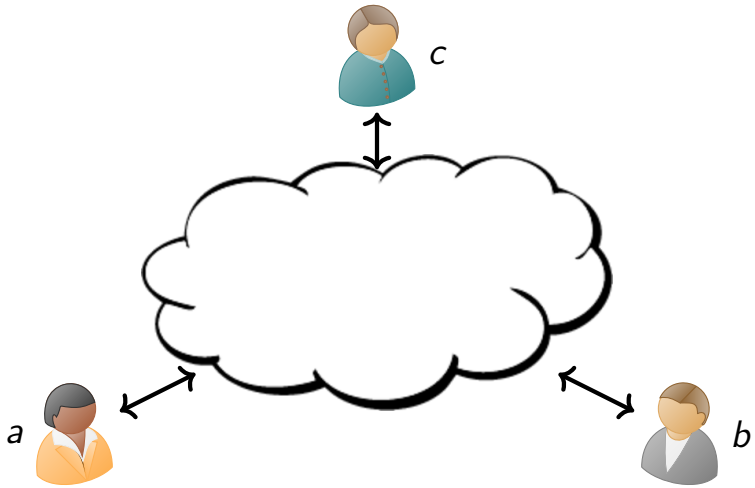
# PRAC: Round-Efficient 3-Party MPC for Dynamic Data Structures

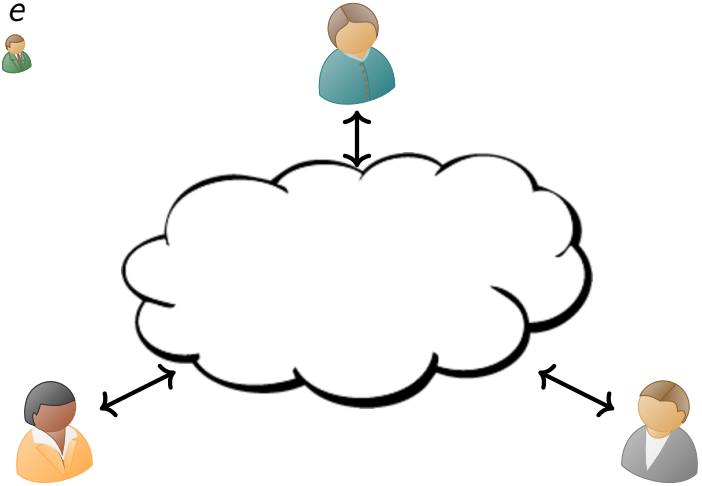
Sajin Sasy, Adithya Vadapalli, **Ian Goldberg**

