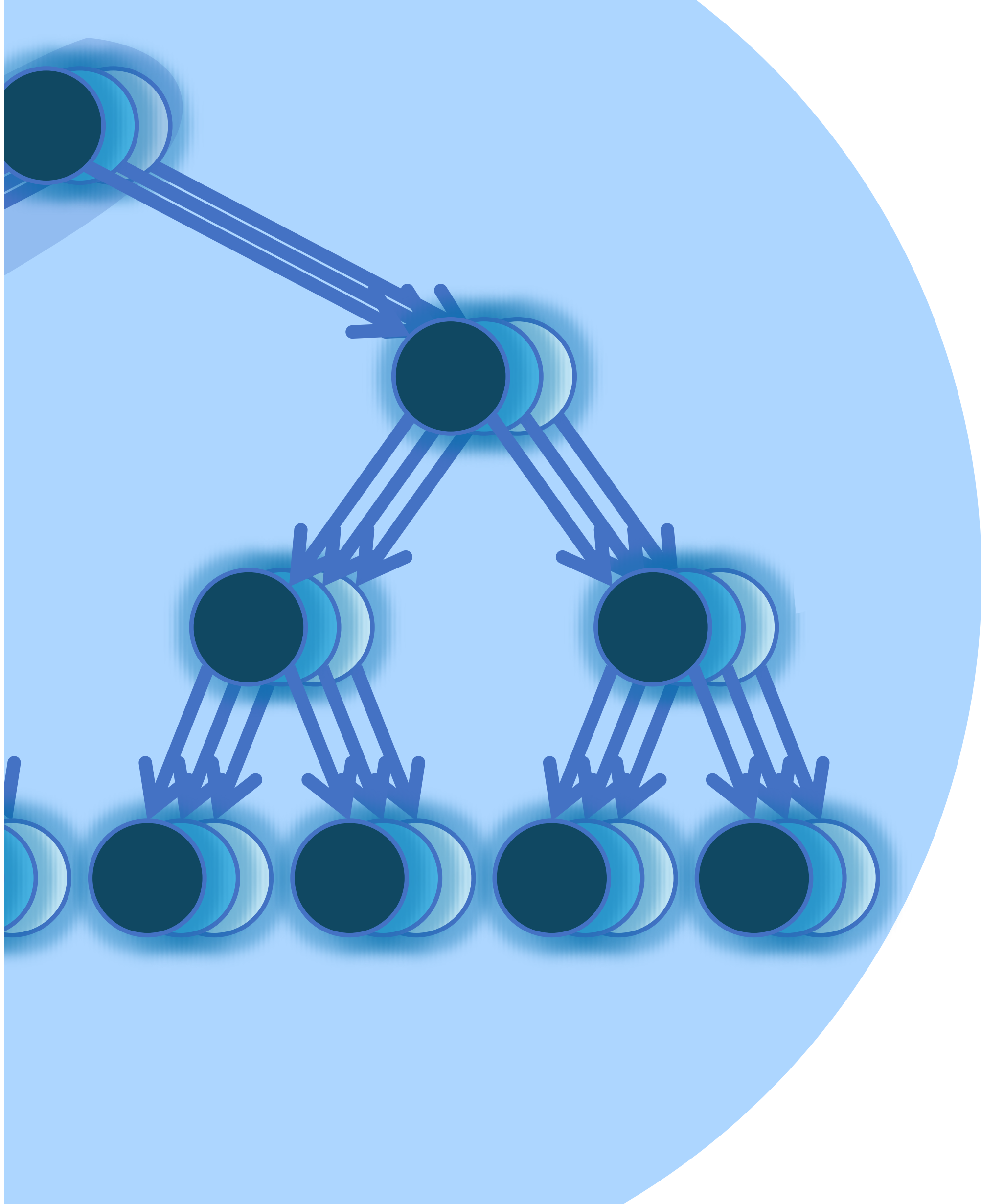


MVP-ORAM: a Wait-free Concurrent ORAM for Confidential BFT Storage

NDSS 2026

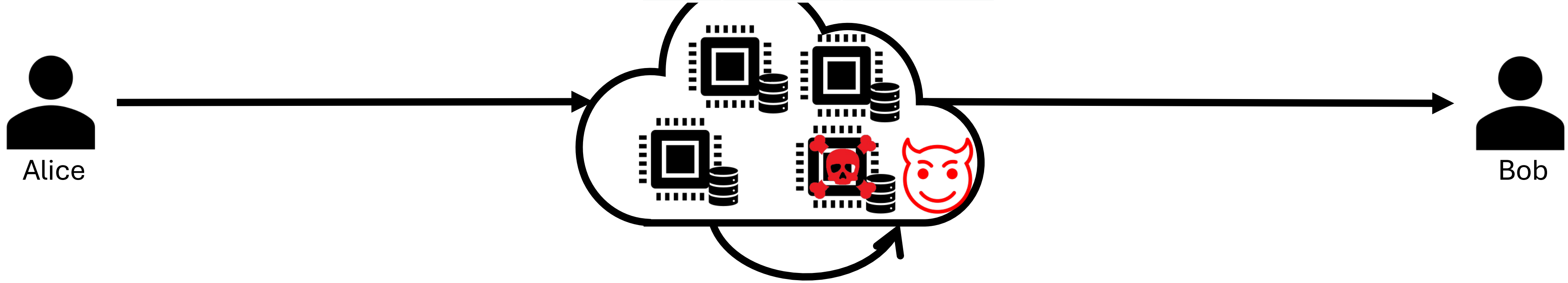
Robin Vassantlal, Hasan Heydari,

Bernardo Ferreira, Alysson Bessani

LASIGE, Faculty of Sciences, University of Lisbon, Portugal

BFT-SMR

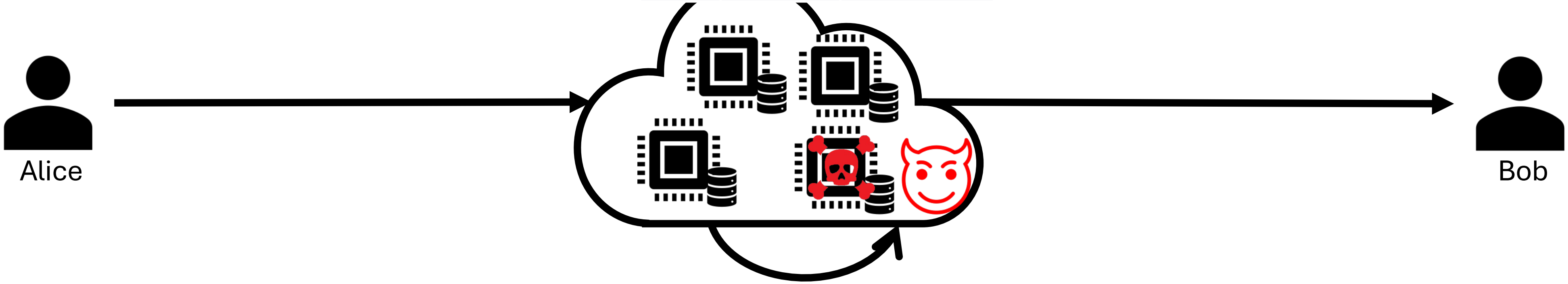
ID	Client	Balance
1	Alice	50k
2	Bob	100k



Byzantine Fault-Tolerant State Machine Replication (BFT-SMR) is a classical technique used for implementing **fault-** and **intrusion-tolerant services**

BFT-SMR

