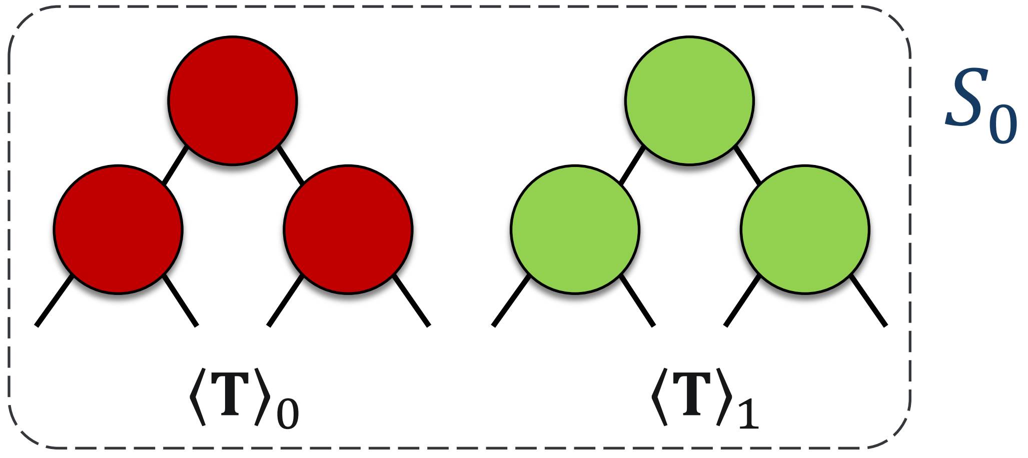
- $\mathbf{q}_0 + \mathbf{q}_1 + \mathbf{q}_2 = \mathbf{q}$, where $\mathbf{q}_i \leftarrow_\$ \mathbb{F}_p^{H+1}$



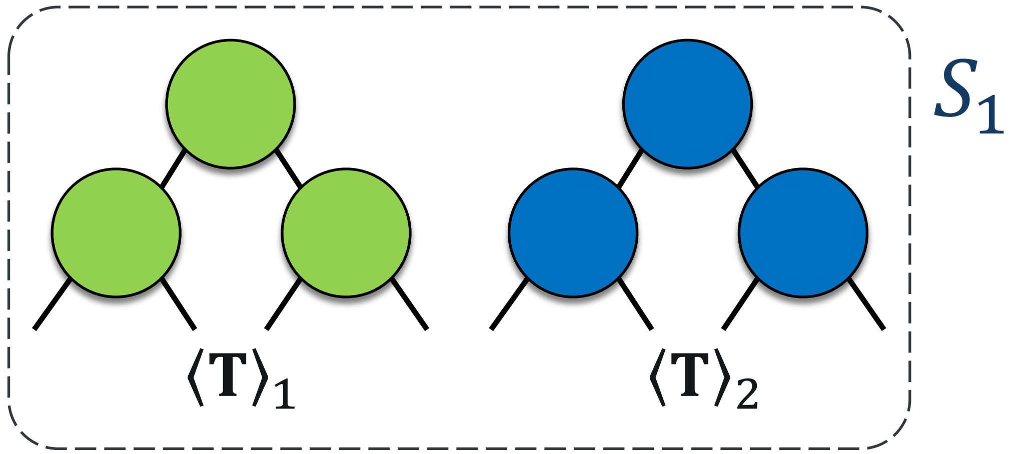
$\mathbf{q}_0, \mathbf{q}_1, \mathbf{q}_2$

$(\mathbf{X}, \mathbf{Y}) \leftarrow \langle\mathbf{R}\rangle_0 + \langle\mathbf{R}\rangle_1 + \langle\mathbf{R}\rangle_2$

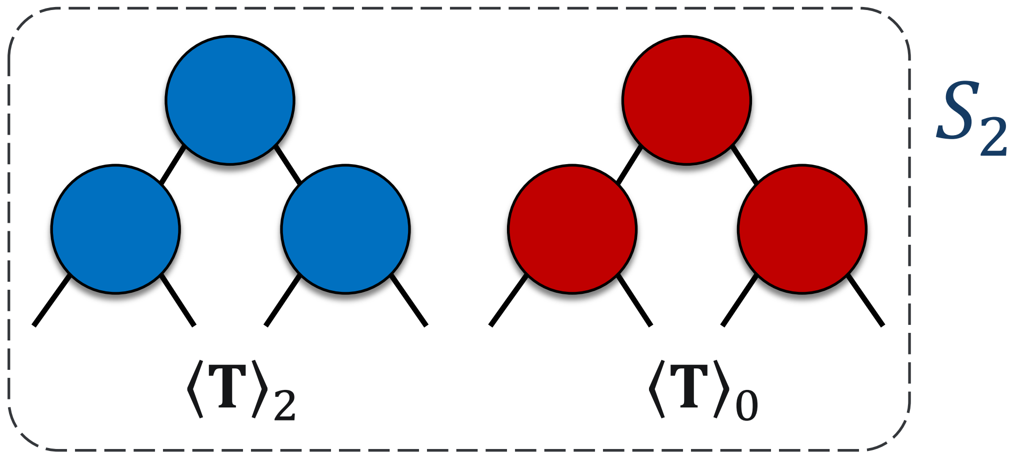
Check if $\alpha\mathbf{X} =? \mathbf{Y}$



$\langle\mathbf{R}\rangle_0 \leftarrow \mathbf{q}_0 \cdot \langle\mathbf{T}\rangle_0 + \mathbf{q}_1 \cdot \langle\mathbf{T}\rangle_0 + \mathbf{q}_0 \cdot \langle\mathbf{T}\rangle_1$



$\langle\mathbf{R}\rangle_1 \leftarrow \mathbf{q}_1 \cdot \langle\mathbf{T}\rangle_1 + \mathbf{q}_2 \cdot \langle\mathbf{T}\rangle_1 + \mathbf{q}_1 \cdot \langle\mathbf{T}\rangle_2$

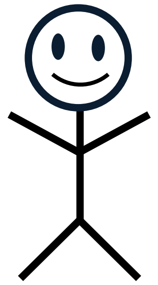


$\langle\mathbf{R}\rangle_2 \leftarrow \mathbf{q}_2 \cdot \langle\mathbf{T}\rangle_2 + \mathbf{q}_0 \cdot \langle\mathbf{T}\rangle_2 + \mathbf{q}_2 \cdot \langle\mathbf{T}\rangle_0$

