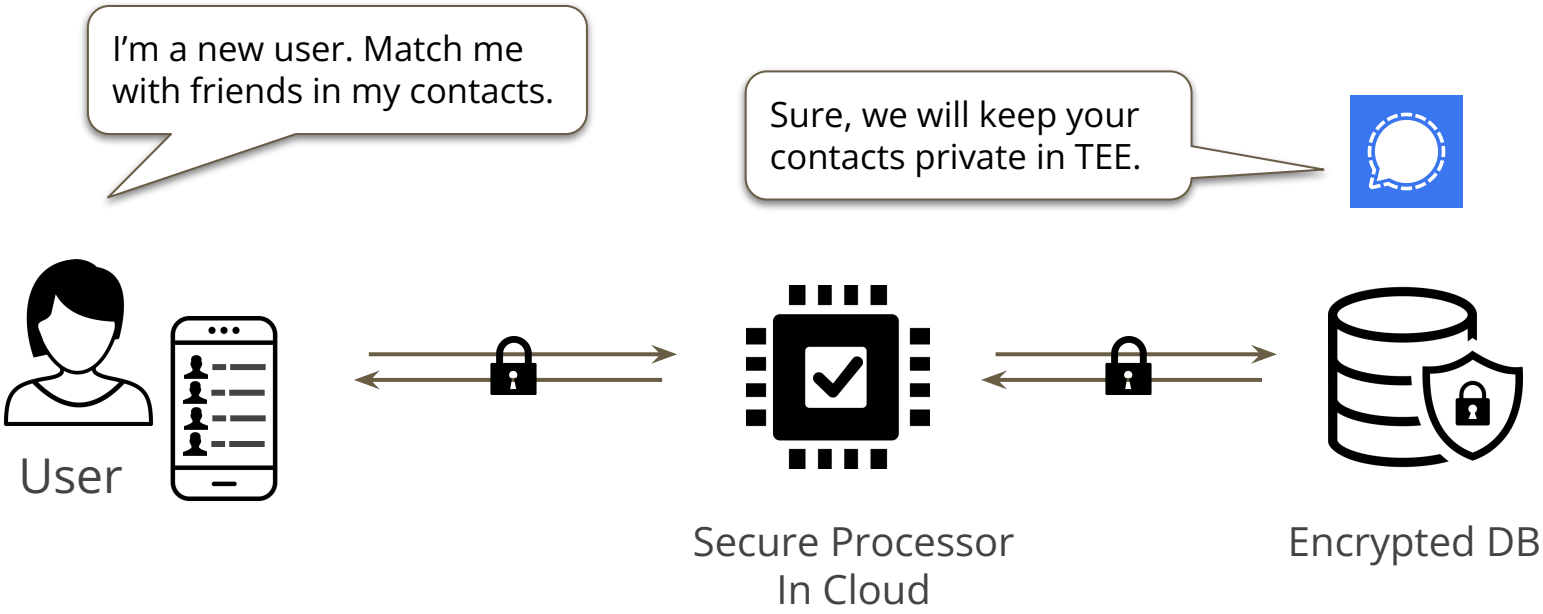
# Oblivious RAM

For Programmable Cryptography and Ethereum

Presenter: Tianyao Gu
Oblivious Labs

# A Web2 Example

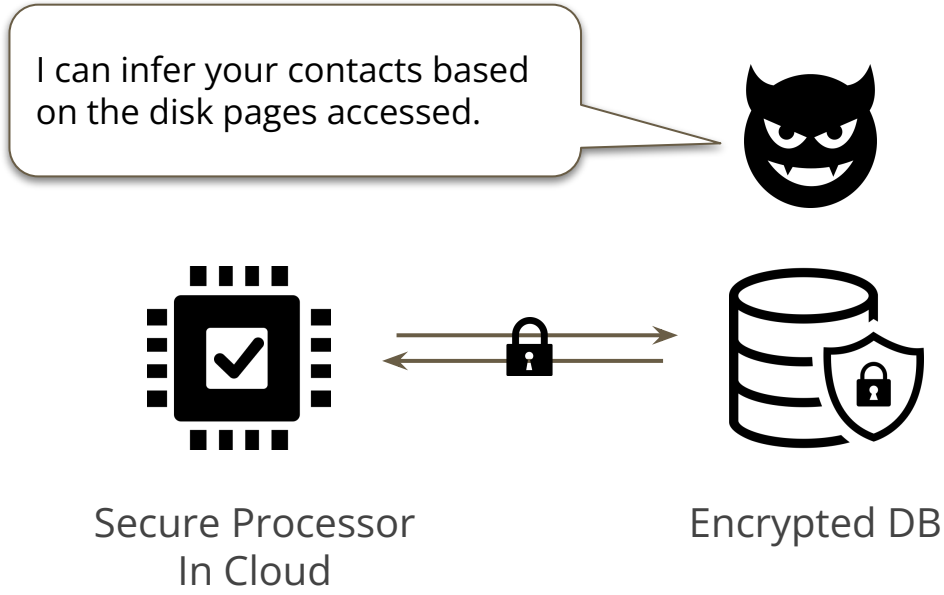
*Signal Private Messenger - Contact Discovery*



# A Web2 Example

*Signal Private Messenger - Contact Discovery*

**Encryption doesn't hide access patterns!**



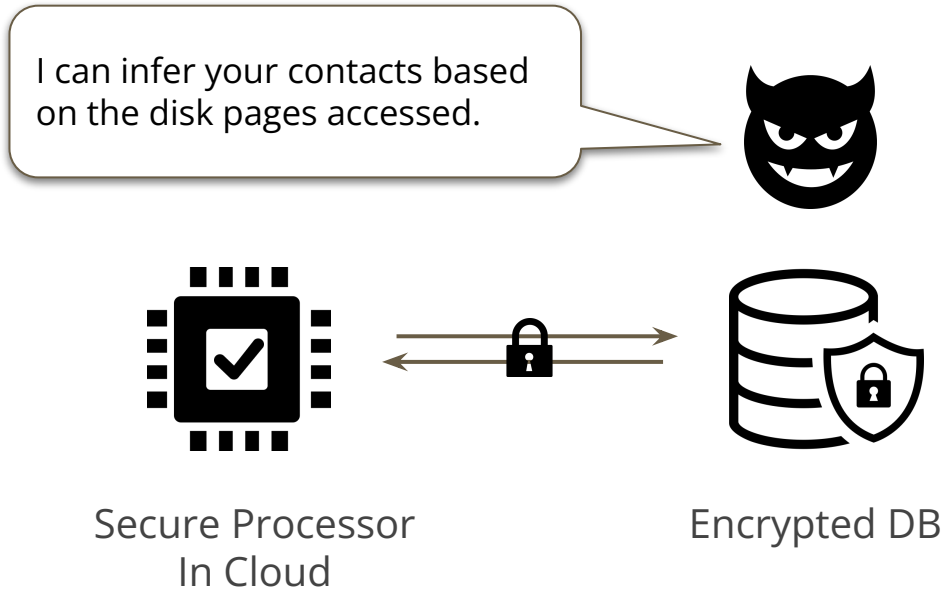
# A Web2 Example

*Signal Private Messenger - Contact Discovery*

Naive Solution

Scan the entire DB

Requires ~500 servers.



# Oblivious RAM (ORAM) to the Rescue

- First proposed by Goldreich and Ostrovsky in 1987.
- Obfuscates access patterns of the program.
- Only poly-log overhead and practically efficient.
- Does NOT need cryptography.

# Let's build an ORAM

- Attempt 1 — Place DB entries in random locations

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| DB[1] | | | | DB[2] | DB[0] | DB[3] | |

Look, this block is accessed again.
These users may share the same contact.