ID	Client	Balance
1	Alice	50k
2	Bob	100k



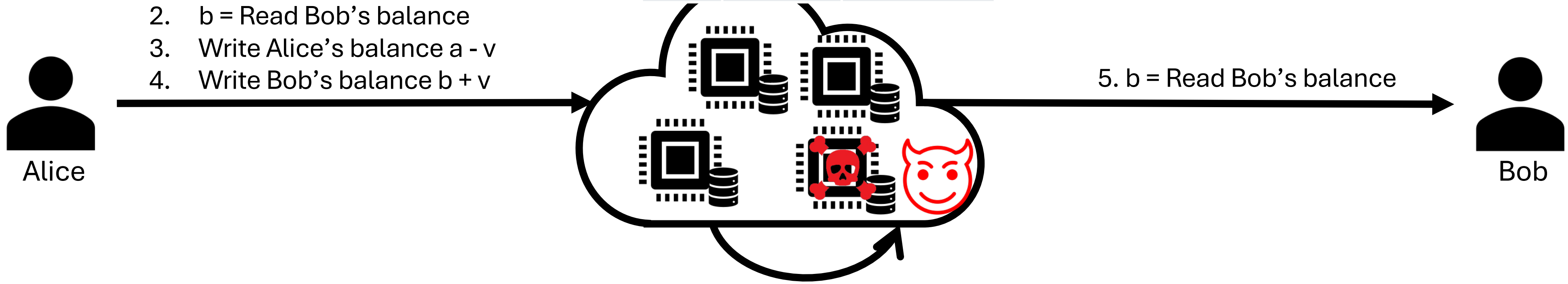
Byzantine Fault-Tolerant State Machine Replication (BFT-SMR) is a classical technique used for implementing **fault-** and **intrusion-tolerant services**

Ensures **availability** and **integrity**

Useful for implementing **blockchains**

Confidential BFT-SMR

ID	Client	Balance
1	Alice	@$%&
2	Bob	#@&#

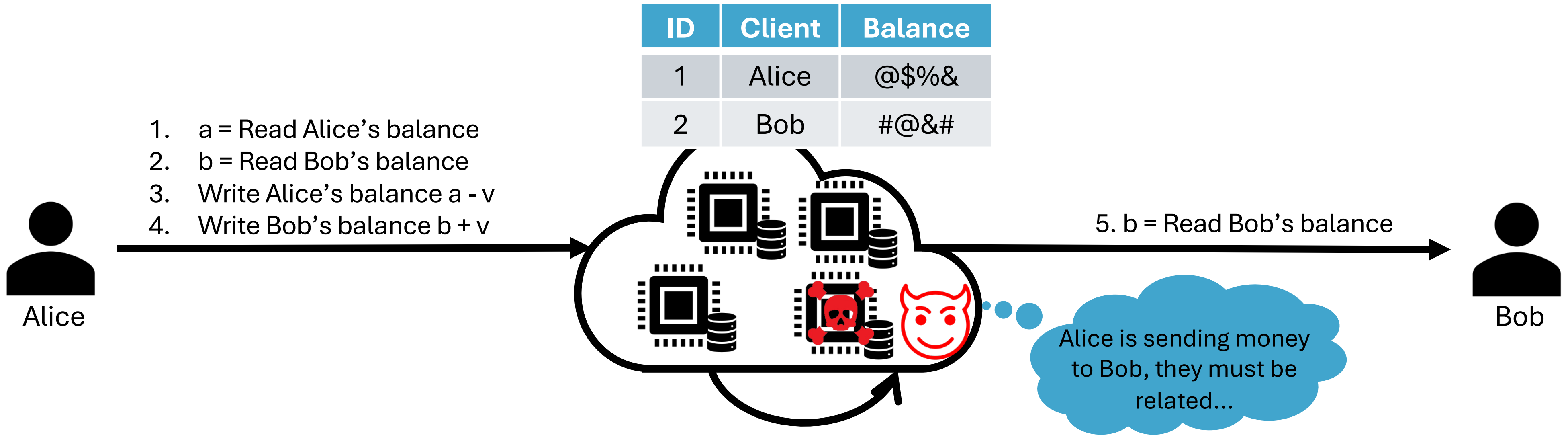


1. a = Read Alice’s balance
2. b = Read Bob’s balance
3. Write Alice’s balance a - v
4. Write Bob’s balance b + v