MACAO Framework - Π_{rss} scheme

Eviction: based on RSS-based matrix multiplication protocol

- RSS-share of evicting block $\mathbf{B}$ and $(H+1)$ RSS-shares of permutation matrices $\mathbf{M}_h$



$\mathbf{M}_h = [\![\mathbf{M}_h]\!]_0, [\![\mathbf{M}_h]\!]_1, [\![\mathbf{M}_h]\!]_2$

$\mathbf{B} = [\![\mathbf{B}]\!]_0, [\![\mathbf{B}]\!]_1, [\![\mathbf{B}]\!]_2$

RSSMatMult($[\![\mathbf{U}]\!], [\![\mathbf{V}]\!]$)

- $\mathbf{X}_i \leftarrow [\![\mathbf{U}]\!]_i \times [\![\mathbf{V}]\!]_i + [\![\mathbf{U}]\!]_{i+1} \times [\![\mathbf{V}]\!]_i + [\![\mathbf{U}]\!]_i \times [\![\mathbf{V}]\!]_{i+1}$
- S_i sends $\left(\mathbf{R}_{i-1}^{(i)}, \mathbf{R}_{i-1}^{(i)}\right)$ to S_{i-1}, $\left(\mathbf{R}_{i-1}^{(i)}, \mathbf{R}_{i-1}^{(i)}\right)$ to S_{i+1}, where $\mathbf{X}_i = \sum_{j=0}^{2} \mathbf{R}_j^{(i)}$

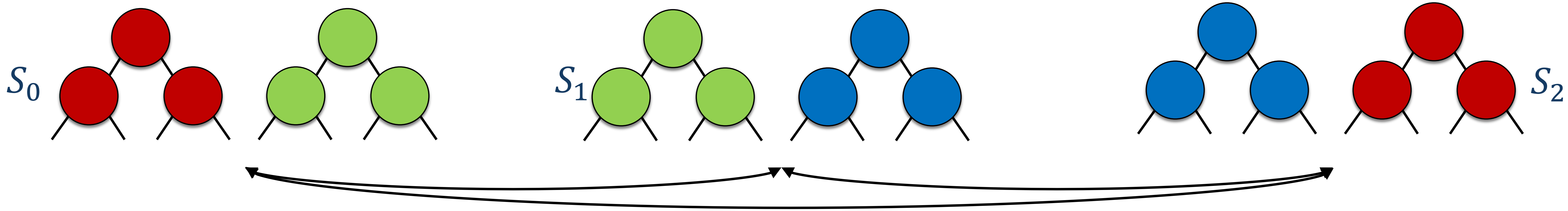
Output: $[\![\mathbf{U}\times\mathbf{V}]\!]_i \leftarrow \mathbf{R}_i^{(0)} + \mathbf{R}_i^{(1)} + \mathbf{R}_i^{(2)}$
$[\![\mathbf{U}\times\mathbf{V}]\!]_{i+1} \leftarrow \mathbf{R}_{i+1}^{(0)} + \mathbf{R}_{i+1}^{(1)} + \mathbf{R}_{i+1}^{(2)}$

MACCheck($\langle\mathbf{T}\rangle$)

- $x \leftarrow \sum_h \sum_i \sum_j r^t [\![\mathbf{T}[i,j]]\!]_h$
- $y \leftarrow \sum_h \sum_i \sum_j r^t [\![\alpha\mathbf{T}[i,j]]\!]_h$
- Pass if $\alpha \cdot x =? \, y$

(Random linear combination)

Jointly execute MACCheck($\langle\mathbf{T}\rangle_h$) to verify eviction integrity



$[\![\mathbf{T}_h']\!] \leftarrow$ RSSMatMult($[\![\mathbf{M}_h]\!], [\![\mathbf{T}_h]\!]$)

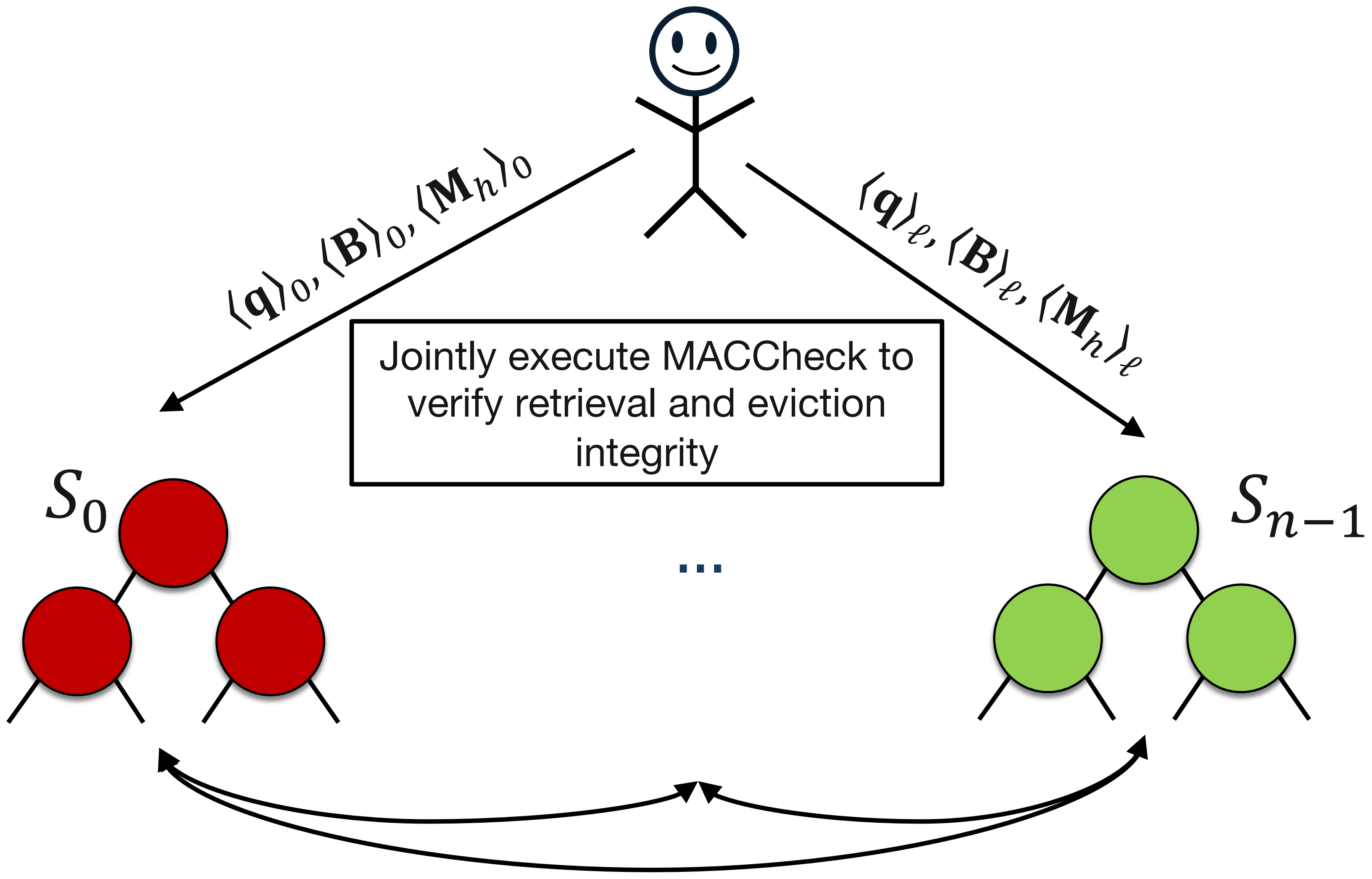
$[\![\alpha\mathbf{T}_h']\!] \leftarrow$ RSSMatMult($[\![\mathbf{M}_h]\!], [\![\alpha\mathbf{T}_h]\!]$)