CrySP Group Meeting, 16 Oct 2023

# MPC: multiparty computation

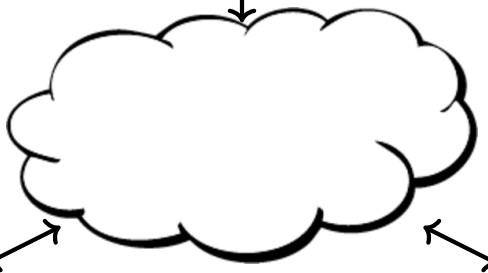


# MPC: multiparty computation

# MPC: multiparty computation

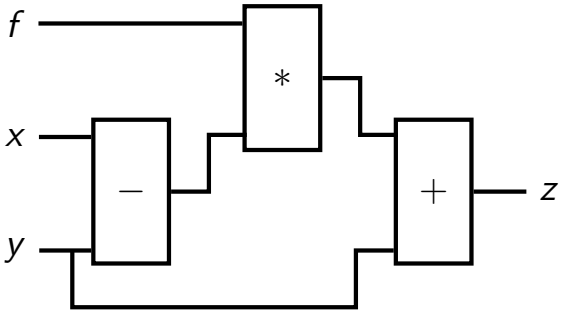

$a_2, b_2, c_2, d_2, e_2$



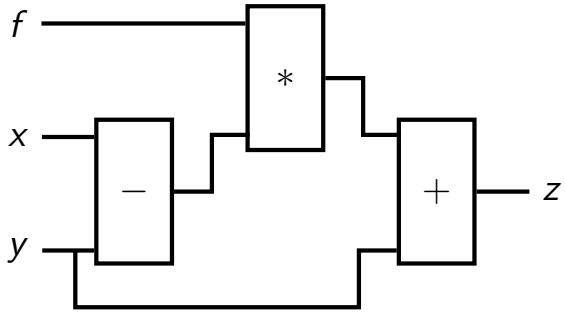
$a_0, b_0, c_0, d_0, e_0$

$a_1, b_1, c_1, d_1, e_1$

# Computation



# Computation




$$z = \begin{cases} y & \text{if } f = 0 \\ x & \text{if } f = 1 \end{cases}$$

# The hard part: memory

$$A = A_0 \oplus A_1, \quad i = i_0 \oplus i_1$$

$$(A_0, i_0), (A_1, i_1) \overset{\text{☁}}{\rightarrow} z_0, z_1$$

$$z_0 \oplus z_1 = (A_0 \oplus A_1)[i_0 \oplus i_1]$$

# The hard part: memory

$$(A_0, i_0, v_0), (A_1, i_1, v_1) \overset{\text{☁}}{\rightarrow} A'_0, A'_1$$

$$(A'_0 \oplus A'_1)[j] = \begin{cases} (A_0 \oplus A_1)[j] & j \neq i_0 \oplus i_1 \\ v_0 \oplus v_1 & j = i_0 \oplus i_1 \end{cases}$$

# The hard part: memory

$$(A_0, i_0, v_0), (A_1, i_1, v_1) \overset{\text{☁}}{\rightarrow} A'_0, A'_1$$

$$(A'_0 \oplus A'_1)[j] = \begin{cases} (A_0 \oplus A_1)[j] & j \neq i_0 \oplus i_1 \\ v_0 \oplus v_1 & j = i_0 \oplus i_1 \end{cases}$$

DORAM: Distributed Oblivious Random Access Memory

# DORAMs: two approaches

Computation-efficient:

e.g., 3P-Circuit ORAM, Ramen, GigaDORAM

Communication-efficient:

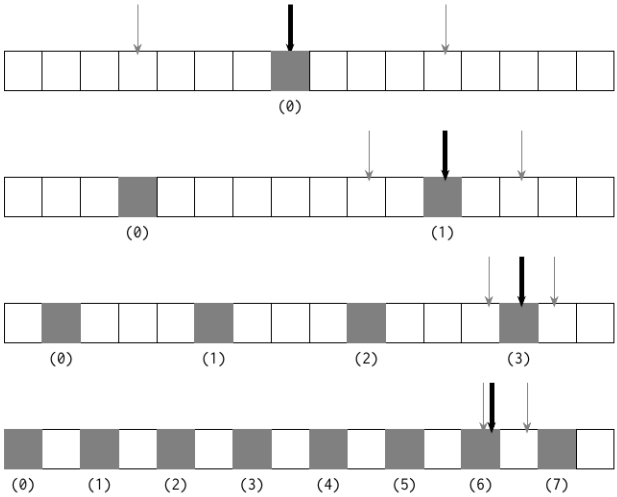
e.g., Floram, Duoram, PRAC

# DORAMs: two approaches

# PRAC: Private Random Access Computations

- General MPC framework for random-access computations
- Underlying DORAM based on Duoram
  (Vadapalli, Henry, Goldberg, USENIX Security 2023)
- Implementation improvements
- Algorithmic improvements
  - use of incremental DPFs and wide DPFs
  - three oblivious data protocols:
    - binary search
    - heaps
    - AVL trees