5. b = Read Bob’s balance

Confidential BFT-SMR additionally ensures **confidentiality**

Confidential BFT-SMR



ID	Client	Balance
1	Alice	@$%&
2	Bob	#@&#

1. a = Read Alice’s balance
2. b = Read Bob’s balance
3. Write Alice’s balance a - v
4. Write Bob’s balance b + v

5. b = Read Bob’s balance

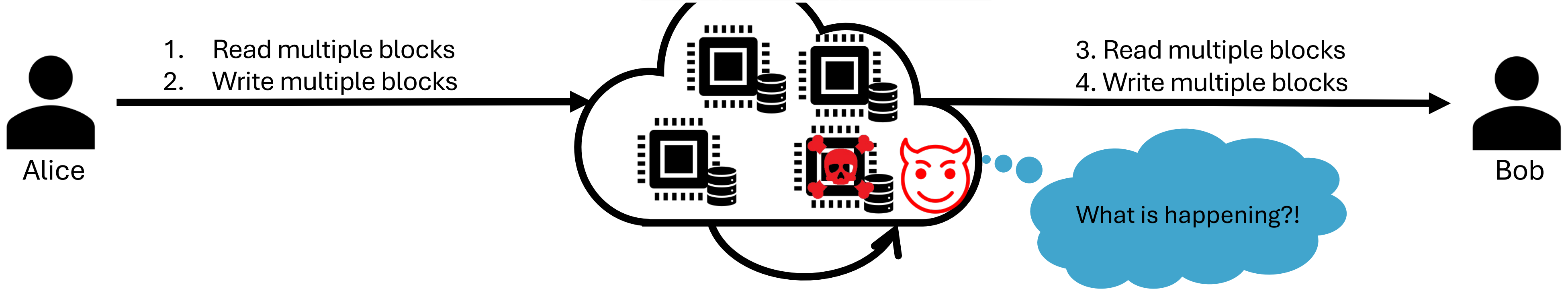
Alice is sending money to Bob, they must be related…

Confidential BFT-SMR additionally ensures **confidentiality**

However, even if data is encrypted, data **access patterns** leaked w/ operations still reveal a lot about it!

ORAM

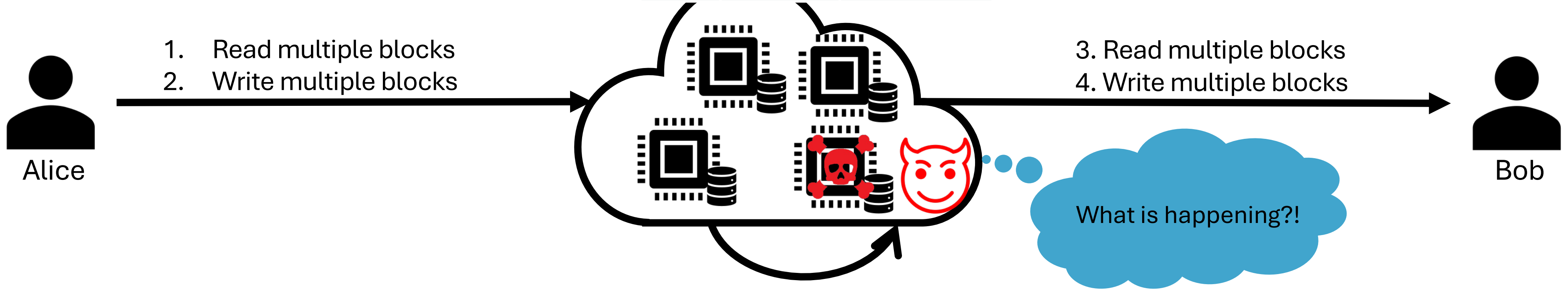
ID	Client	Balance
1	Alice	@$%&
2	Bob	#@&#



Oblivious RAM (ORAM) hides access patterns by accessing multiple data blocks w/ each operation and continuously shuffling data positions

ORAM

ID	Client	Balance
1	Alice	@$%&
2	Bob	#@&#

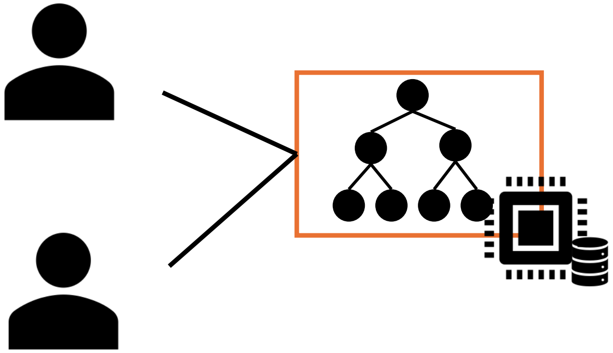


Oblivious RAM (ORAM) hides access patterns by accessing multiple data blocks w/ each operation and continuously shuffling data positions

But existing protocols are still limited regarding both **fault tolerance** and **concurrency**!