$\mathbf{T}_h$: holding block and current blocks at level h

MACAO Framework - Π_{spdz} scheme

Both retrieval and eviction are based on SPDZ-based authenticated matrix multiplication protocol

- Retrieval: Select query $\langle \mathbf{q} \rangle = (\langle 0 \rangle, \ldots, \langle 1 \rangle, \ldots, \langle 0 \rangle)^{H+1}$
- Eviction: SPDZ-share of evicting block $\mathbf{B}$ and $(H+1)$ SPDZ-shares of permutation matrices $\mathbf{M}_h$



SPDZMatMult($\langle \mathbf{q} \rangle, \langle \mathbf{T} \rangle$) / SPDZMatMult($\langle \mathbf{M}_h \rangle, \langle \mathbf{T}_h \rangle$)

SPDZMatMult($\langle \mathbf{U} \rangle, \langle \mathbf{V} \rangle$)

Initialization: Each S_i has $\langle \mathbf{A} \rangle_i, \langle \mathbf{B} \rangle_i, \langle \mathbf{C} \rangle_i$, authenticated shares of Beaver triples ($\mathbf{C} = \mathbf{A} \times \mathbf{B}$, $\alpha \mathbf{C} = \alpha(\mathbf{A} \times \mathbf{B})$)

- $[\![\mathbf{E}]\!]_i \leftarrow [\![\mathbf{U}]\!]_i - [\![\mathbf{A}]\!]_i$, $[\![\mathbf{P}]\!]_i \leftarrow [\![\mathbf{V}]\!]_i - [\![\mathbf{B}]\!]_i$
- Open $\mathbf{E}$ and $\mathbf{P}$
- MACCheck($\mathbf{E}$) and MACCheck($\mathbf{P}$)

Output: $[\![\mathbf{U} \times \mathbf{V}]\!]_i \leftarrow [\![\mathbf{C}]\!]_i + \mathbf{E} \times [\![\mathbf{B}]\!]_i + [\![\mathbf{A}]\!]_i \times \mathbf{P} + \mathbf{E} \times \mathbf{P}$
$[\![\alpha \mathbf{U} \times \mathbf{V}]\!]_i \leftarrow [\![\alpha \mathbf{C}]\!]_i + \mathbf{E} \times [\![\mathbf{B}]\!]_i + [\![\mathbf{A}]\!]_i \times \mathbf{P} + [\![\alpha]\!]_i \mathbf{E} \times \mathbf{P}$

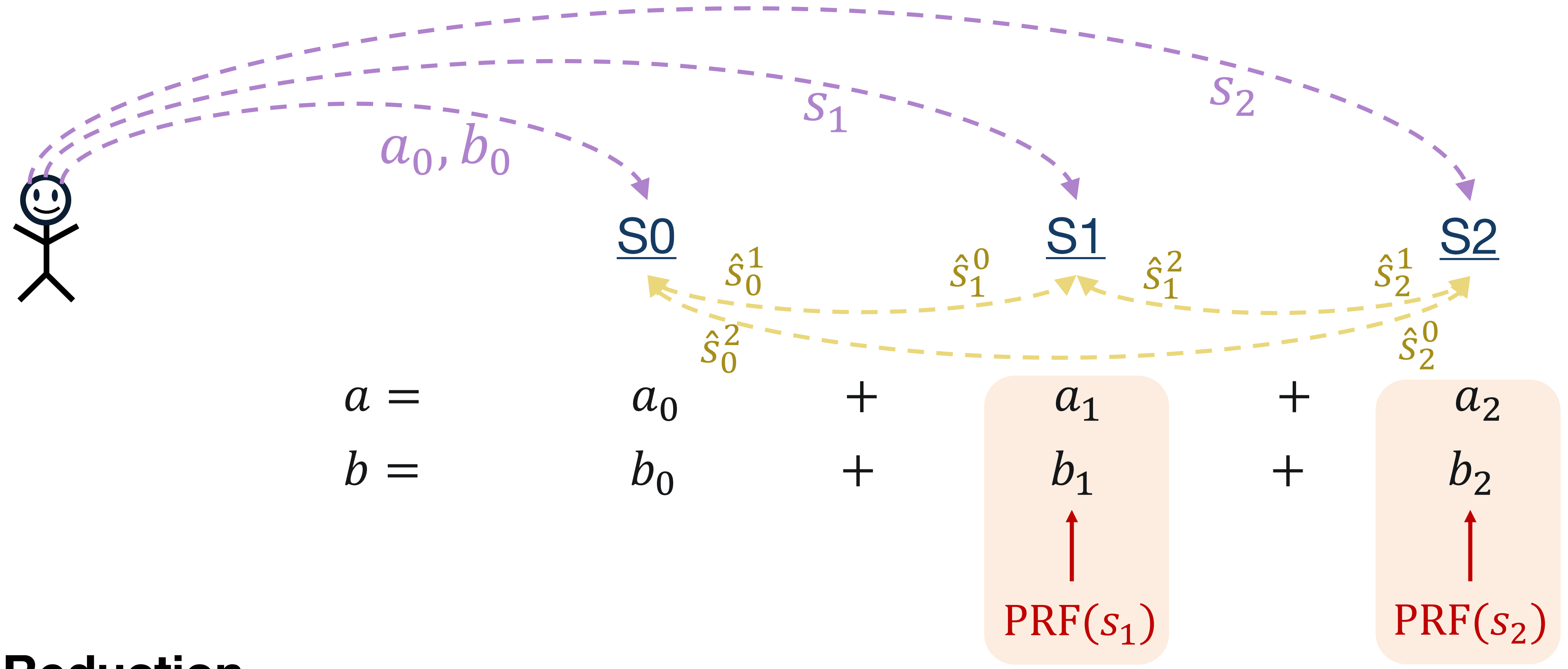
MACCheck($\mathbf{T}, [\![\alpha \mathbf{T}]\!]$)