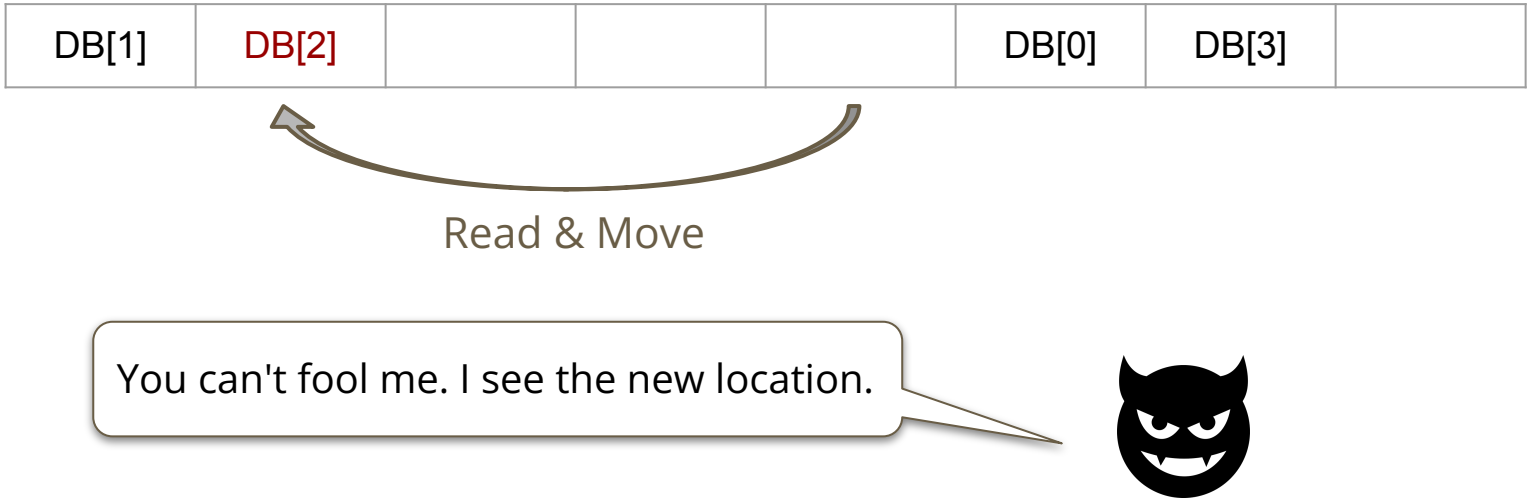
# Let's build an ORAM

- Attempt 2 — Move entry to a random new location after read

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| DB[1] | | | | DB[2] | DB[0] | DB[3] | |

# Let's build an ORAM

- Attempt 2 — Move entry to a random new location after read



# Let's build an ORAM

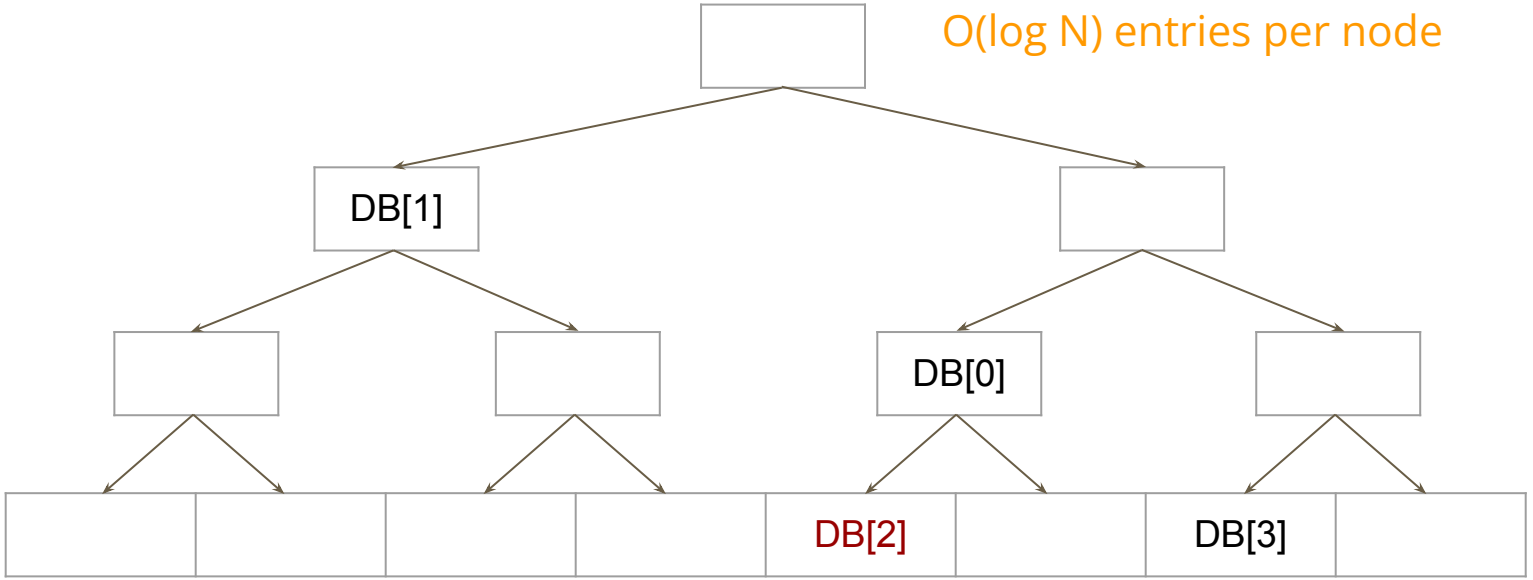
- How to hide the movement?