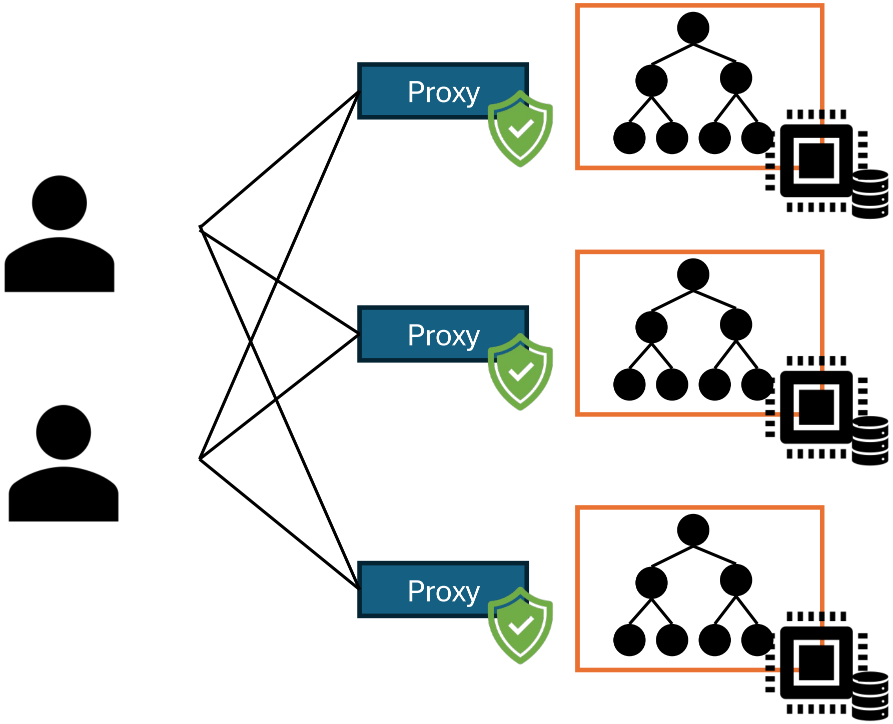
ORAM Fault Tolerance Landscape

Single server
(no fault-tolerance)



Most previous ORAMs (Path ORAM, OPRAM, ...)

Crash fault-tolerant replication
between proxies



QuORAM (USENIX Security'22)

ORAM Fault Tolerance Summary

- The only **fault-tolerant ORAM** supports only **crash faults**
  - i.e., no **Byzantine fault tolerance** in ORAM so far

ORAM Fault Tolerance Summary

- The only **fault-tolerant ORAM** supports only **crash faults**
  - i.e., no **Byzantine fault tolerance** in ORAM so far

- How difficult is it to add **Byzantine fault tolerance** to **ORAM?**
  - Actually **easy**, just run **ORAM** over a **BFT-SMR** protocol

ORAM Fault Tolerance Summary

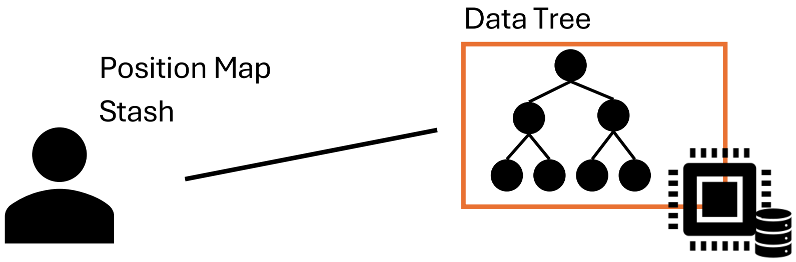
- The only **fault-tolerant ORAM** supports only **crash faults**
  - i.e., no **Byzantine fault tolerance** in ORAM so far

- How difficult is it to add **Byzantine fault tolerance** to **ORAM?**
  - Actually **easy**, just run **ORAM** over a **BFT-SMR** protocol

- The problem is that **blockchains** and **intrusion-tolerant services** expect clients to be **concurrent** and **independent**

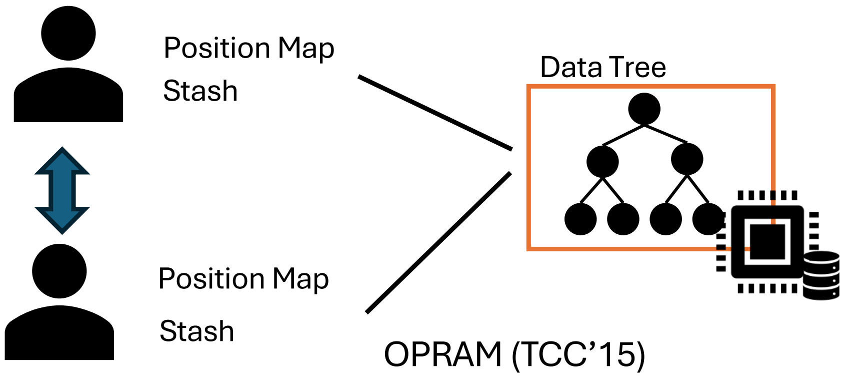
ORAM Concurrency Landscape

Single client (no concurrency)



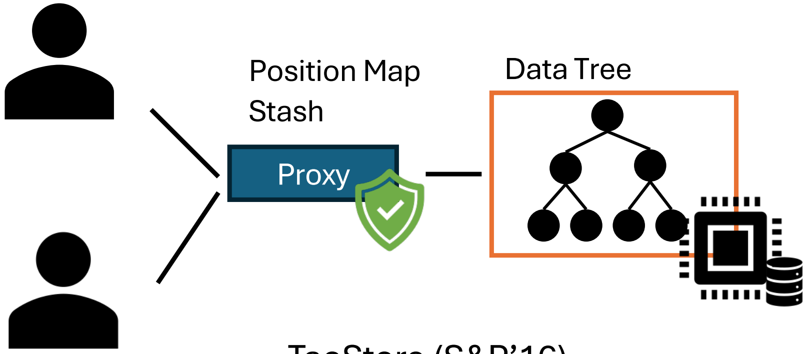
Path ORAM (CCS’13)

Intra-client communication



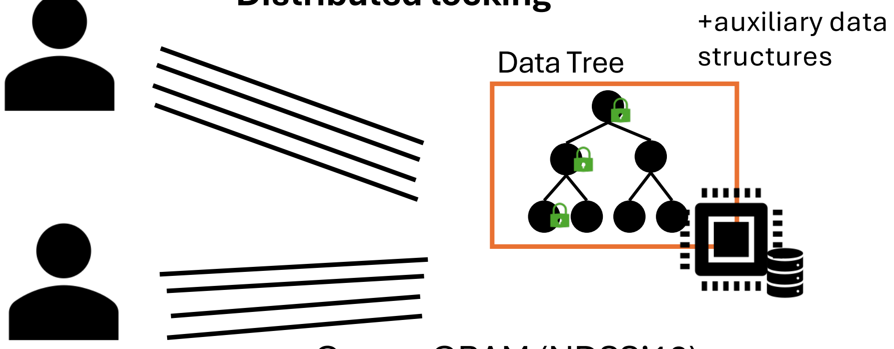
OPRAM (TCC’15)