# Binary search

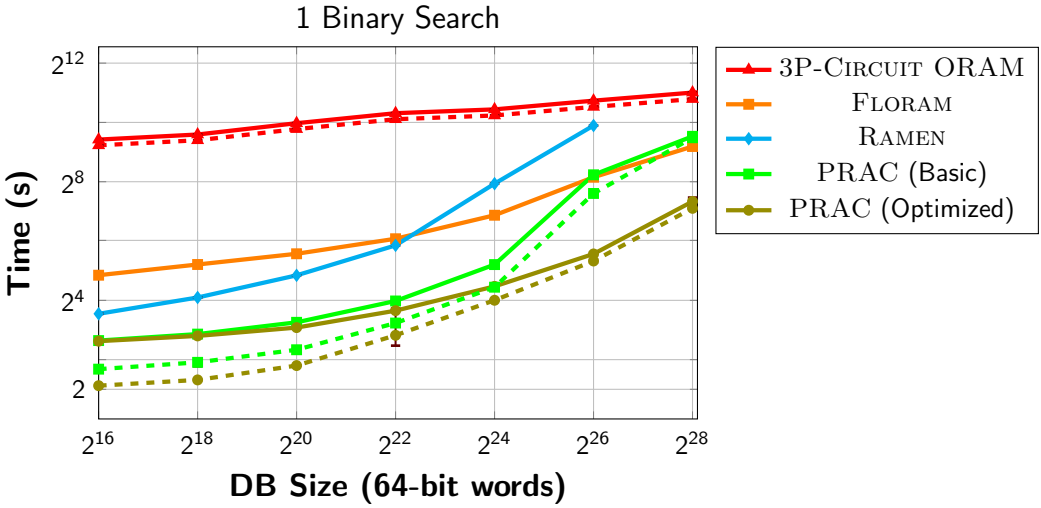


# Binary search

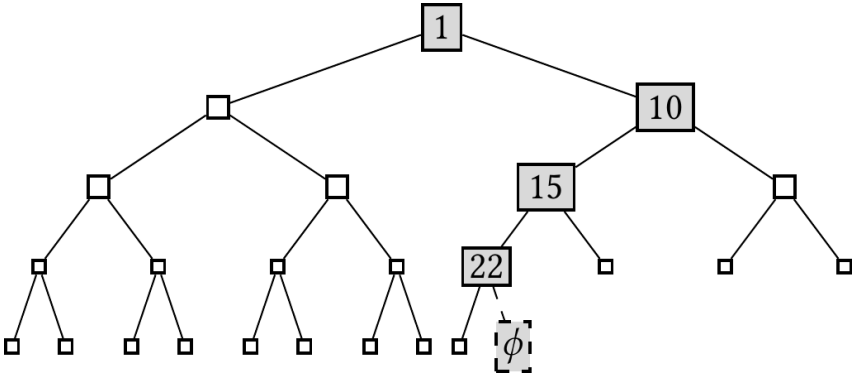
Analytical comparison of MPC binary search

| Protocol | Communication | | Computation |
|---|---|---|---|
| | Bandwidth | Rounds | |
| Floram | $\mathcal{O}(\lg^2 n)$ | $\mathcal{O}(\lg^2 n)$ | $\mathcal{O}(n \lg n)$ |
| Blanton & Yuan | $\mathcal{O}(\sqrt{n})$ | $\mathcal{O}(\lg n)$ | $\mathcal{O}(n)$ |
| PRAC (Basic) | $\mathcal{O}(\lg^2 n)$ | $\mathcal{O}(\lg n)$ | $\mathcal{O}(n \lg n)$ |
| PRAC (Optimized) | $\mathcal{O}(\lg n)$ | $\mathcal{O}(\lg n)$ | $\mathcal{O}(n)$ |

