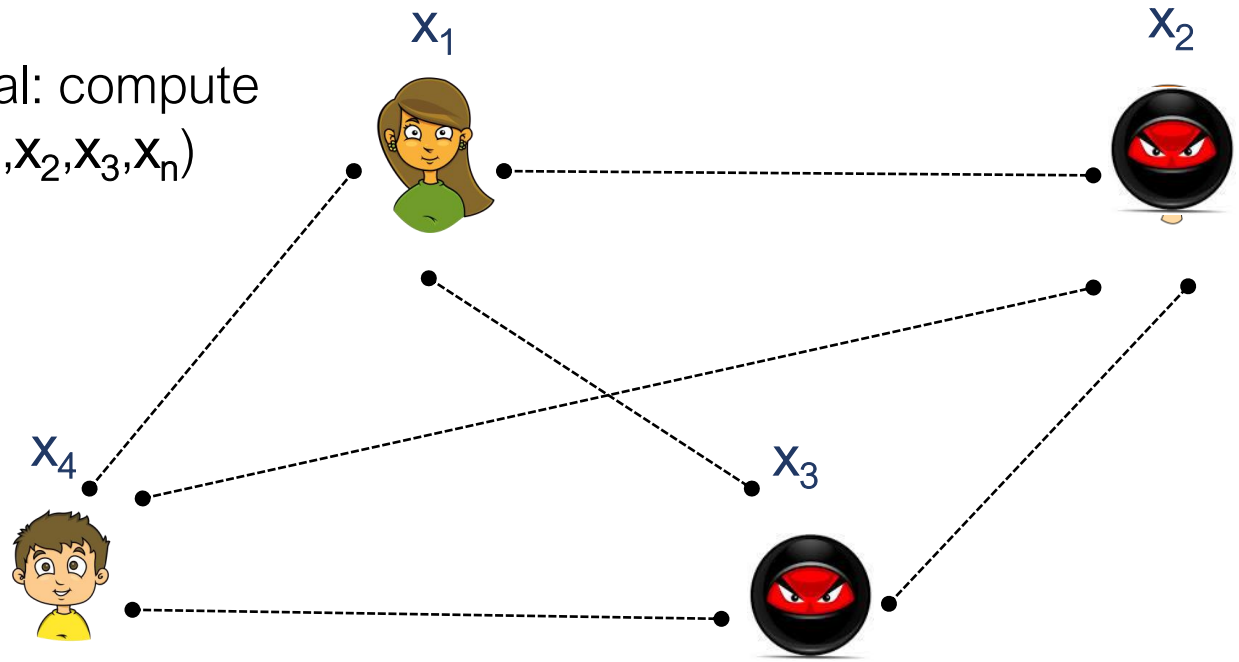
Gradual GRAM and Secure Computation for RAM Programs

Carmit Hazay
Mor Lilintal
Bar-Ilan University

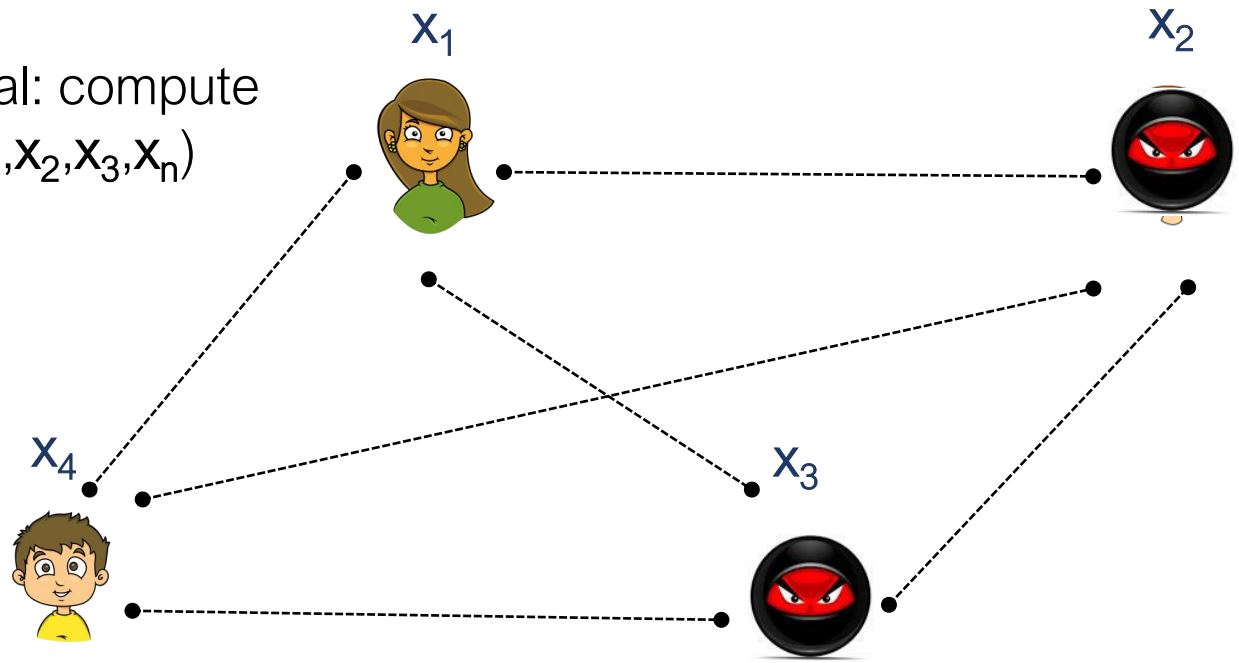
Secure Multi-Party Computation (MPC)

Goal: compute $f(x_1,x_2,x_3,x_n)$



Secure Multi-Party Computation (MPC)