Trusted Proxy



TaoStore (S&P’16)

Distributed locking



ConcurORAM (NDSS’19)

ORAM Concurrency Summary

- **Concurrent ORAMs** designs severely **limit concurrency** and **prevent wait-freedom**

- ***Wait-freedom*****:** every **client** is guaranteed to **finish** its operation in a **finite number of steps** (i.e., **independently** of the **delays** and **faults** of **other clients**)

ORAM Concurrency Summary

- **Concurrent ORAMs** designs severely **limit concurrency** and **prevent wait-freedom**

- ***Wait-freedom*:** every **client** is guaranteed to **finish** its operation in a **finite number of steps** (i.e., **independently** of the **delays** and **faults** of **other clients**)
  - **Essential property** in **BFT-SMR** and **concurrent systems** as it improves **concurrency** and provides **client fault tolerance**
    - Clients do not block waiting on each other
    - Faulty clients can not prevent other clients from terminating

Wait-Freedom vs ORAM Security

- However, achieving **wait-freedom** in **ORAM** introduces **new challenges**
  - As it **prevents** an essential property in **concurrent ORAM** known as **collision-freedom**

- ***<u>Collision-freedom:</u>*** no two clients should ever **access** the **same block** at the **same time**
  - As it would reveal access patterns

Wait-Freedom vs ORAM Security

- However, achieving **wait-freedom** in **ORAM** introduces **new challenges**
  - As it **prevents** an essential property in **concurrent ORAM** known as **collision-freedom**

- ***<u>Collision-freedom:</u>*** no two clients should ever **access** the **same block** at the **same time**
  - As it would reveal access patterns

- Indeed, **without client-synchronization** it seems **impossible** to achieve **collision-freedom**, which in turn **prevents wait-freedom**
  - At least in asynchronous networks, but more on that later ☺

Multi-Version Path ORAM

MVP-ORAM

The 1st Wait-Free ORAM

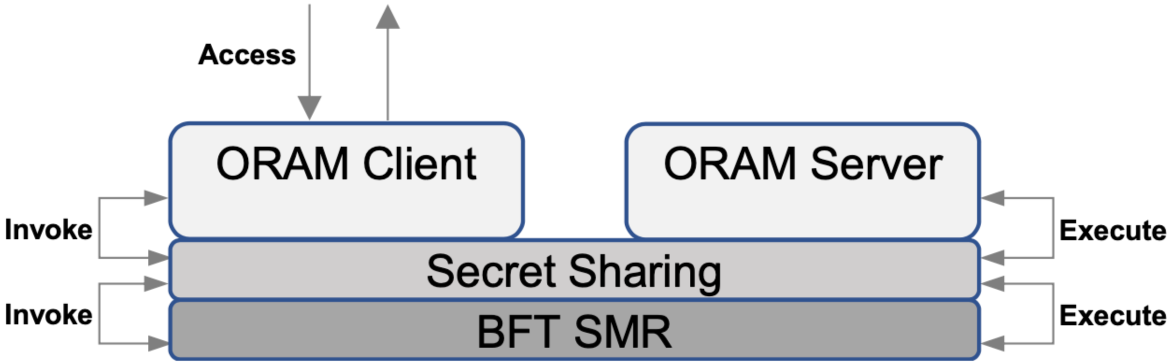
Can be easily replicated through BFT-SMR