- $x \leftarrow \sum_i \sum_j r^t \mathbf{T}[i,j]$
- $y \leftarrow \sum_i \sum_j r^t [\![\alpha \mathbf{T}[i,j]]\!]$
- Pass if $\alpha \cdot x = y$

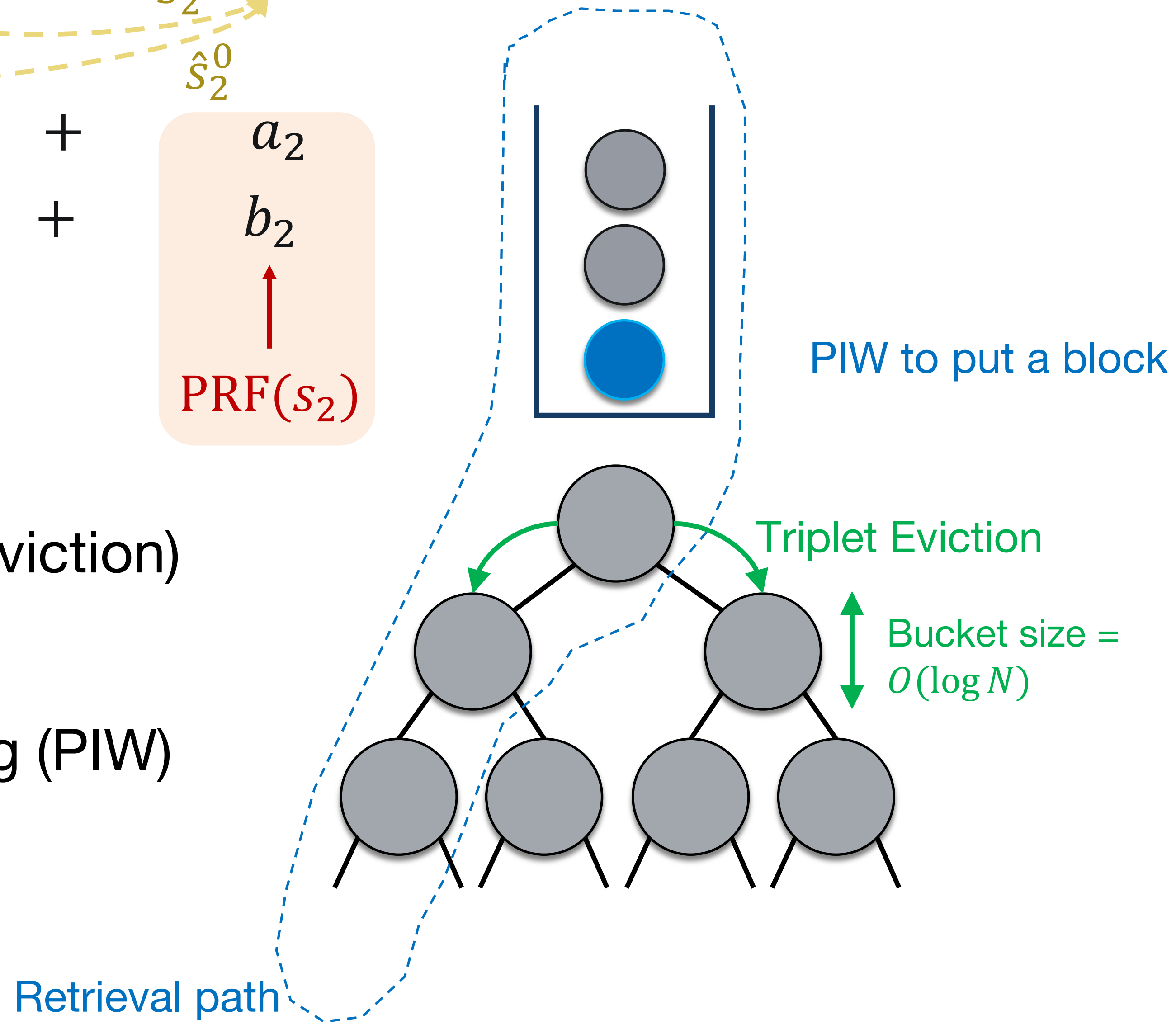
(Random linear combination)

MACAO Framework - Extension

- **Bandwidth Reduction**
  - Pseudo-random function (PRF) to generate additive shares locally [CDI05, DSZ14, RWTS+17]