Goal: compute
$f(x_1,x_2,x_3,x_n)$

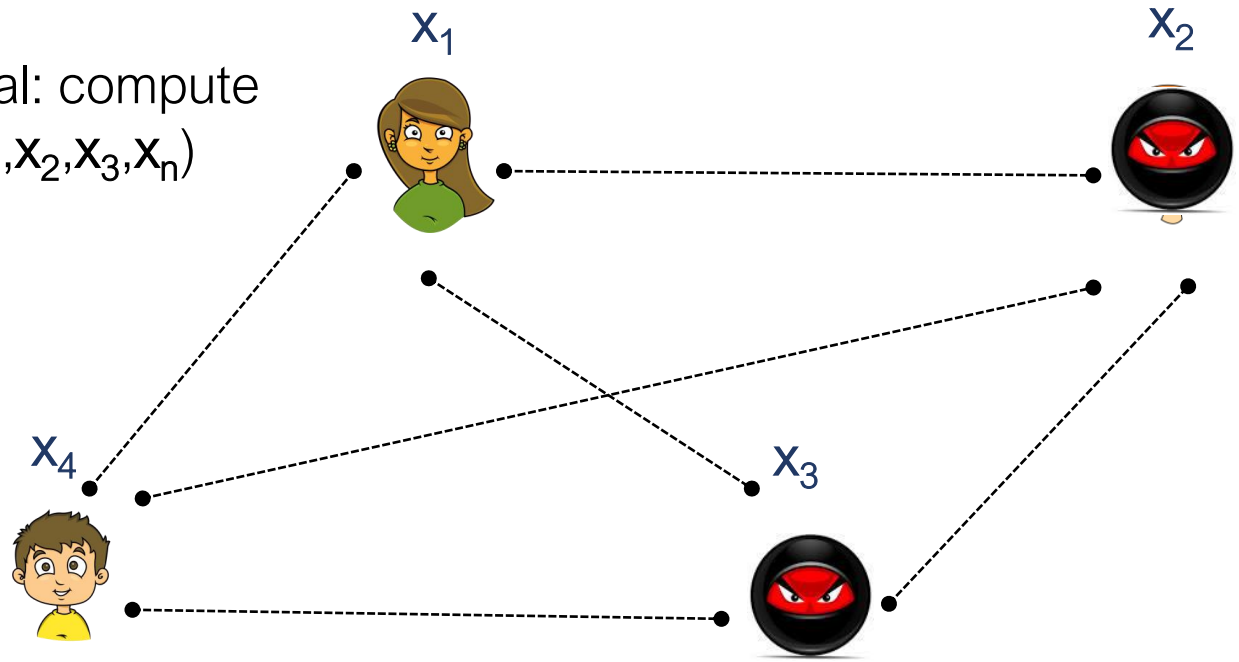


Passive vs. Active

Honest majority vs. Dishonest majority

Secure Multi-Party Computation (MPC)

Goal: compute $f(x_1,x_2,x_3,x_n)$



MPC is useful for many applications

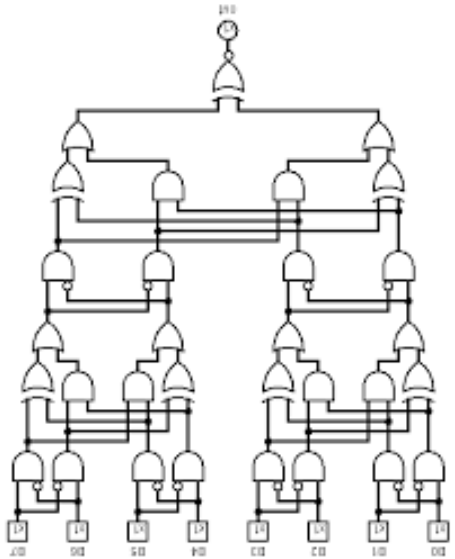
- Auctions with private bids
- Privacy-preserving data mining
- Private health records
- Cryptographic key protection
- Secure statistical analyses
- Smart city research – gender inequity
- Private blockchains
- …