Point-to-point
No transit

Fixed route
& schedule

# Let's build an ORAM

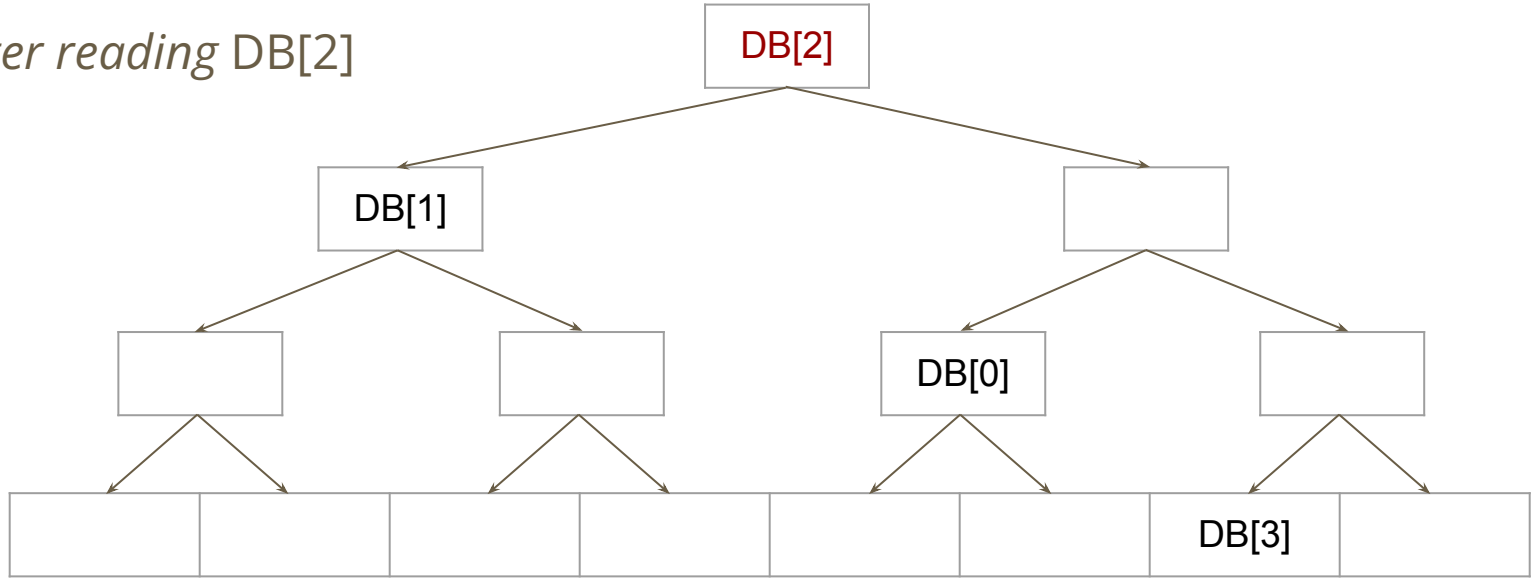
- Binary-tree ORAM [SCS+11] — Entries may reside in ancestor nodes