Multi-Version Path ORAM

MVP-ORAM

The 1st Wait-Free ORAM

Can be easily replicated through BFT-SMR



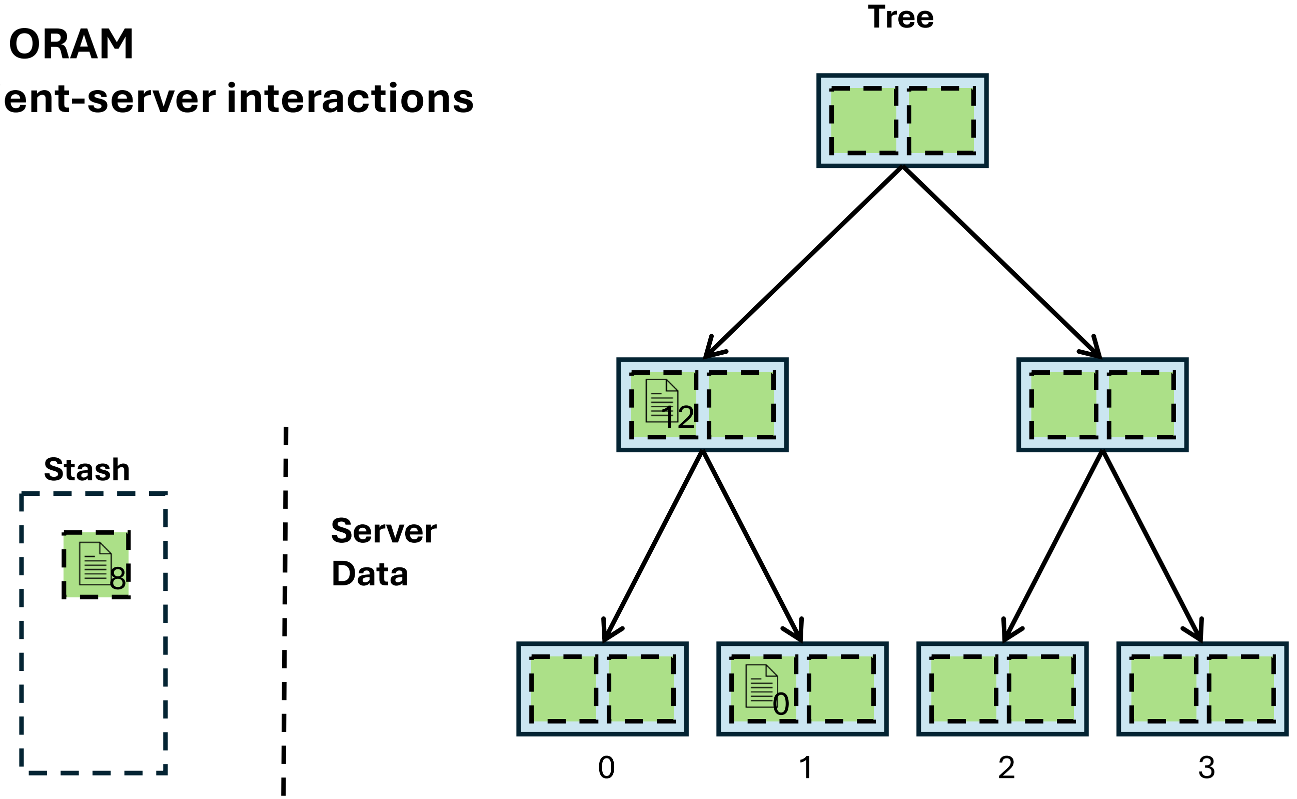
Path ORAM

- **MVP-ORAM** is based on **Path ORAM**
  - Due to **low number** of **client-server interactions** (important for BFT-SMR)

Client Data

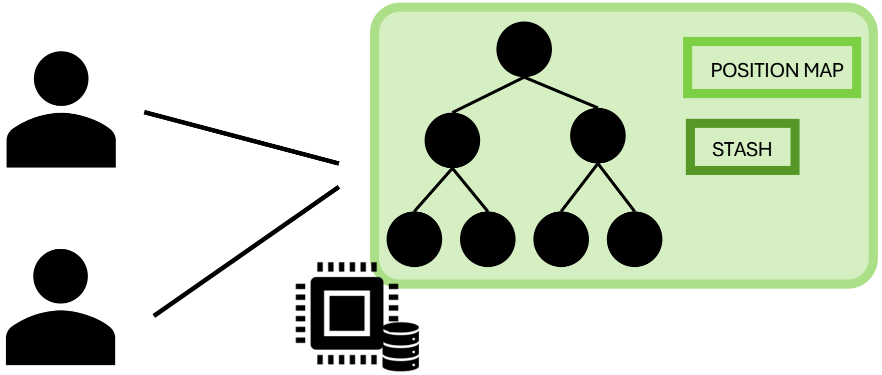
Position Map

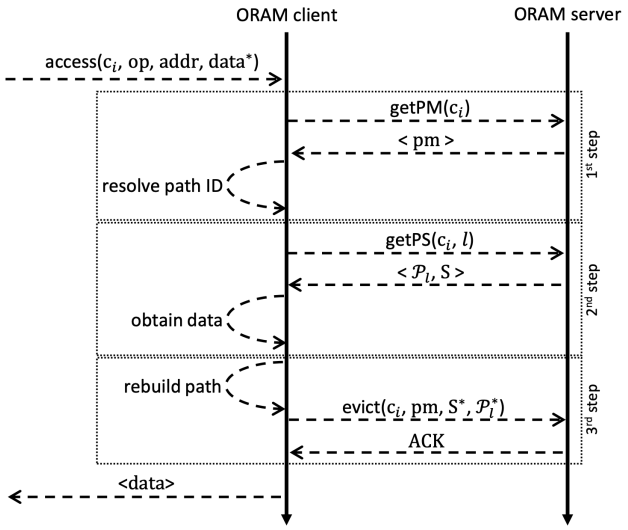
Address	Path
0	1
...	...
12	0



MVP-ORAM

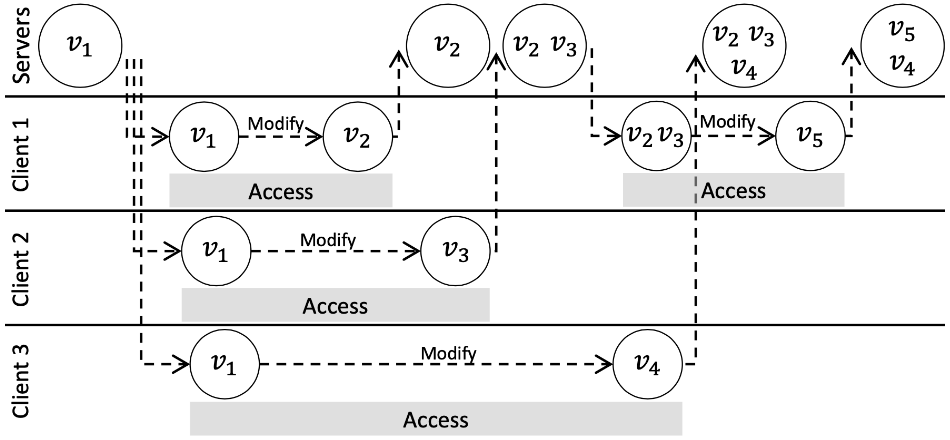
- In MVP-ORAM, **server stores all data-structures**
  - To access a block, client **1^{st} gets Position Map**, then **Path** and **Stash**





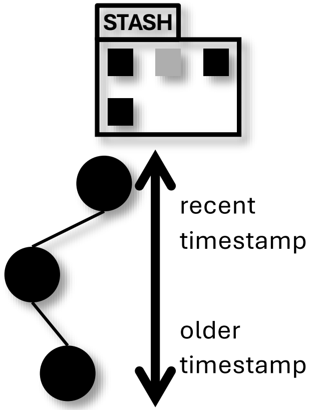
MVP-ORAM

- Due to no synchronization, there will be **concurrent updates**
  - **Server** stores these as **different versions** of the ORAM data
  - Before new access, clients **fetch existing versions** and **merge them together**



