

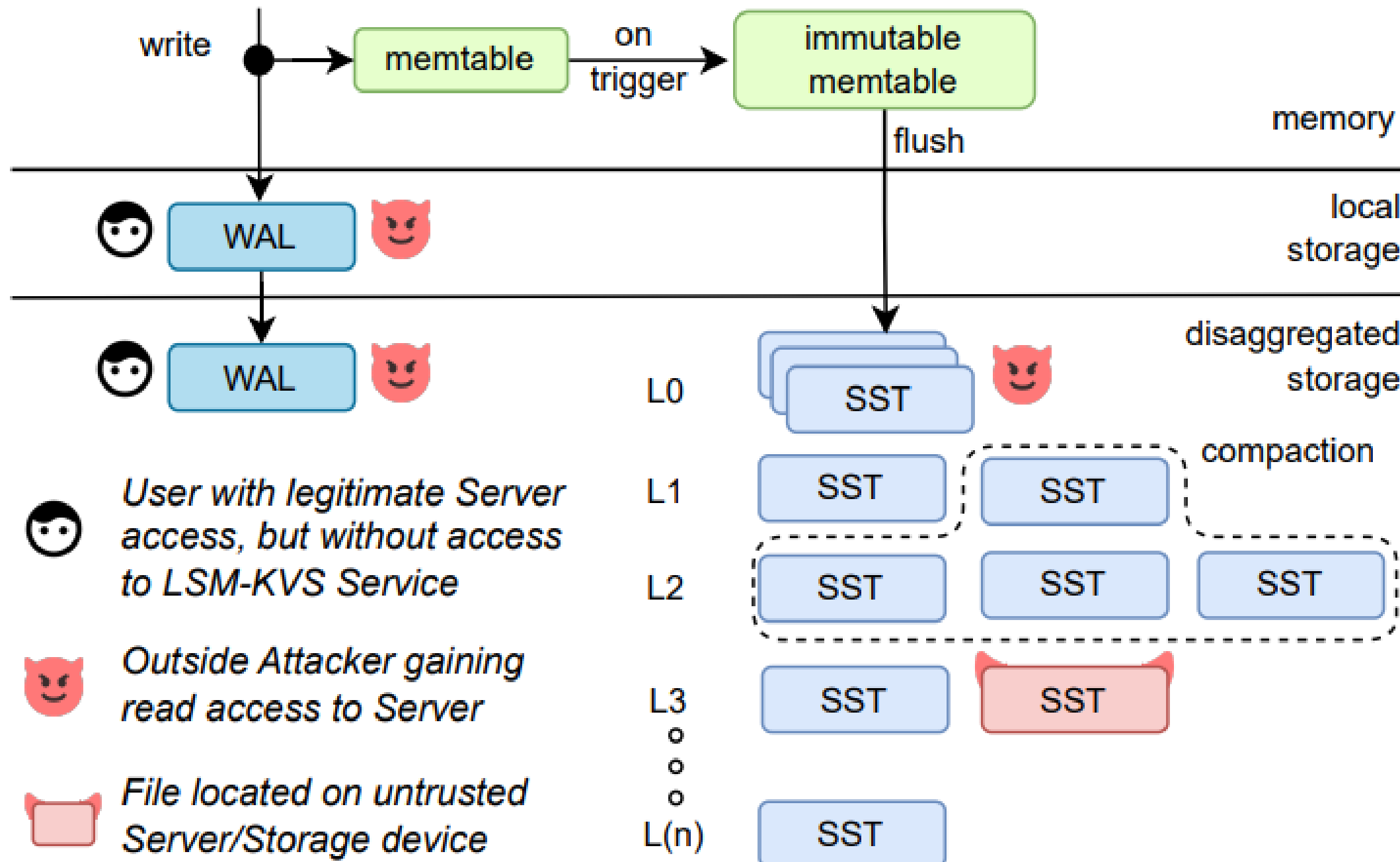
# SHIELD: Encrypting Persistent Data of LSM-KVS from Monolithic to Disaggregated Storage

Viraj Thakkar<sup>1</sup>, Dongha Kim<sup>1</sup>, Yingchun Lai<sup>2</sup>, Hokeun Kim<sup>1</sup>, Zhichao Cao<sup>1</sup>  
<sup>1</sup>Arizona State University, <sup>2</sup>Apache Pegasus (Incubating) Community



## THREAT MODEL

Plaintext LSM-KVS files on multiple servers with untrusted applications and large attack surface.