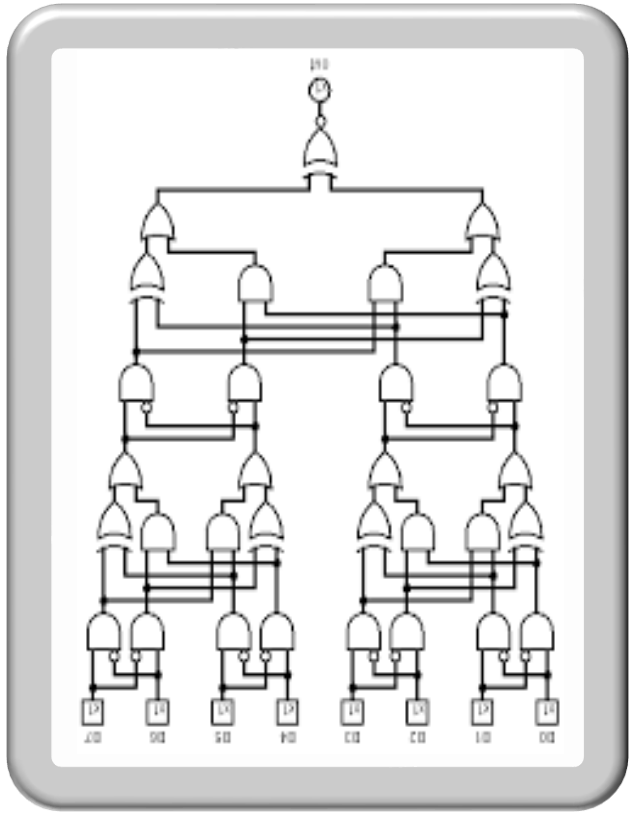
Passive vs. Active

Honest majority vs. Dishonest majority

Secure Two-Party Computation

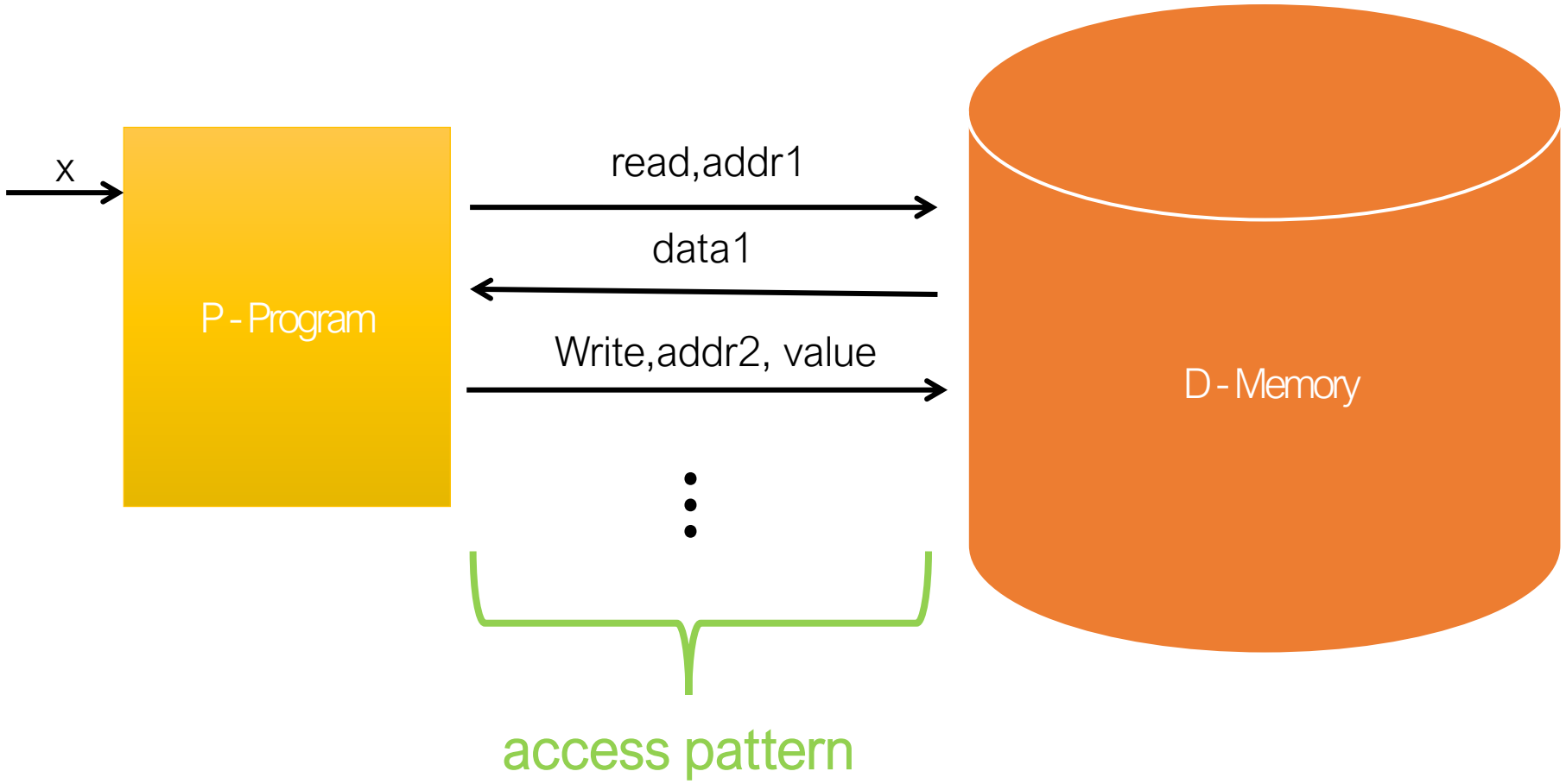
Sender



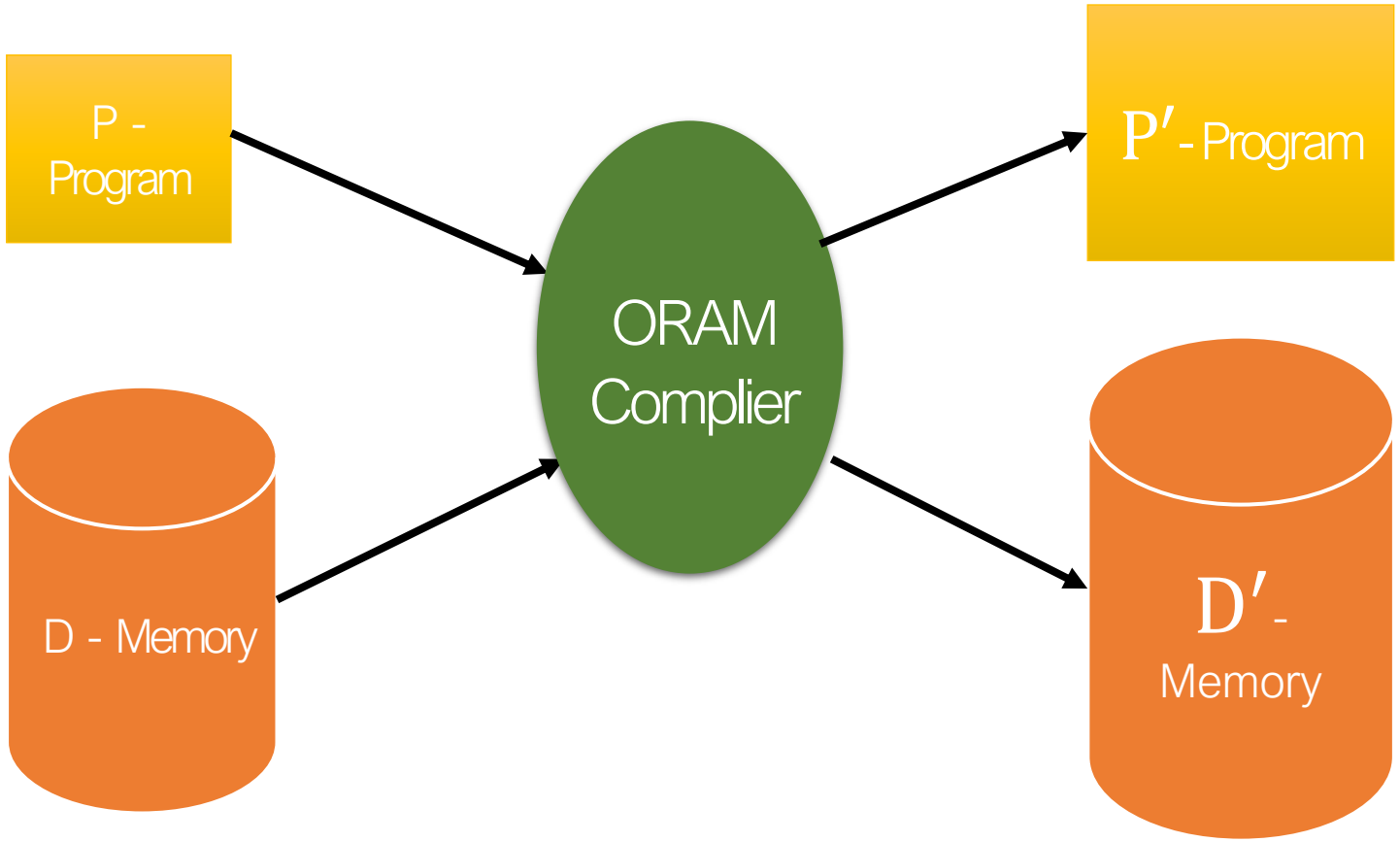


Receiver

The RAM Model



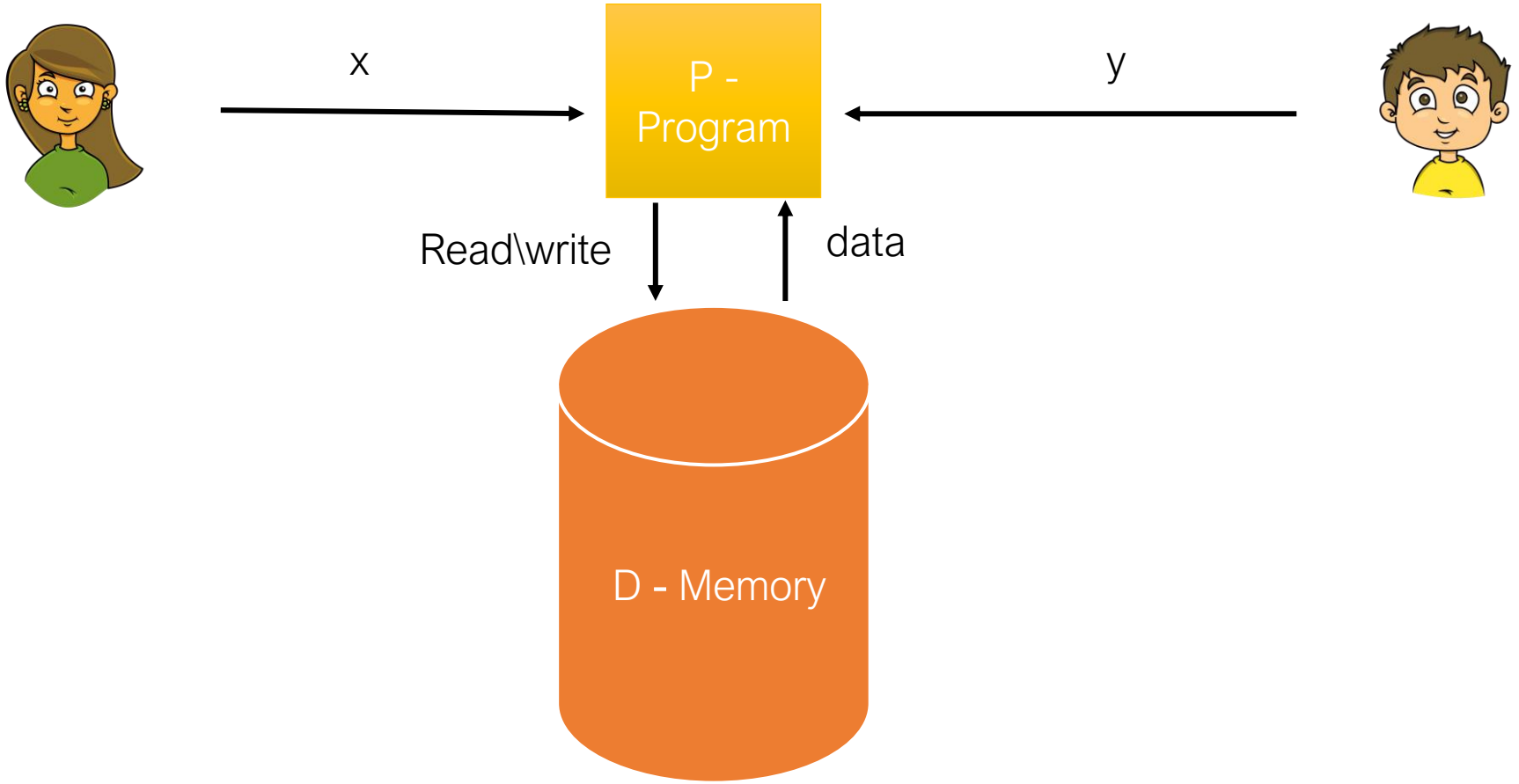
Oblivious RAM [Gol87,Ost90,GO96]