# Let's build an ORAM

- Write entry to the root node and maintain the tree

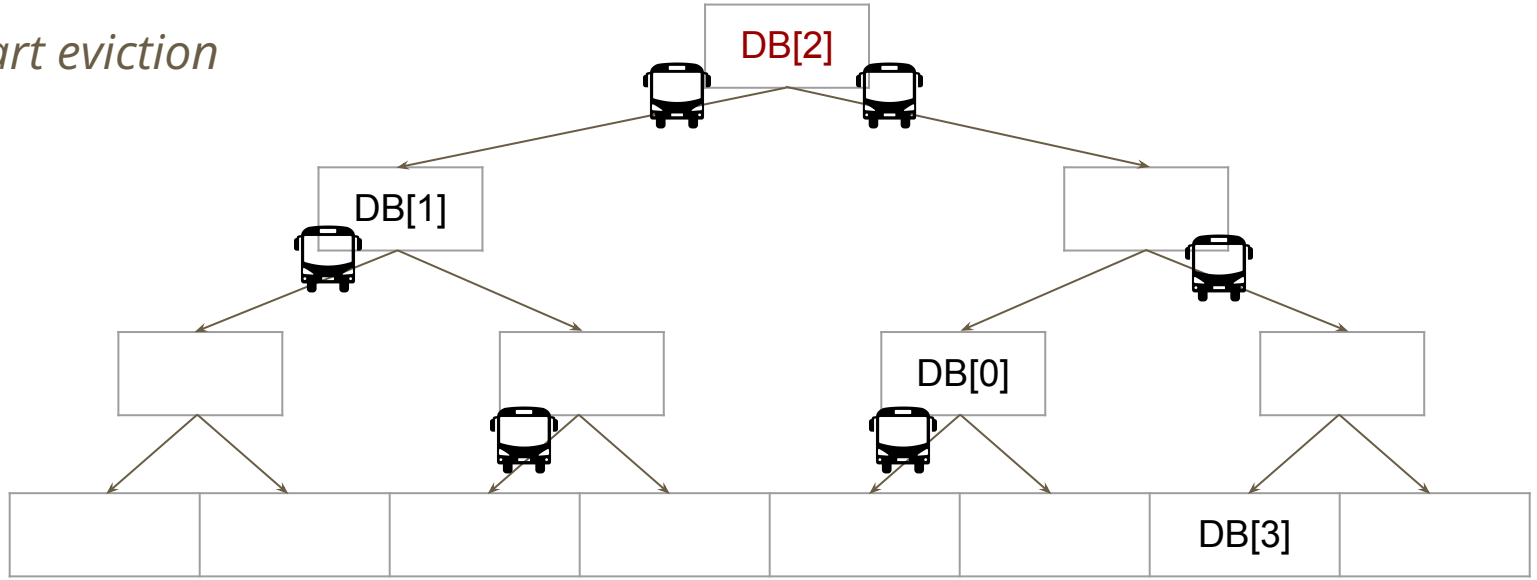
*After reading* DB[2]