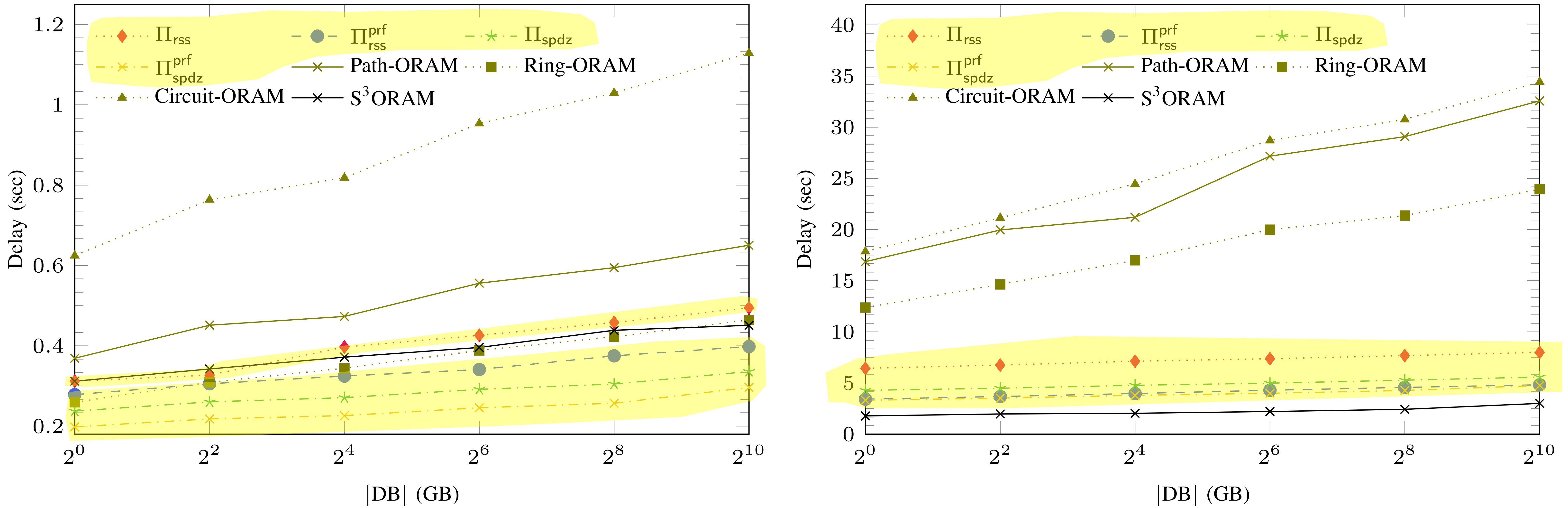


- **Client Storage Reduction**
  - Stash sized $O(\log N)$ was stored at the client (due to Circuit-ORAM eviction)
  - Two ways to reduce client stash storage
    1. Store stash at the server-side, and use Private-Information Writing (PIW) to privately put the block into the stash
    2. Triplet Eviction [SvDFR+16]
       - Stash not needed in place of $O(\log N)$ bucket size)



MACAO Framework – Performance (1/3)

- MACAO schemes were 7× faster than single-server ORAMs and up to 1.5× slower than S³ORAM

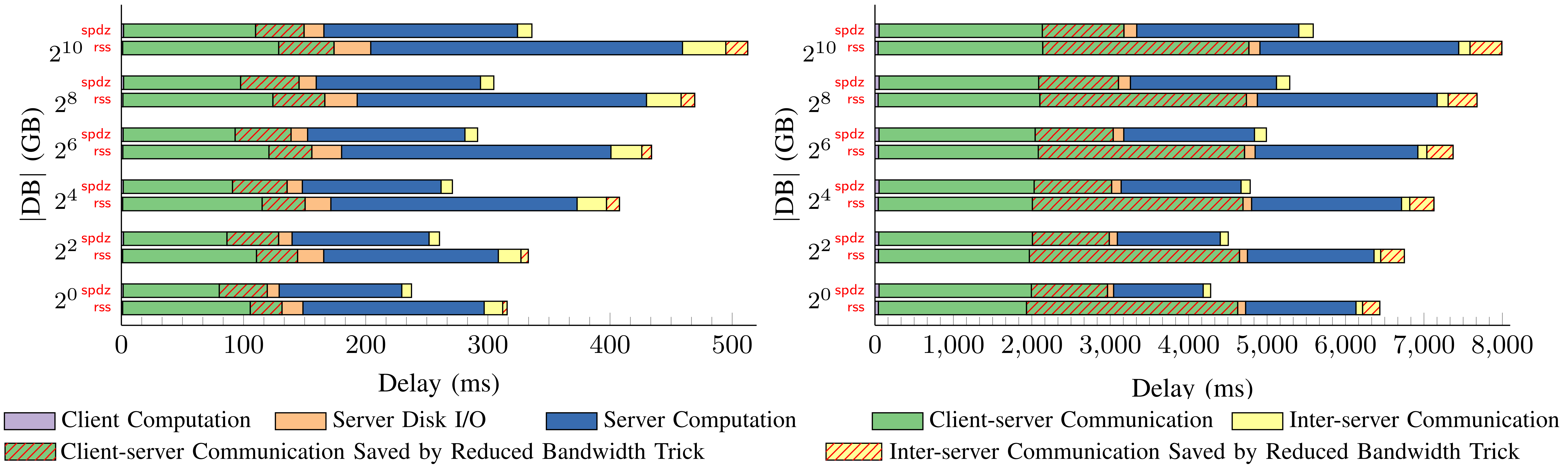


End-to-end delay of MACAO schemes and their counterparts.

Configuration: Library: NTL, tomcrypt, zeroMQ, pthread; Client: Macbook Pro 2018; Servers: Amazon EC2 c5.4xlarge, EBS-based storage; Client-server bandwidth: 29/5 Mbps; Inter-server bandwidth: 250/250 Mbps; DB Size: 1GB – 1TB; Block size: 4KB, 256KB

MACAO Framework – Performance (2/3)

- Server computation contributed the most portion to the overall delay
- Bandwidth reduction trick significantly reduced the communication costs



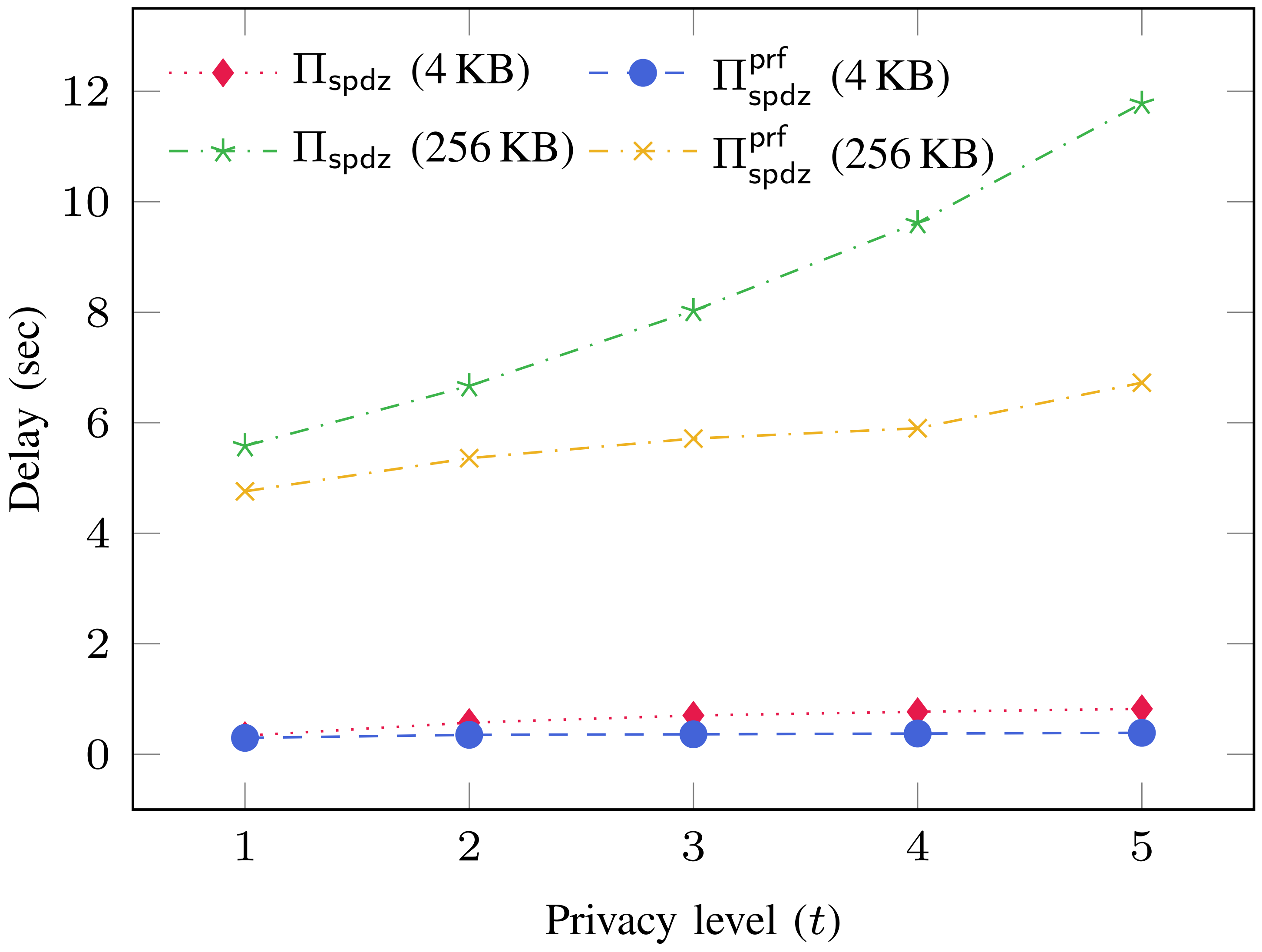
(a) |**b**|= 4 KB **(b)** |**b**|= 256 KB