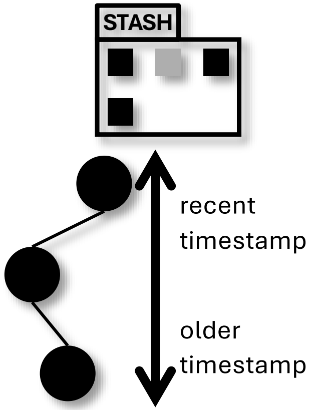
Dealing with collisions

- However, there is still the problem of **collisions** in **concurrent ORAM accesses**
    - Choose **at random** any **possible path** that allows reading a block
    - Keep **most popular blocks** either in **stash** or closer to the **root of the tree**



Dealing with collisions

- However, there is still the problem of **collisions** in **concurrent ORAM accesses**
    - Choose **at random** any **possible path** that allows reading a block
    - Keep **most popular blocks** either in **stash** or closer to the **root of the tree**
- Idea is that **most accessed blocks** will have **many possible paths**
    - Reducing probability of concurrent clients accessing same block through the same path

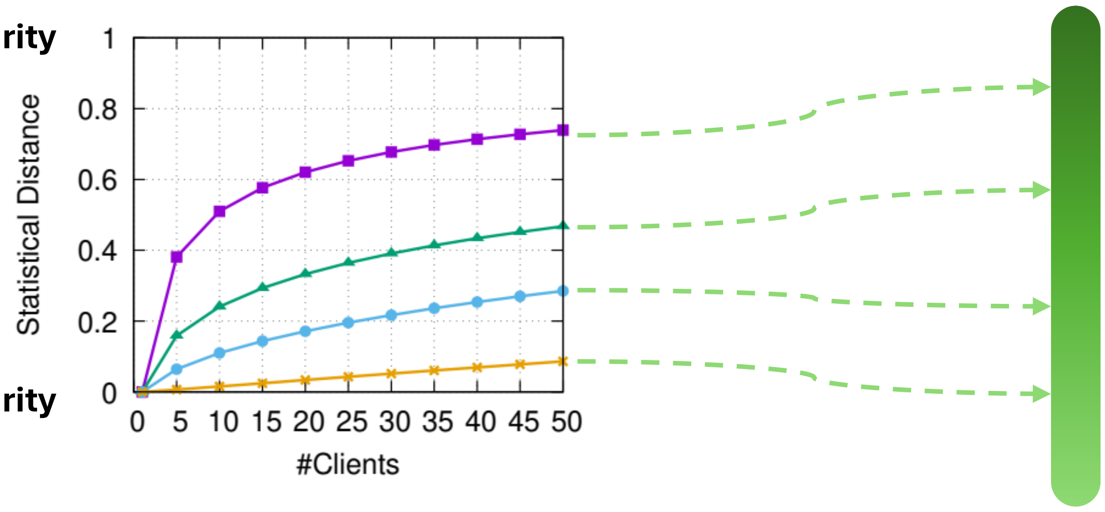


However, collisions can still happen...

e.g., all concurrent clients access a leaf block: only 1 path possible

Bad security

Good security



Access distribution

uniform

skewed

Access different blocks: Requested blocks more likely to be stored near leaves, decreasing number of possible paths

Access blocks from small subset: Requested blocks more likely to be in stash or higher in the tree, increasing number of possible paths

Worst case measurement

Security Notion for Asynchronous Wait-Free ORAM

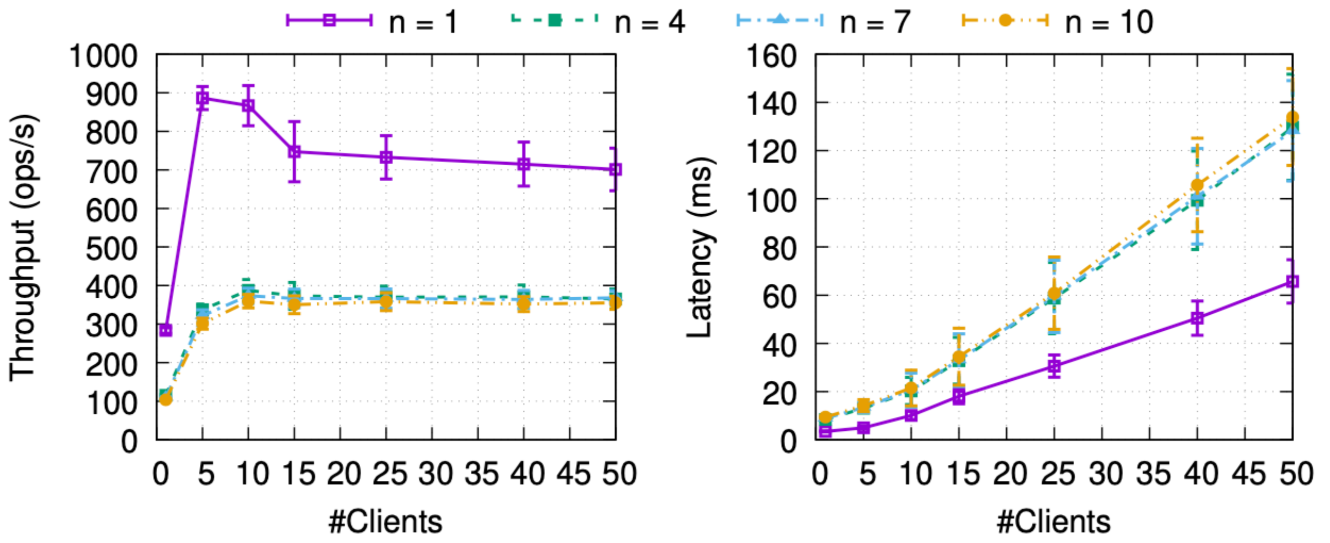
- **ORAM security** depends on:
  - **Database size *N*** (as in previous ORAMs)
  - # of **concurrent clients *c***
  - **Distribution** of **concurrent accesses *D***

- Also assumes adversary cannot inject its own queries
  - As that would allow it to force collisions between any blocks