# Let's build an ORAM

- Write entry to the root node and maintain the tree

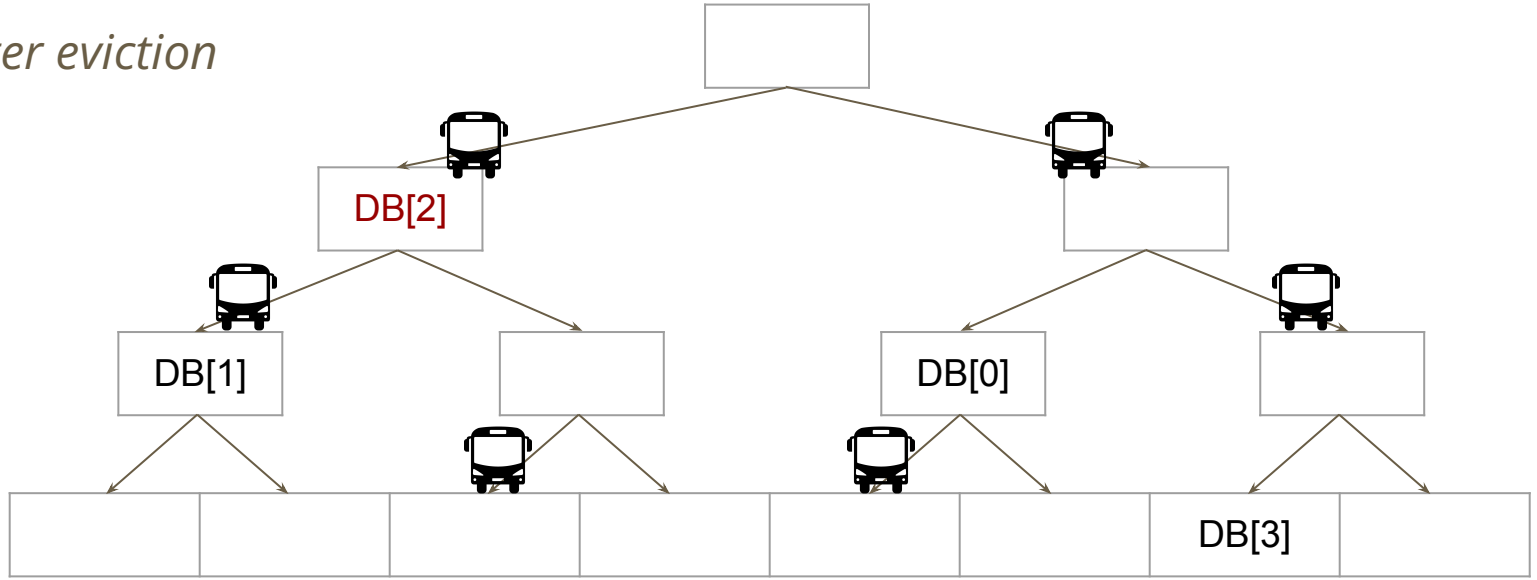
*Start eviction*



# Let's build an ORAM

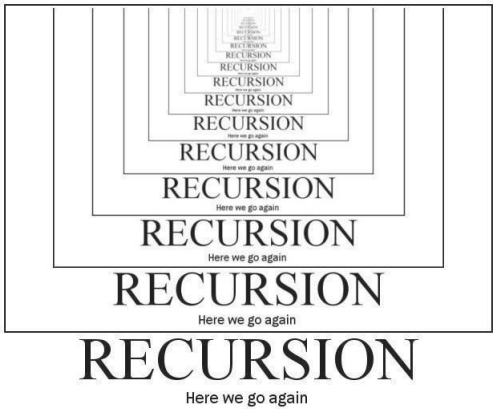
- Write entry to the root node and maintain the tree

*After eviction*



# Track entries' locations

- Store the locations with another (smaller) ORAM.
- Recurse until linear scan becomes efficient.