Cost breakdown of MACAO schemes

Configuration: Library: NTL, tomcrypt, zeroMQ, pthread; Client: Macbook Pro 2018; Servers: Amazon EC2 c5.4xlarge, EBS-based storage; Client-server bandwidth: 29/5 Mbps; Inter-server bandwidth: 250/250 Mbps; DB Size: 1GB – 1TB; Block size: 4KB, 256KB

MACAO Framework – Performance (3/3)

- Bandwidth reduction trick also helped to reduce the delay when increasing number of servers for higher privacy levels



End-to-end delay with varied privacy levels

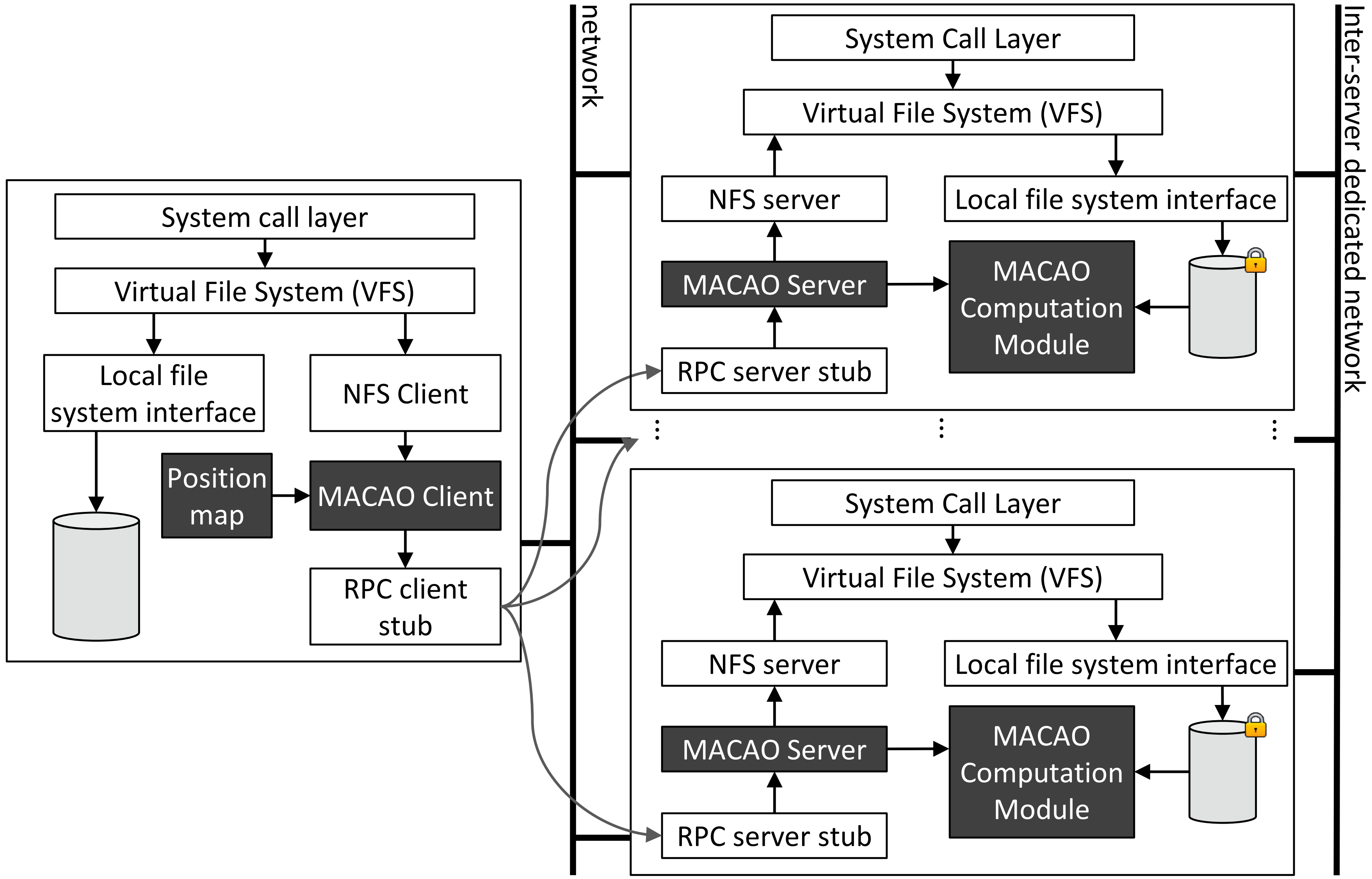
Configuration: Library: NTL, tomcrypt, zeroMQ, pthread; Client: Macbook Pro 2018; Servers: Amazon EC2 c5.4xlarge, EBS-based storage; Client-server bandwidth: 29/5 Mbps; Inter-server bandwidth: 250/250 Mbps; DB Size: 1GB – 1TB; Block size: 4KB, 256KB

Conclusion & Future Work

- Proposed MACAO, a multi-server active ORAM framework providing integrity, access pattern obliviousness against active adversaries, and secure computation capability.
  - Based on Authenticated additive secret sharing and tree ORAM paradigm

Future Work

- Oblivious Distributed File System (ODFS) implementation
- Multi-user Oblivious Storage based on MACAO



The proposed ODFS Model

Thank you for your attention!