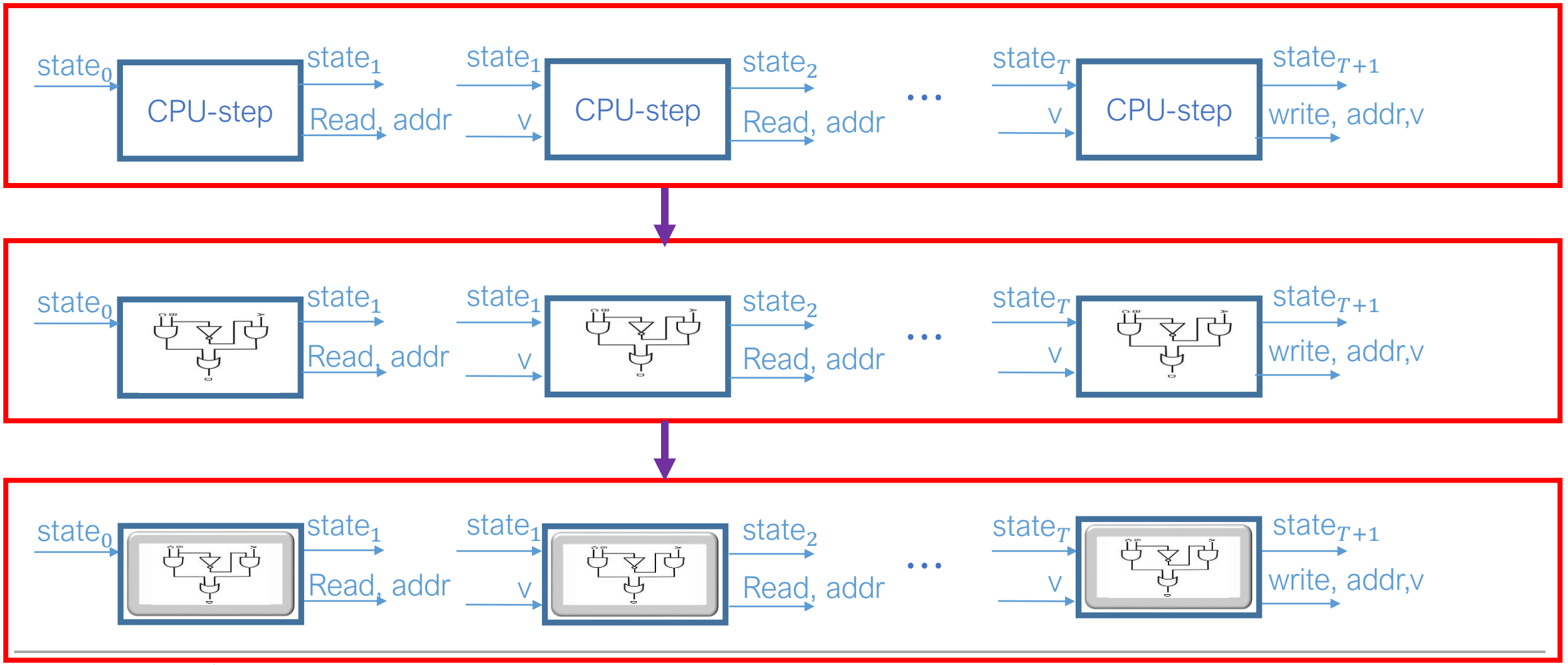
Oblivious RAM Schemes

Scheme	Block size	Amortized overhead	Worst case overhead
Trivial ORAM	$\Omega(1)$	$O(N)$	$O(N)$
Square Root [GO96]	$\Omega(1)$	$O(\sqrt{N}\log N)$	$O(N\log N)$
Hierarchical ORAM [GO96]	$\Omega(1)$	$O(\log^3 N)$	$O(N\log^2 N)$
Tree ORAM [SCSL11] **Simple ORAM** [CP13]	$\Omega(\log N)$	$O(\log^3 N)$	$O(\log^3 N)$
Circuit ORAM [WCS15]	$\Omega(\log^2 N)$	$O(\log N)$	$O(\log N)$
OptORAMa [AKL+18]	$\Omega(\log N)$	$O(\log N)$	$O(N)$

Secure Computation for RAM Programs



P as a Sequence of CPU-Steps



Prior Work

Non-constant round:

- [DMN11, GKK+12, KS14, DS17, KY18]

Constant round:

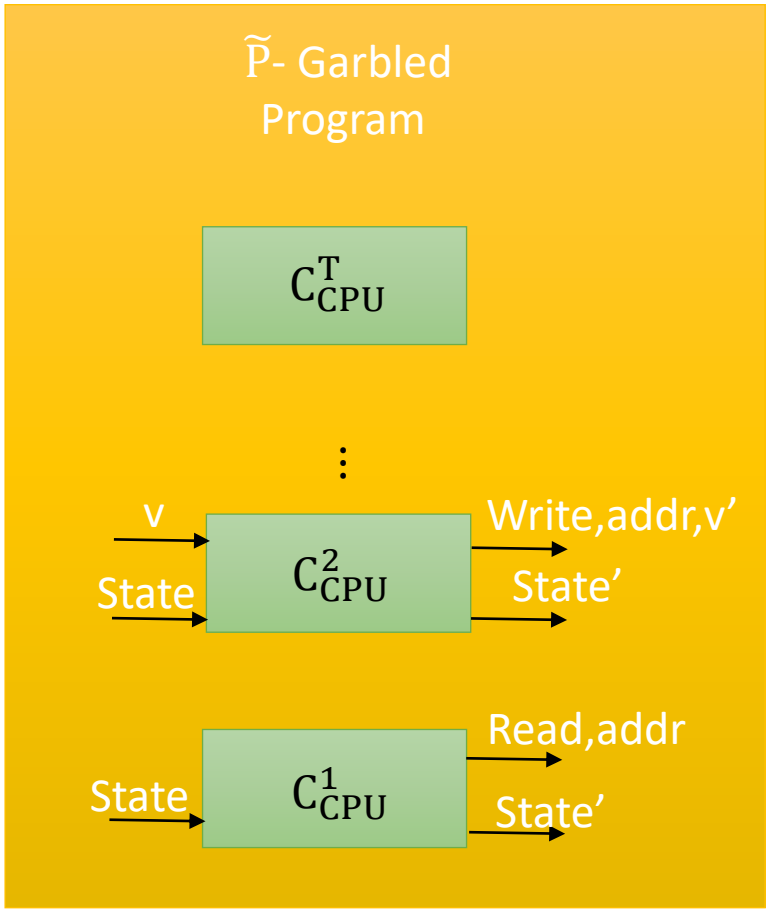
- 2PC – semi-honest [LO13,GHL+14,GLOS15,GLO15]
- 2PC – malicious [HY16,Mia16]
- MPC - malicious [GGMP16]