## MOTIVATION

- LSM-KVS in disaggregation has components (compaction, storage, WAL) across servers. **Solution must be flexible to disaggregated setups.**
- Data Encryption Key (DEK) practices (**Unique DEK per file and DEK-rotation**) necessary for robust protection in disaggregation.
- SOTA presents **high overhead (350-3,750%)** from using hardware-based TEEs. It's focus on in-memory protection **misses bottlenecks** (WAL-write), while using a **single DEK**.

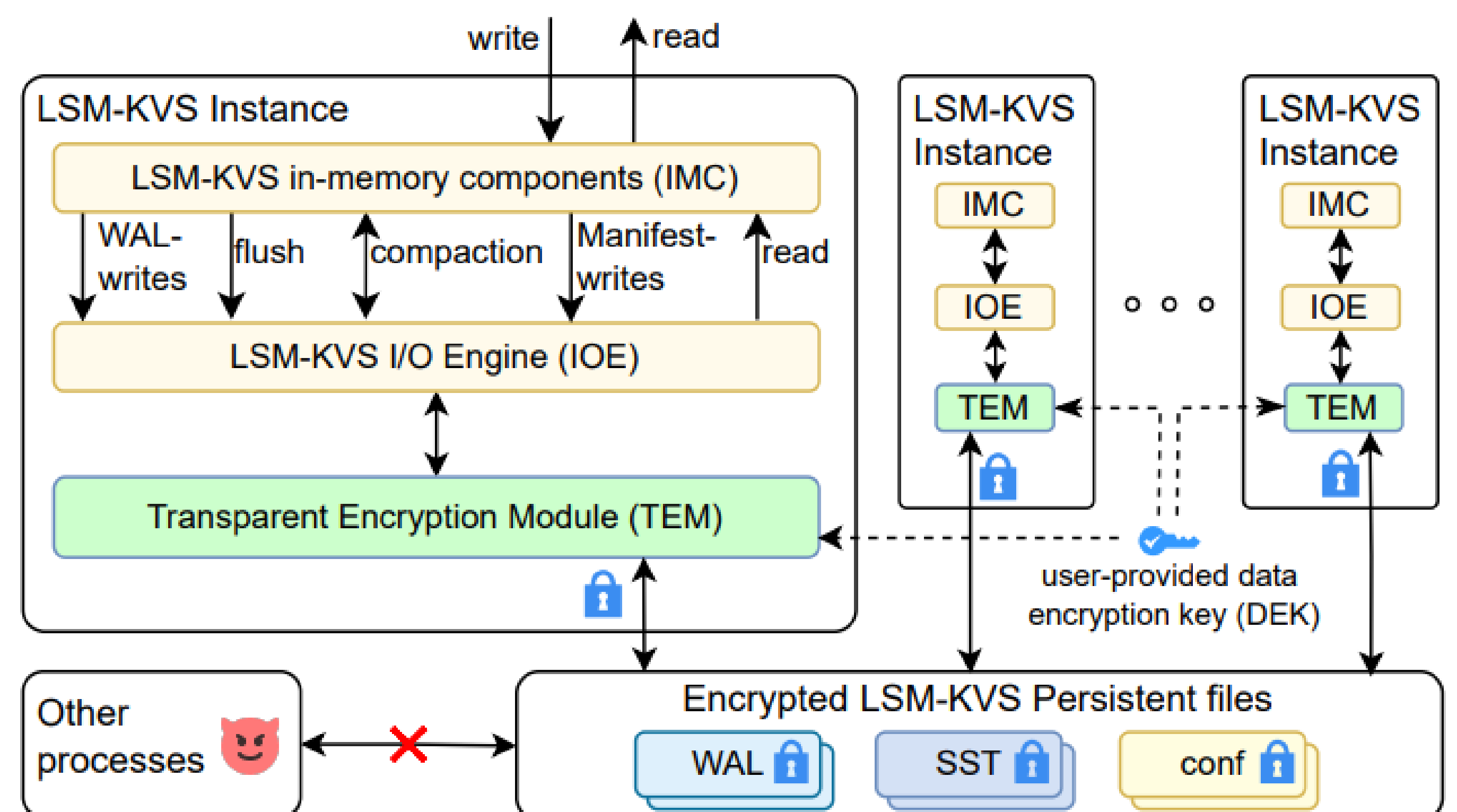
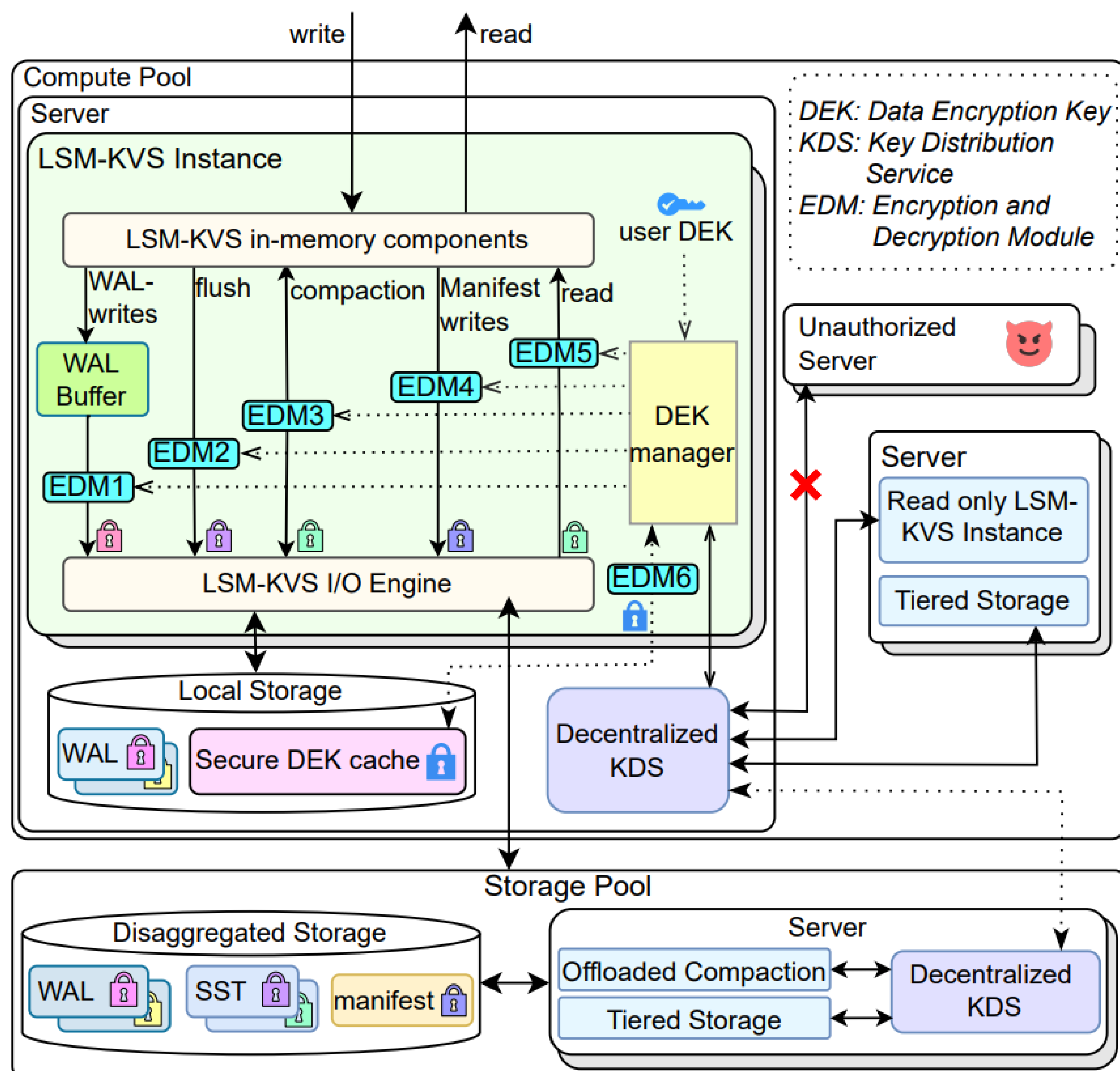
## CHALLENGES