Security Notion for Asynchronous Wait-Free ORAM

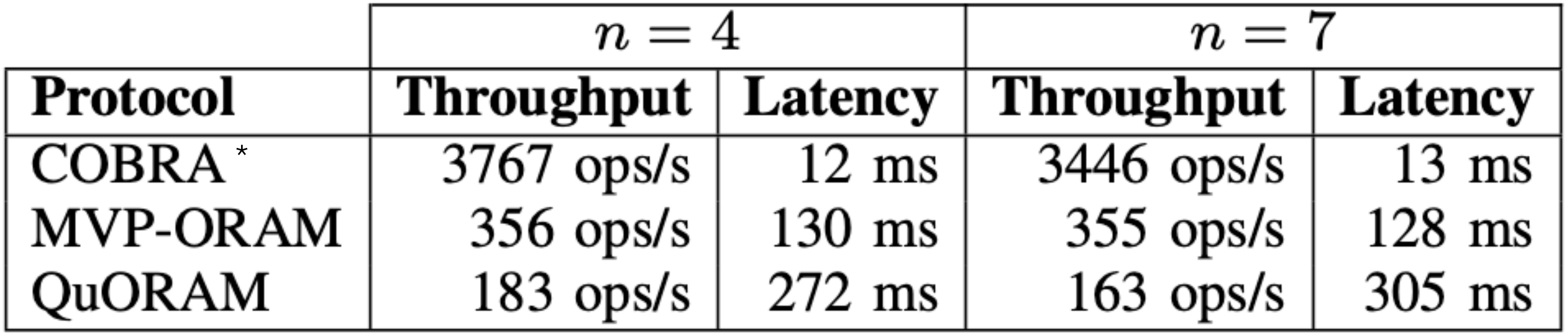
- **ORAM security** depends on:
  - **Database size *N*** (as in previous ORAMs)
  - # of **concurrent clients *c***
  - **Distribution** of **concurrent accesses *D***
- Also assumes adversary cannot inject its own queries
  - As that would allow it to force collisions between any blocks
- But is there really no way to provide wait-freedom w/ standard ORAM security? (There is, but need to sacrifice performance and asynchrony)

Experimental Results



Implemented in Java on top of BFT-SMaRt

Experimental Results



	$n = 4$		$n = 7$	
Protocol	**Throughput**	**Latency**	**Throughput**	**Latency**
COBRA *	3767 ops/s	12 ms	3446 ops/s	13 ms
MVP-ORAM	356 ops/s	130 ms	355 ops/s	128 ms
QuORAM	183 ops/s	272 ms	163 ops/s	305 ms

* COBRA is a confidential BFT datastore on top of BFT-SMaRt w/ DPSS (SP'22)

Key Takeaways

- **MVP-ORAM** is the first **wait-free Byzantine** fault-tolerant **ORAM**
  - Ensures **wait-freedom** by eliminating **client synchronization**
  - Tolerates **Byzantine server faults** through integration w/ **confidential BFT-SMR** data store
  - **Avoids collisions** in databases w/ **skewed access distributions**
  - **Weaker security notion** dependent on **#clients** and **access distribution**

MVP-ORAM: a Wait-free Concurrent ORAM for Confidential BFT Storage

Robin Vassantlal Hasan Heydari **Bernardo Ferreira** Alysson Bessani

{rvassantal, hheydari, **blferreira**, anbessani}@ciencias.ulisboa.pt

LASIGE, Faculty of Sciences, University of Lisbon, Portugal

Thank You! Questions?

NDSS 2026

LASIGE driven by excellence

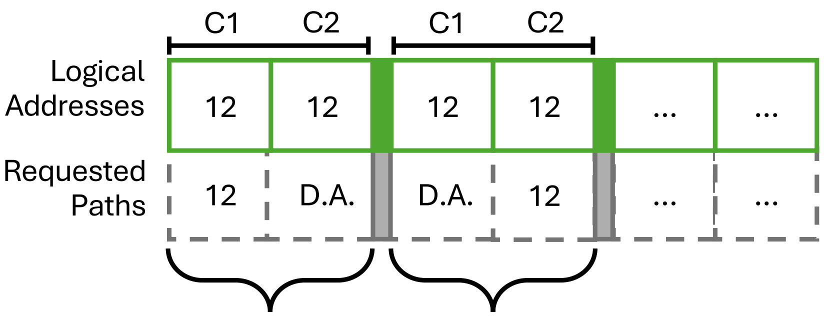
Ciências ULisboa

U LISBOA UNIVERSIDADE DE LISBOA

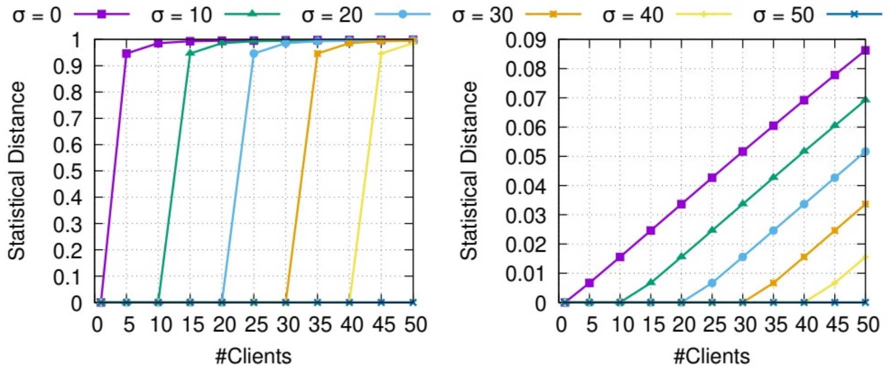
Strong MVP-ORAM

- Provides both **wait-** and **collision-freedom** by **sacrificing asynchrony*** assumptions and introducing **dummy accesses**†

An access becomes $\sigma + 1$ accesses, w/ σ =#dummy accesses



No concurrent accesses to the same path



(a) Uniform Distribution

(b) Skewed Distribution

* Requires assuming the relative speed of clients in executing operations is approximately the same

† Degrades performance by a factor of σ