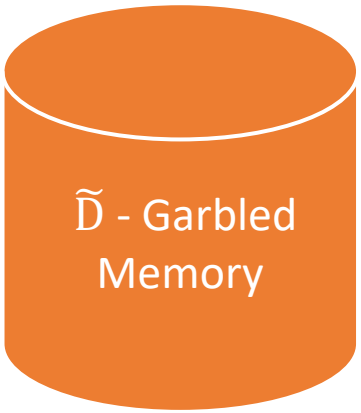
Drawbacks of Prior Work

- Memory is empty, thus no reason to initially run the garbled memory algorithm
- Passive-to-active amplification is complicated
- Security analysis in two phases via UMA

Garbled RAM [LO13]

Sender





Receiver

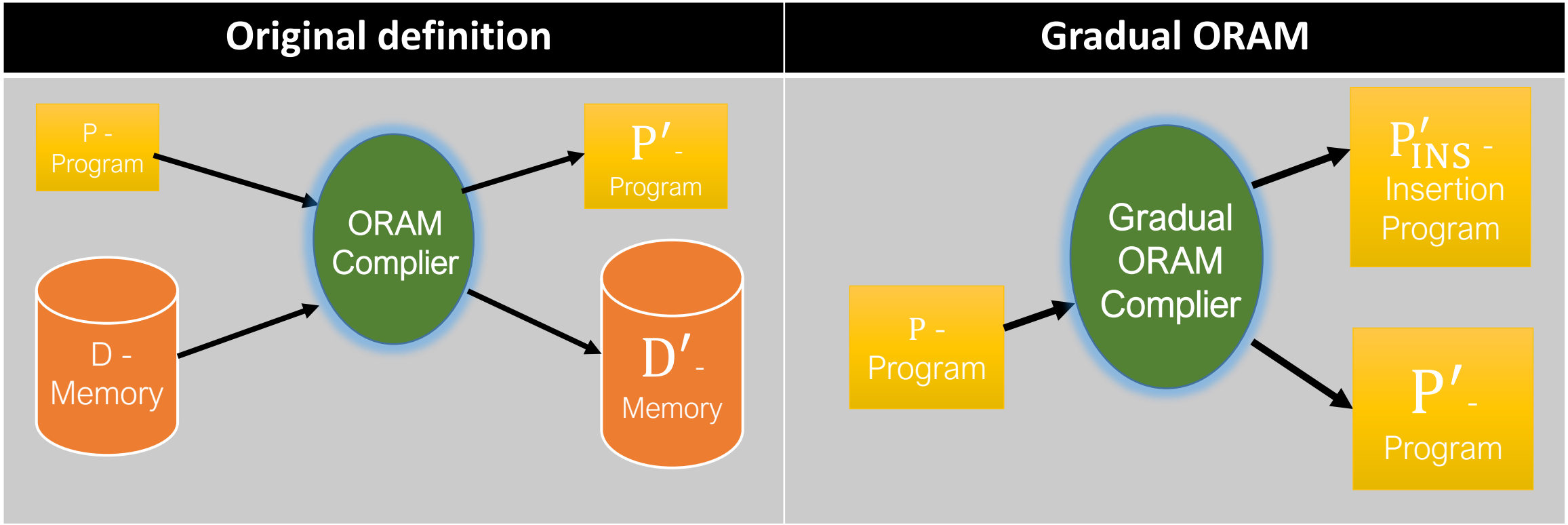
- **Correctness.** Pr $[\mathrm{GEval}^{\widetilde{D}}(\widetilde{P},\tilde{x}) = P^D(x)] = 1.$
- **Security.** $(\widetilde{D},\widetilde{P},\tilde{x}) \cong \mathrm{SIM}(N,T,P,y).$

Our Results

1. **Gradual ORAM for tree-based ORAMs**
    - Update on the fly
    - Instantiated with Simple ORAM [CP13] and Circuit ORAM [WCS15]

Gradual ORAM

- **No initialization** phase
- Memory is **gradually** initialized



Our Results

1. **Gradual ORAM for tree-based ORAMs**
   - Update on the fly
   - Instantiated with Simple ORAM [CP13] and Circuit ORAM [WCS15]
2. **Gradual GRAM**
   - No memory garbling algorithm
   - Instantiated with [GLOS15] GRAM

Advantages of Using Gradual ORAM

1. GRAM enjoys all the advancements of garbled circuits such as half gates, Free-XOR, Pipelining, and OT extension

2. Most of work can be shifted to an offline phase

3. Amplifying security to the malicious setting is immediate by applying standard techniques such as cut-and-choose or authenticated garbling

4. The space complexity of the garbler algorithm can be made minimal

Garbled RAM [GLOS15]