?

MACAO code: https://github.com/thanghoang/MACAO

References

- [CDI05] Cramer, Ronald, Ivan Damgård, and Yuval Ishai. "Share conversion, pseudorandom secret-sharing and applications to secure computation." In Theory of Cryptography Conference, pp. 342-362. Springer, Berlin, Heidelberg, 2005.
- [SCSL11] Shi, Elaine, T-H. Hubert Chan, Emil Stefanov, and Mingfei Li. "Oblivious RAM with O ((logN) 3) worst-case cost." In International Conference on The Theory and Application of Cryptology and Information Security, pp. 197-214. Springer, Berlin, Heidelberg, 2011.
- [DPSZ11] Damgård, Ivan, Valerio Pastro, Nigel Smart, and Sarah Zakarias. "Multiparty computation from somewhat homomorphic encryption." In Annual Cryptology Conference, pp. 643-662. Springer, Berlin, Heidelberg, 2012.
- [DSZ14] Demmler, Daniel, Thomas Schneider, and Michael Zohner. "Ad-hoc secure two-party computation on mobile devices using hardware tokens." In 23rd {USENIX} Security Symposium ({USENIX} Security 14), pp. 893-908. 2014.
- [WCS15] Wang, Xiao, Hubert Chan, and Elaine Shi. "Circuit oram: On tightness of the goldreich-ostrovsky lower bound." In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, pp. 850-861. 2015
- [SvDFR+16] Devadas, Srinivas, Marten van Dijk, Christopher W. Fletcher, Ling Ren, Elaine Shi, and Daniel Wichs. "Onion ORAM: A constant bandwidth blowup oblivious RAM." In Theory of Cryptography Conference, pp. 145-174. Springer, Berlin, Heidelberg, 2016.
- [HOY+17] Hoang, Thang, Ceyhun D. Ozkaptan, Attila A. Yavuz, Jorge Guajardo, and Tam Nguyen. "S3oram: A computation-efficient and constant client bandwidth blowup oram with shamir secret sharing." In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pp. 491-505. 2017.
- [RWTS+17] Riazi, M. Sadegh, Christian Weinert, Oleksandr Tkachenko, Ebrahim M. Songhori, Thomas Schneider, and Farinaz Koushanfar. "Chameleon: A hybrid secure computation framework for machine learning applications." In Proceedings of the 2018 on Asia Conference on Computer and Communications Security, pp. 707-721. 2018.

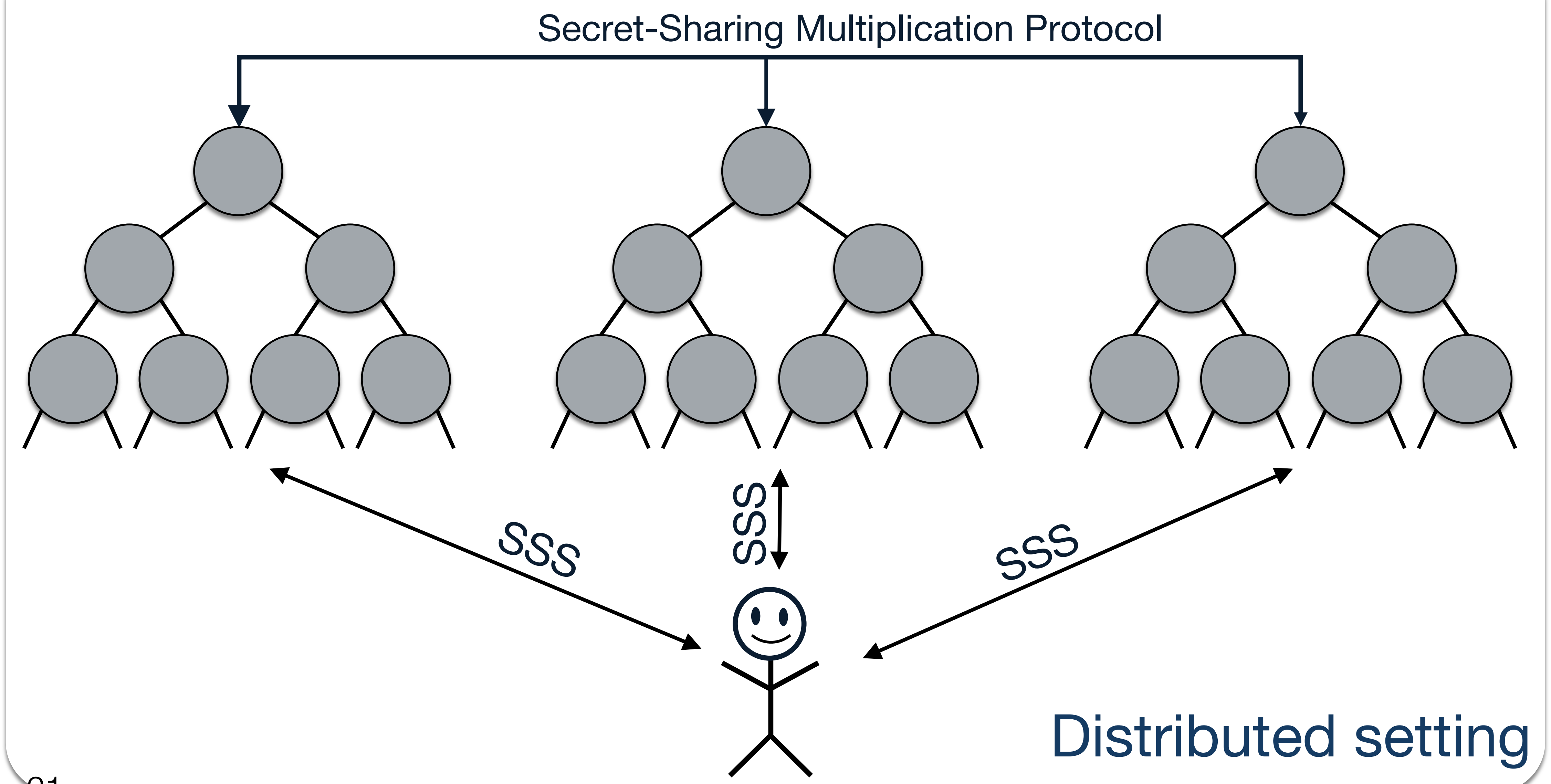
Our Motivation

- Single-server active ORAM (e.g., Onion-ORAM) offers O(1) bandwidth blowup and malicious security
  - High computation overhead due to Homomorphic Encryption (HE)
  - Cut-and-choose technique → incurs higher communication and computation overhead for malicious security
- Multi-server active ORAM (i.e., S^3ORAM) offers O(1) bandwidth with efficient computation
  - However, it only offers semi-honest security

An efficient multi-server ORAM with active security?