1. How and where to embed DEK-practices in LSM-KVS?
2. How to **mitigate the WAL-Write Encryption Bottleneck**?
3. How to **co-ordinate DEK/LSM-file relationship** for flexible setups?

## ENCFS (DISTRIBUTED) & SHIELD (DISAGGREGATED STORAGE)

**EncFS (For Monolith and Distributed):**

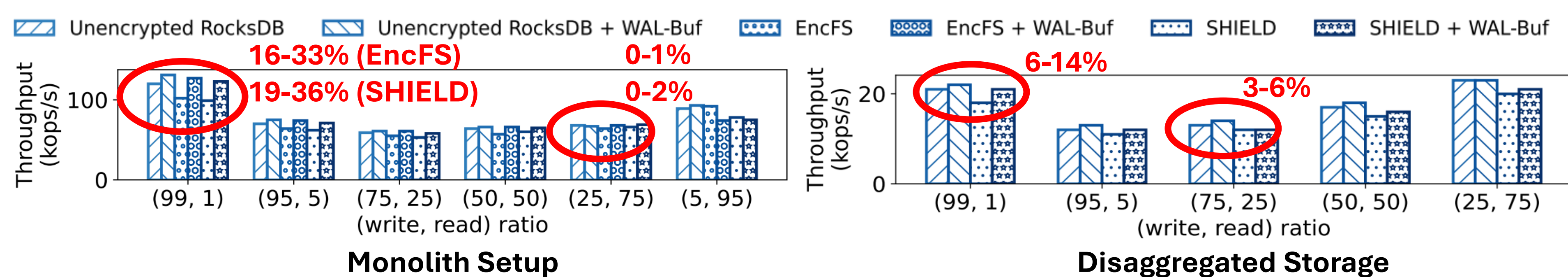
- **Intercept IO to/from FS** to enable transparent encryption in LSM-KVS.
- Simple solution but **DEK leak can compromise all data.**
- For **servers completely in your control**. Can be killed on-demand.