The garbled program - $\tilde{\mathbf{P}}$

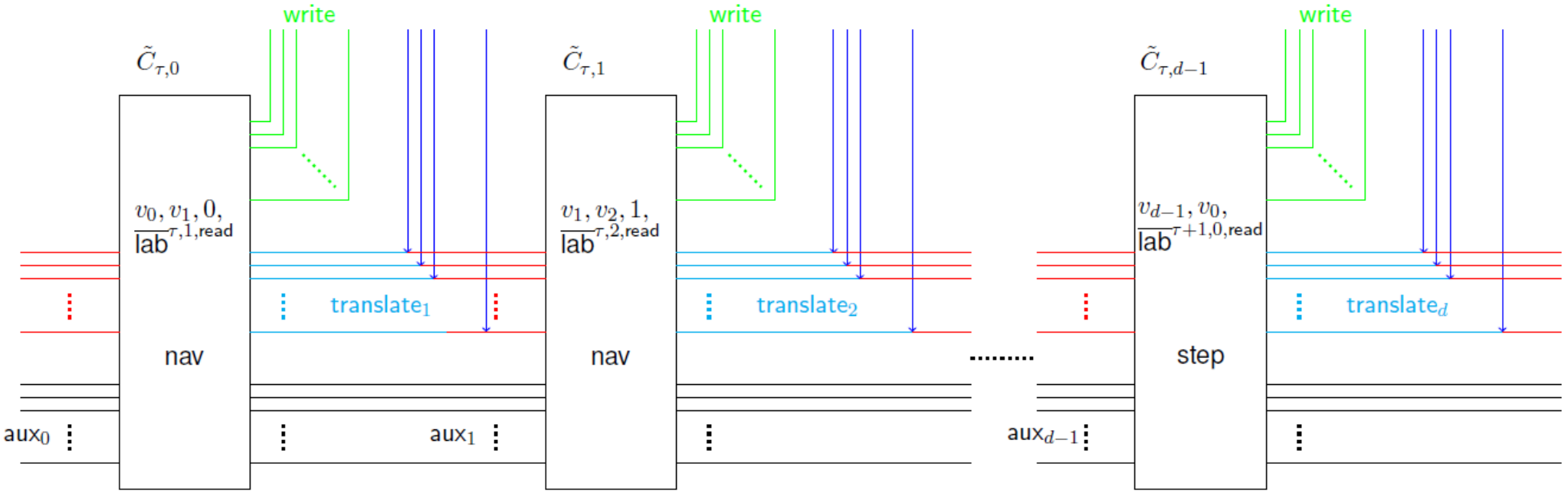
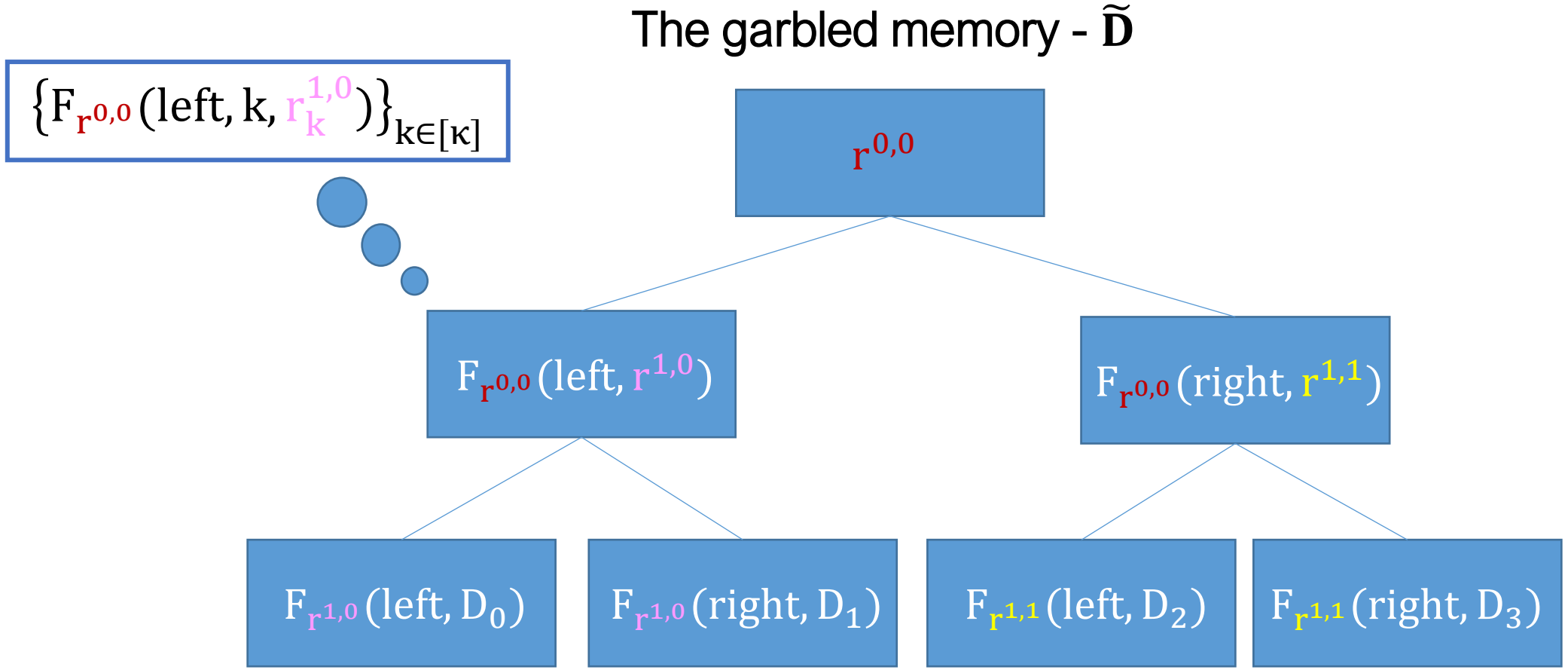
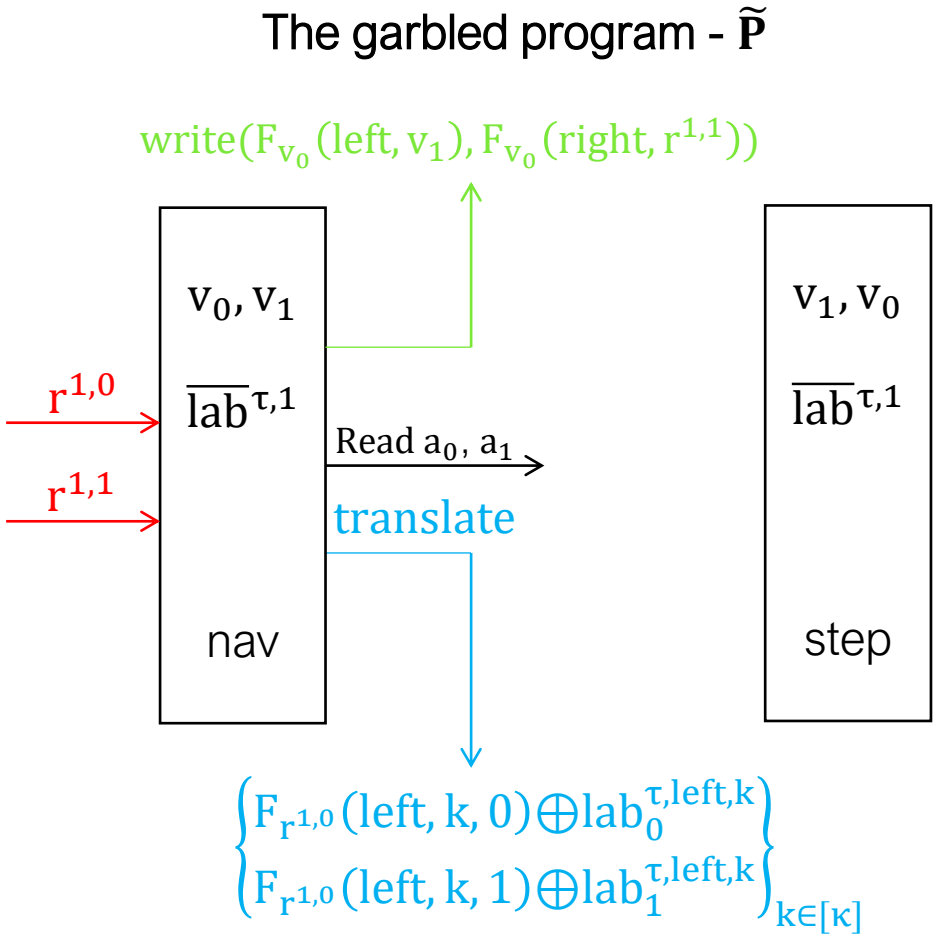