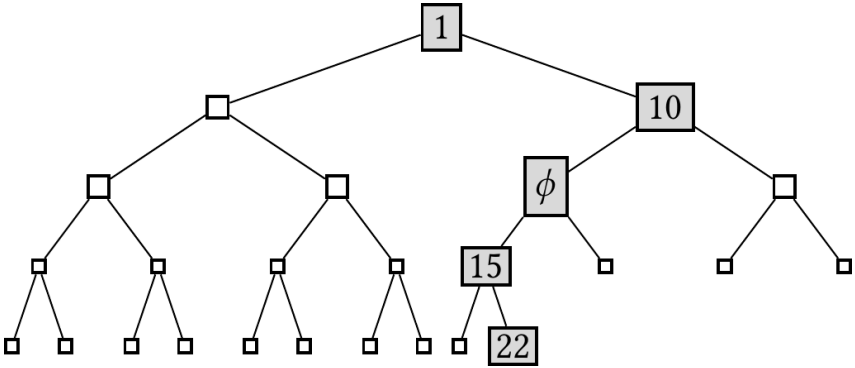
# Binary search



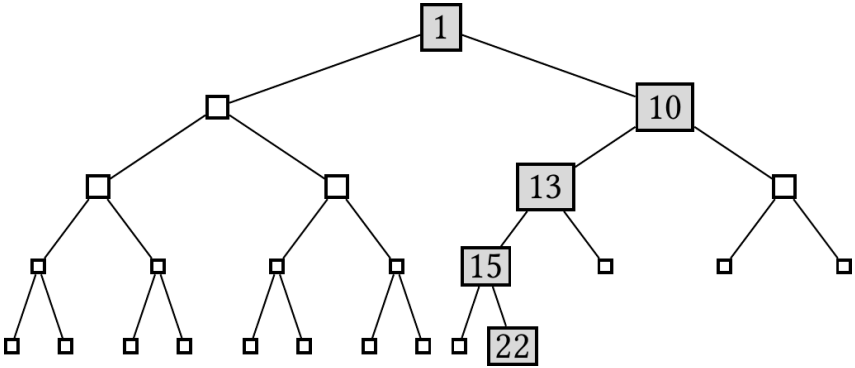
# Heaps: insertion



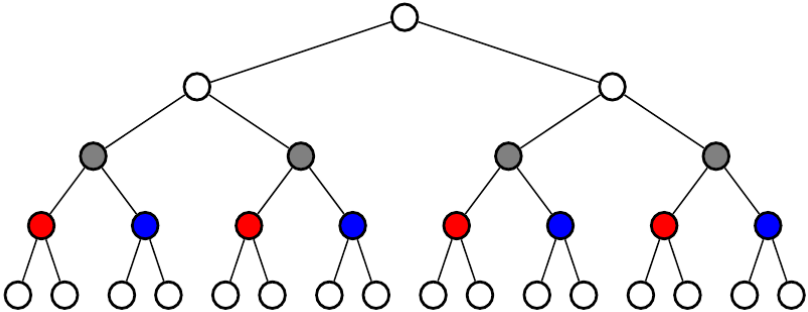
# Heaps: insertion



# Heaps: insertion



# Heaps: extraction



# Heaps

Analytical comparison of MPC Heap protocols

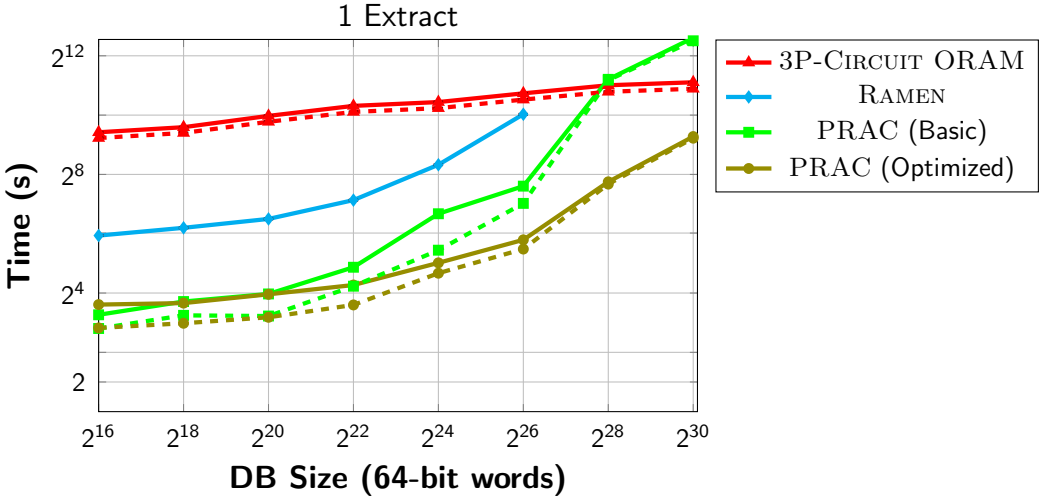
| **Protocol** | **Insert** | | |
|---|---|---|---|
| | **Communication** | | **Computation** |
| | **Bandwidth** | **Rounds** | |
| Mazloom et al. | $\mathcal{O}(\lg^2 n)$ | $\mathcal{O}(\lg^2 n)$ | $\mathcal{O}(\lg^2 n)$ |
| PRAC (Basic) | $\mathcal{O}(\lg n)$ | $\mathcal{O}(\lg n)$ | $\mathcal{O}(\lg n)$ |
| PRAC (Optimized) | $\mathcal{O}(\lg n)$ | $\mathcal{O}(\lg \lg n)$ | $\mathcal{O}(\lg n)$ |

# Heaps

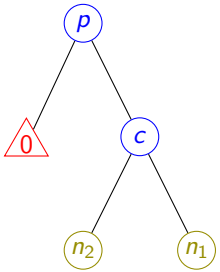
Analytical comparison of MPC Heap protocols

| **Protocol** | **Extract** | | |
|---|---|---|---|
| | **Communication** | | **Computation** |
| | **Bandwidth** | **Rounds** | |
| Mazloom et al. | $\mathcal{O}(1)$ | $\mathcal{O}(1)$ | $\mathcal{O}(n)$ |
| PRAC (Basic) | $\mathcal{O}(\lg^2 n)$ | $\mathcal{O}(\lg n)$ | $\mathcal{O}(n \lg n)$ |