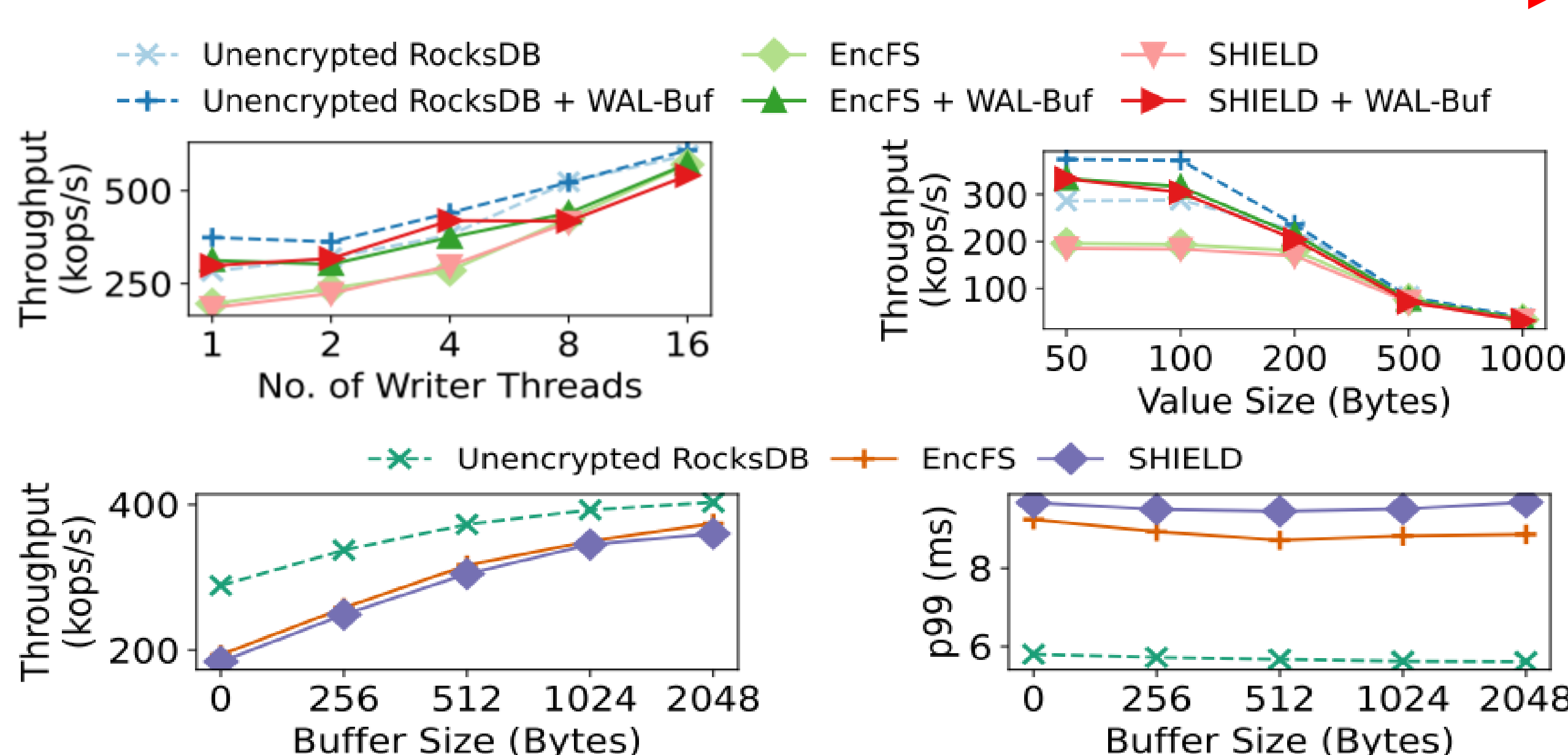
**SHIELD (For Disaggregated):**

- SST encrypted **after LSM block chunking**. And integration into **Compaction path for DEK-rotation**.
- **Size-configurable WAL-buffer** to mitigate WAL-bottleneck
- Decentralized key distribution service (e.g., SStoolkit, Kerberos) for DEK provision the **utilizes unique DEK identifiers**.
- **Local DEK-cache** secured with user-selected password for other LSM instances to use, avoiding network round-trips.
- **DEK-identifiers stored in LSM-KVS files' metadata**. Authorized servers avoid lookup tables, requesting DEK from KDS for passive and flexible DEK sharing.