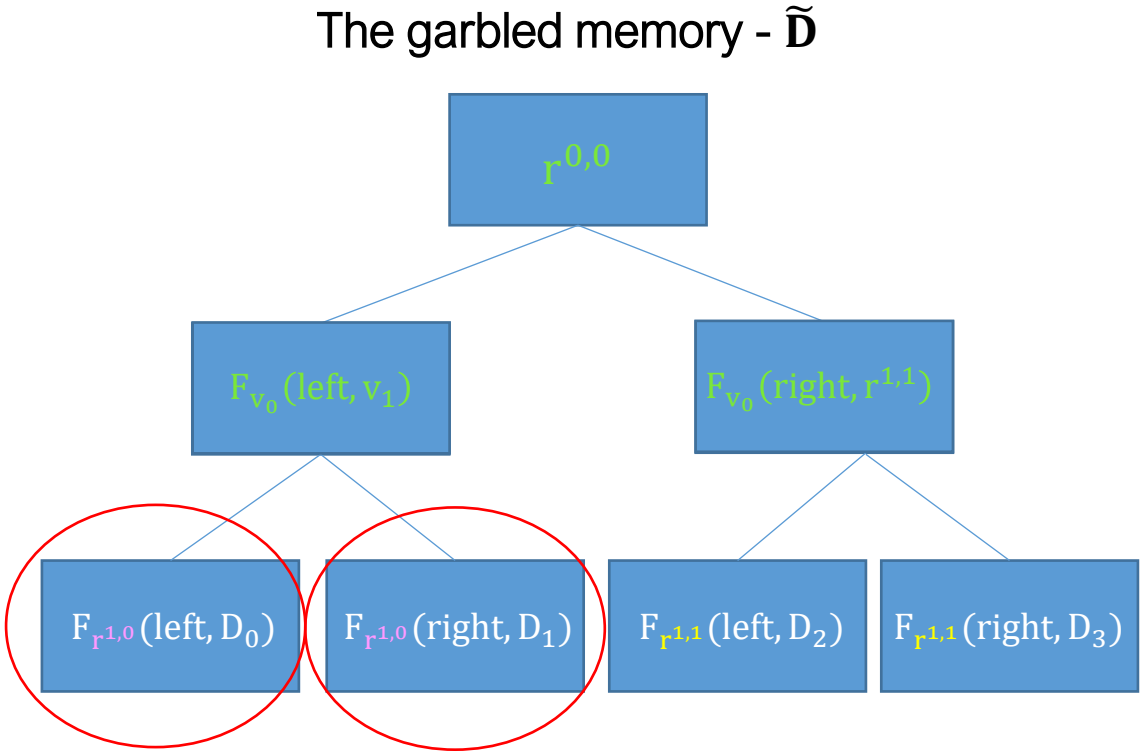
Figure 5: Visualization of one time step of the RAM Program.

Garbled RAM [GLOS15]

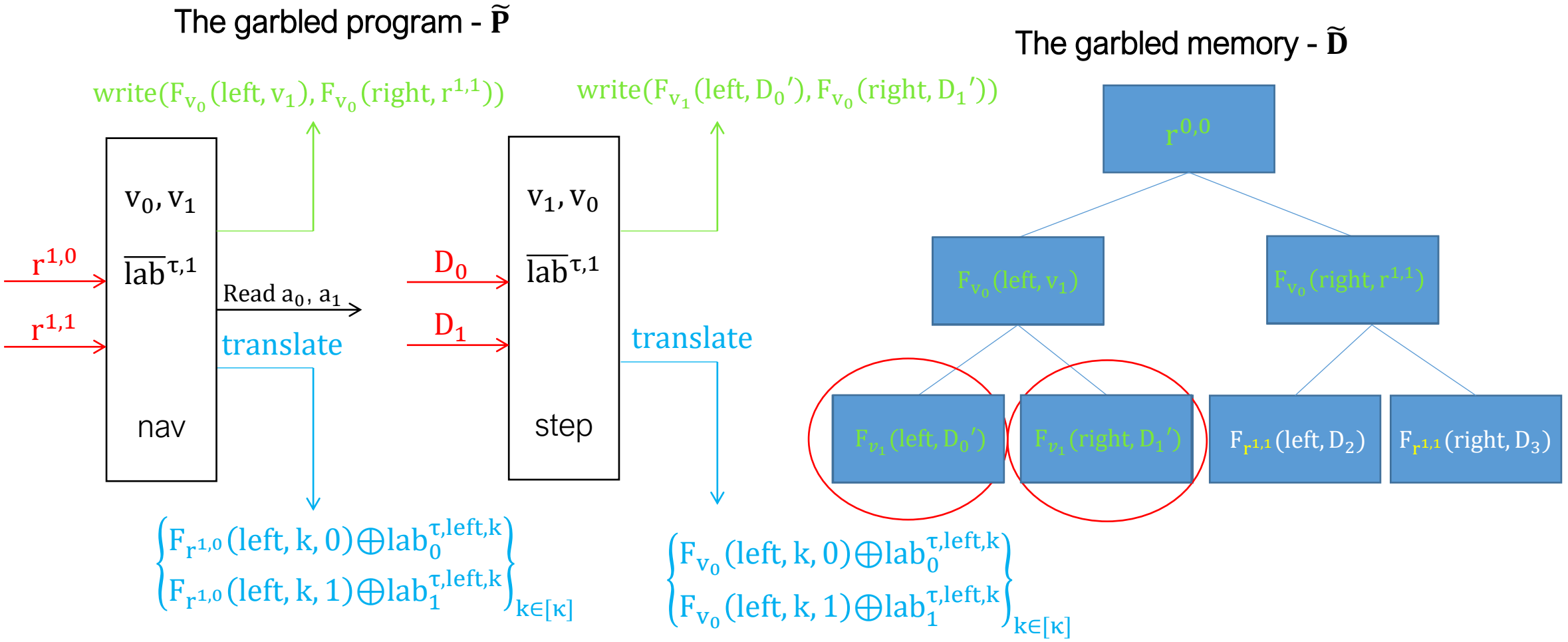


Garbled RAM [GLOS15]