| PRAC (Optimized) | $\mathcal{O}(\lg n)$ | $\mathcal{O}(\lg n)$ | $\mathcal{O}(n)$ |

# Heaps


1 Extract
Time (s)
$2^{12}$
$2^{8}$
$2^{4}$
2
$2^{16}$
$2^{18}$
$2^{20}$
$2^{22}$
$2^{24}$
$2^{26}$
$2^{28}$
$2^{30}$
DB Size (64-bit words)
3P-Circuit ORAM
Ramen
PRAC (Basic)
PRAC (Optimized)

# AVL Trees

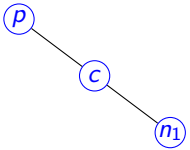



Case 1: $n_1$ inserted
Case 2: $n_2$ inserted

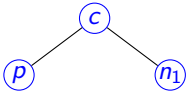
Case 1: L(R) Rotation

$F_{dr} = d_{pc} \oplus d_{cn} = 0$

After first ORotate:



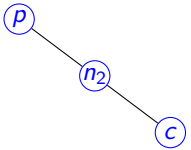

After second ORotate:



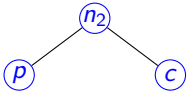

Case 2: RL(LR) Rotation

$F_{dr} = d_{pc} \oplus d_{cn} = 1$

After first ORotate:




After second ORotate:



# Takeaways

- Generally for MPC in reasonable deployments, minimizing communication (espeically rounds) is much more important than minimizing (parallelizable) computation
- PRAC provides communication-optimized MPC framework for computations requiring random-access memory and dynamic data structures, including:
  - oblivious binary search (improved over prior work)
  - oblivious heaps (improved over prior work)
  - oblivious AVL trees (for the first time!)