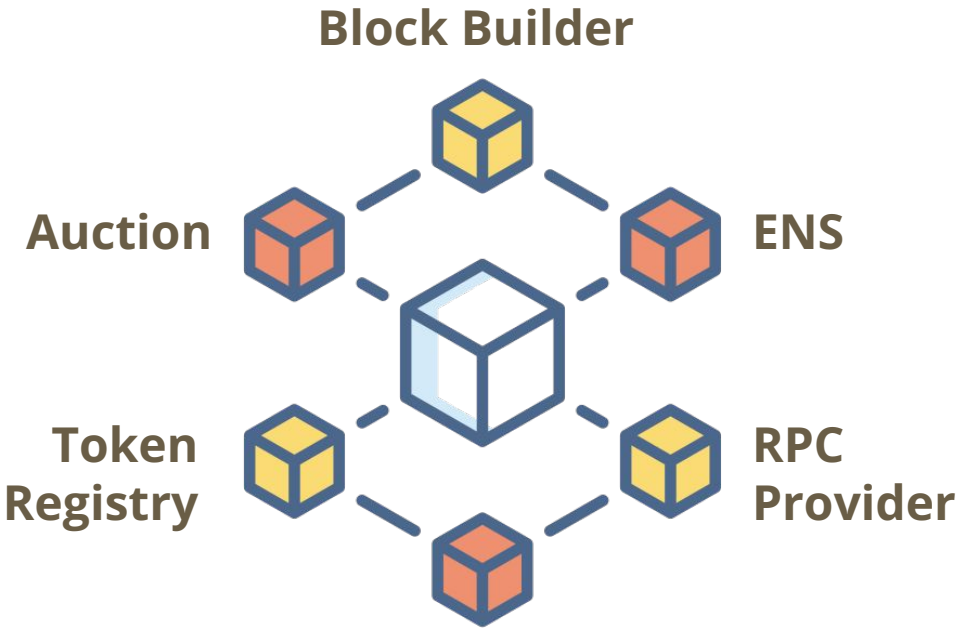
# Variants of Tree-based ORAM

- Plenty of works based this paradigm

  Path ORAM [SDS+13]
  Circuit ORAM [WCS14] → $O(\log^2 N)$ overhead
  Bucket ORAM [FNR+15]
  Onion ORAM [DDF+15]
  Ring ORAM [RFK+15]
  OnionRing ORAM [CCR19]
  rORAM [CAC+19]
  .....

- Open question: O(log N) statistically-secure ORAM?

# Enhancing Privacy in Ethereum




Blockchain icons created by Triangle Squad - Flaticon

# ORAM Meets with MPC

- Compilers / Frameworks with ORAM

  Obliv-C [ZE+13], ObliVM [LWN+15], GraphSC [NWI+15], MP-SPDZ [Keller20]

- **Garbled RAM** [ LO13, GHL+14, GLO15, LO17, HKO21, PLS22, HKO23]

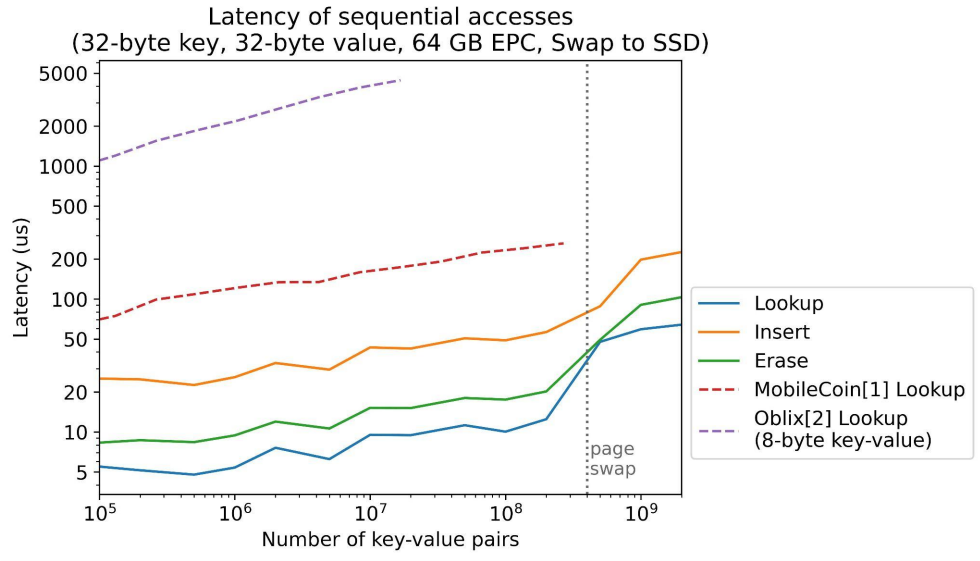
  Equip garbled circuit with ORAM => Constant-round 2PC / MPC.

- Make large private state possible for the Frog Game.

# Concrete Performance

- Intel SGX w/ 100 GB key-value database: ~ **50 us** per access

  (i.e., 20,000 iops)



# Concrete Performance

- Garbled RAM for constant-round 2PC (semi-honest)

  ~**15 MB** of communication per access

  for memory space: 2^20 words
  word width: 128 bit
  security parameter: $\lambda = 128$, $\sigma = 40$

- Linear scan costs 8GB for the same setup

# Thank you

Open-source ORAM implementation:
https://github.com/obliviouslabs/oram