S³ORAM [HOY+17]

- Tree-ORAM paradigm
- Exploit the efficiency of multi-party computation in distributed setting
  - Shamir Secret Sharing (SSS) Scheme
  - **Retrieval:** SSS-Private Information Retrieval – **Eviction:** Permutation Matrix



S³ORAM System Model:

- $\ell \geq 2t + 1$ servers
- # colluding servers <= t
- All servers are **semi-honest**

MACAO Framework – Summary

Asymptotic comparison of state-of-the-art ORAM schemes.

Scheme	Bandwidth Overhead†		Block Size*	Client Block Storage‡	# servers§	Security	Comp. over Enc. Data
	Client-server	Server-server					
Ring-ORAM [53]	$\mathcal{O}(\log N)$	-	$\Omega(1)$	$\mathcal{O}(\log N)$	1	Semi-Honest	×
CKN+18 [16]	$\mathcal{O}(\log N)$	-	$\Omega(\log^2 N)$	$\mathcal{O}(1)$	3	Semi-Honest	×
GKW18 [32]	$\mathcal{O}(\log N)$	-	$\Omega(1)$	$\mathcal{O}(\log N)$	2	Semi-Honest	×
S^3ORAM [33]	$\mathcal{O}(1)$	$\mathcal{O}(\log N)$	$\Omega(\log^2 N)$	$\mathcal{O}(1)$	$2t+1$	Semi-Honest	✓
Path-ORAM [64]	$\mathcal{O}(\log N)$	-	$\Omega(1)$	$\mathcal{O}(\log N)$	1	Malicious	×
Circuit-ORAM [66]	$\mathcal{O}(\log N)$	-	$\Omega(1)$	$\mathcal{O}(\log N)$	1	Malicious	×
SS13 [61]	$\mathcal{O}(1)$	$\mathcal{O}(\log N)$	$\Omega(\log^2 N)$	$\mathcal{O}(\sqrt{N})$	2	Malicious	×
LO13 [42]	$\mathcal{O}(\log N)$	-	$\Omega(1)$	$\mathcal{O}(1)$	2	Malicious	×
Onion-ORAM [22]	$\mathcal{O}(1)$	-	$\Omega(\log^6 N)$	$\mathcal{O}(1)$	1	Malicious	✓
MACAO (Π_{rss})	$\mathcal{O}(1)$	$\mathcal{O}(\log N)$	$\Omega(\log N)$	$\mathcal{O}(\log N)$	3	Malicious	✓
MACAO (Π_{spdz})					$t+1$		

MACAO Security

Definition 1 (*Simulation-based Multi-server ORAM Security with Verifiability*)**.** Considering the ideal and real worlds as follows.

- **Ideal world.** Let $\mathcal{F}_{\text{oram}}$ be an ideal functionality, which maintains the latest version of the database on behalf of the client, and answers the client's requests as follows.
  - **Setup:** Environment $\mathcal{Z}$ provides DB to the client, who sends DB to $\mathcal{F}_{\text{oram}}$. $\mathcal{F}_{\text{oram}}$notifies simulator $\mathcal{S}_{\text{oram}}$ the setup is complete and the DB size. $\mathcal{S}_{\text{oram}}$ returns ok or abort to $\mathcal{F}_{\text{oram}}$. $\mathcal{F}_{\text{oram}}$ returns ok or $\bot$ to client accordingly.
  - **Access:** Environment $\mathcal{Z}$ specifies $\text{op} \in \{\text{read}(\text{bid}, \bot), \text{write}(\text{bid}, \text{data})\}$ as client's input. Client sends op to $\mathcal{F}_{\text{oram}}$. $\mathcal{F}_{\text{oram}}$ notifies $\mathcal{S}_{\text{oram}}$ (without revealing op). If $\mathcal{S}_{\text{oram}}$ returns ok to $\mathcal{F}_{\text{oram}}$, $\mathcal{F}_{\text{oram}}$ sends $\text{data}' \leftarrow \text{DB}[\text{bid}]$ to client, and updates $\text{DB}[\text{bid}] \leftarrow \text{data}$ if $\text{op} = \text{write}$. Client returns data' to $\mathcal{Z}$. If $\mathcal{S}_{\text{oram}}$ returns abort to $\mathcal{F}_{\text{oram}}$, $\mathcal{F}_{\text{oram}}$ returns $\bot$ to client.
- **Real world.** $\mathcal{Z}$ gives the client DB. Client executes Setup protocol with servers $(S_0, \dots, S_{\ell-1})$ on DB. For each access, $\mathcal{Z}$ specifies an input $\text{op} \in \{\text{read}(\text{bid}, \bot), \text{write}(\text{bid}, \text{data})\}$ to client. Client executes Access protocol with servers $(S_0, \dots, S_{\ell-1})$. $\mathcal{Z}$ gets the view of the adversary $\mathcal{A}$ after each access. Client outputs to $\mathcal{Z}$ the accessed block or abort.

A protocol $\Pi_{\mathcal{F}}$ securely realizes $\mathcal{F}_{\text{oram}}$ in the presence of a malicious adversary corrupting t servers iff for any PPT real-world adversary corrupting t servers, there exists a simulator $\mathcal{S}_{\text{oram}}$, such that for all non-uniform, polynomial-time Z, there exists a negligible function negl such that

$$\left|\Pr\left[\text{REAL}_{\Pi_{\mathcal{F}},\mathcal{A},\mathcal{Z}(\lambda)} = 1\right] - \Pr\left[\text{IDEAL}_{\mathcal{F}_{\text{oram}},\mathcal{S}_{\text{oram}},\mathcal{Z}(\lambda)} = 1\right]\right| \leq \text{negl}(\lambda)$$

Theorem 1 (*MACAO security*). MACAO framework is statistically (information-theoretically) secure by Definition 1.