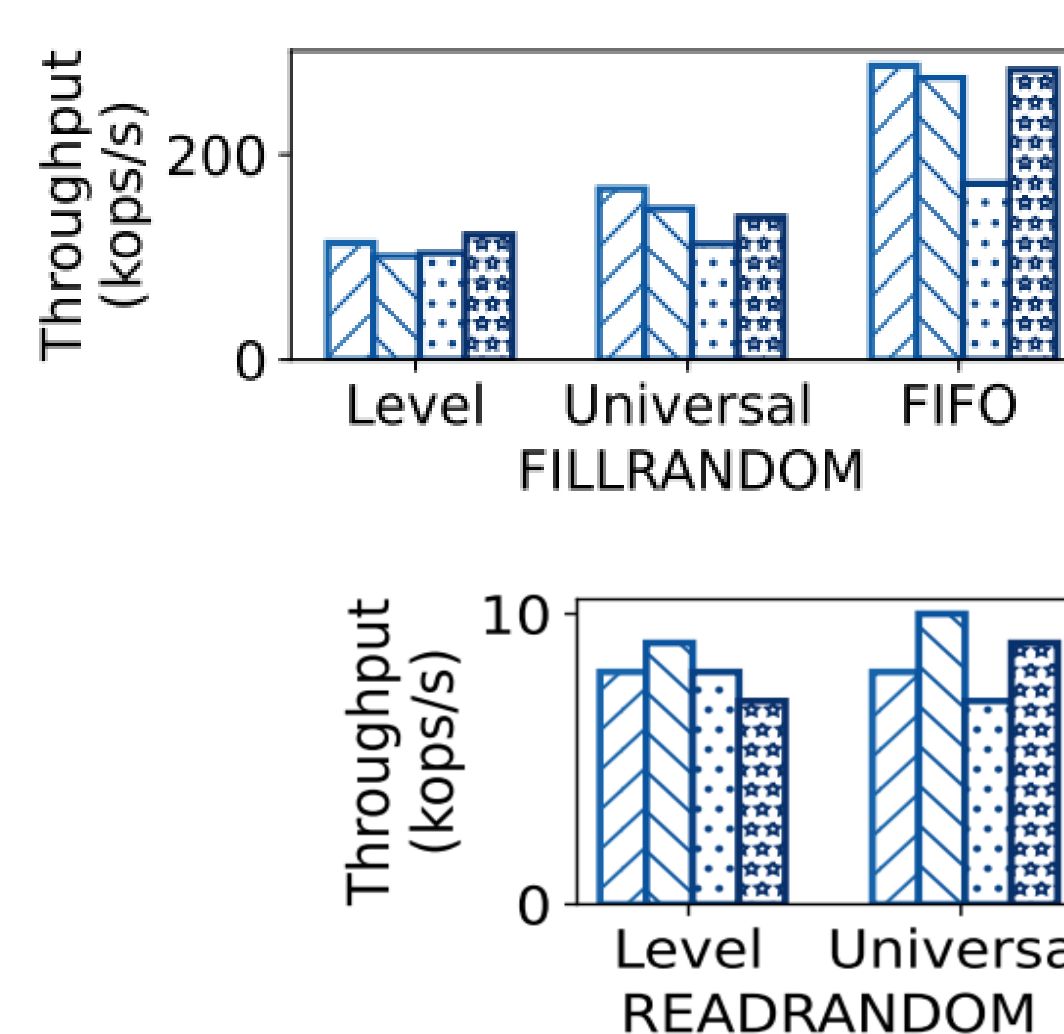
## EVALUATION RESULTS



**Converging Trend for EncFS & SHIELD**



**Compatible with different Compaction Styles**



## DISCUSSION

- We use 128-bit AES in CTR mode for encryption. **SHIELD is compatible with other encryption algorithms.**
- The solution comes with 0-36% overhead. **Encryption has a cost.** SHIELD aims to avoid the penalty and reduce round trips.
- We promise updates for **3 RocksDB major revisions** for the SHIELD codebase.

## FUTURE WORK

- What's the deal with TEEs? SOTA suffers high penalties, **can TEE-penalty be avoided?** Exploration needed.
- SHIELD focuses of DS. **Disaggregated Memory with CXL or RDMA devices** will be intriguing future work.