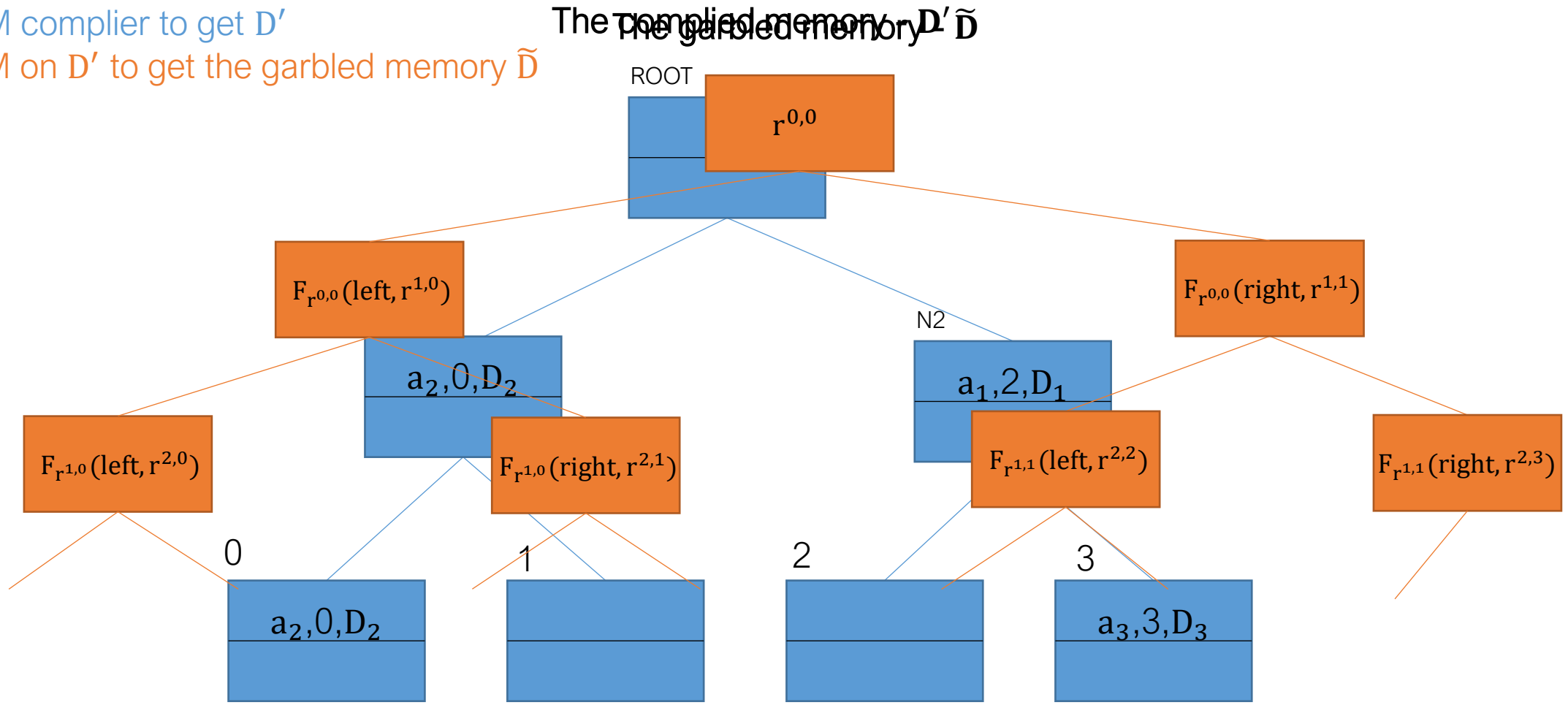


Garbled RAM [GLOS15]



Prior Work [GLOS15] – Full Security

1. Run ORAM complier to get D'
2. Run GRAM on D' to get the garbled memory $\tilde{\mathrm{D}}$

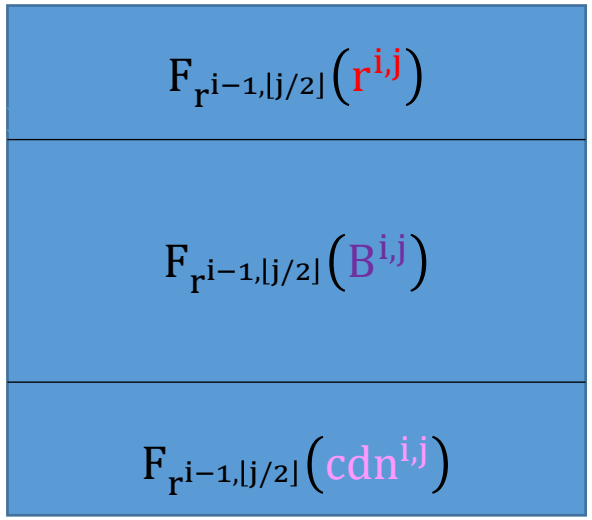


Our Work – Full Security

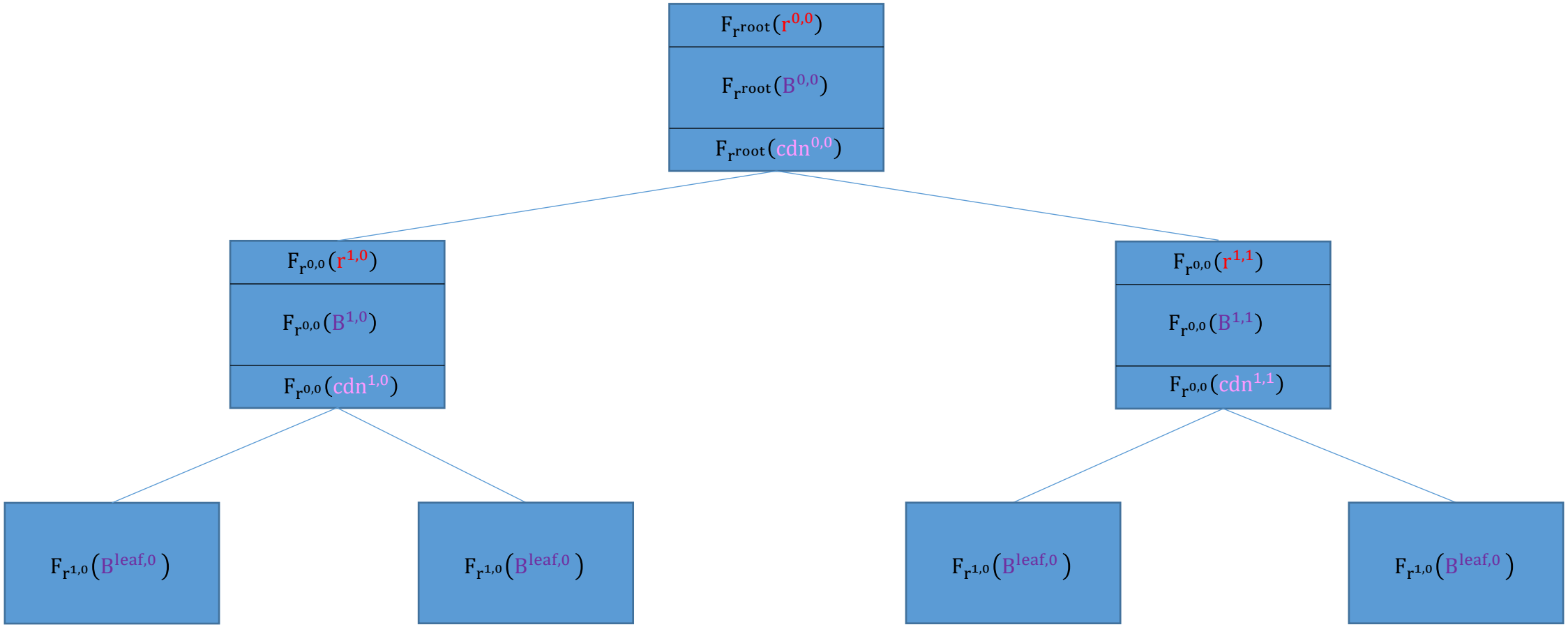
The Garbled memory (on the fly):

Each internal node contains a PRF evaluation under the parent's key of

- a key associated with this node (as in GLOS)
- a bucket with a data (as in the Simple ORAM)
- Information about the children



Our Work – Full Security



Efficiency Gains

Table 1: Communication and computation costs for the different GRAMs schemes. We use n to denote the number of memory entries and κ the security parameter.

Approach	**Overall Number of Steps**	**Overall CPU-Steps Circuits Sizes per a Single ORAM Access**
[GLOS15]	$O((n+T)\cdot \log^2 n)$	$O((\log n - 1)\cdot \kappa^2 + \kappa \cdot B)$
Gradual [GLOS15] + Simple ORAM	$O((n+T)\cdot \log n)$	$O(\kappa^2 + \kappa \cdot U \cdot \log n)$
Gradual [GLOS15] + Circuit ORAM	$O((n+T)\cdot \log n)$	$O(\kappa^2 + \kappa \cdot U)$


Thank You