Rethinking Block Storage Encryption with Virtual Disks

Danny Harnik
IBM Research

Oded Naor
Technion

Effi Ofer
IBM Research

Or Ozery
IBM Research

Block Storage Encryption

- Block storage is an abstraction of disks.
- Disk I/O is done at a sector granularity
  - Typically 512 bytes, today 4096 bytes.
  - Addressed via LBA (Logical Block Address)

- **Disk encryption** requires encrypting all data before it hits the disk
  - Main goal is Data-at-rest security - Protect against physical disk theft
  - Encryption is done at a sector granularity

- To keep alignment, use **length preserving encryption**
  - The ciphertext sector is of the exact same length as the plaintext sector

The Implications of Length Preserving Encryption

- No room for any additional per-sector information

1. **Encryption is deterministic** –
    - Encryption of the same plaintext will always result in the same ciphertext
    - Rules out ***Semantic Security***
    - Leaks information about data repeats at the granularity of an encryption block
    - **General encryption avoids determinism by using a per-sector nonce (IV)**
        - But in disk encryption no place to store the IV

2. **No authentication of encryption** –
    - Encryption is a 1-1 mapping, so every cipher maps to a legal plaintext
    - Changes or manipulation of ciphertext will be unnoticed by the data owner
    - **General encryption battles this using an integrity checksum (MAC)**
        - But in disk encryption no place to store the checksum

Handling of Block Storage Encryption Today

1. **Use LBA (sector number) for IV** –
    - Different addresses will never repeat the IV
    - Only overwrites use the same IV
2. **Devise schemes that are safe with repeating IV** –
    - Most popular is **AES-XTS** (before that AES-CBC)
    - Only information leaked is whether two “sub-blocks” at the same address have the same plaintext
        - With AES-XTS a sub-block is 32 or 16 bytes

- Note: AES-GCM leaks actual information about the data when IV repeats

Recap - Block Storage Encryption Today

- Today many block storage encryption mechanisms use **AES-XTS**
  - Android, Apple Filevault, Microsoft BitLocker and Linux dm-crypt (LUKS)

- **AES-XTS** has the following security compromises:
  - **Leaking change locations:** Given two versions of data written to the same sector one can detect exactly which sub-blocks have changed
  - **Data manipulation attacks:** Given two versions of a specific sector one can create a combination of sub-blocks from the two and form an encryption of data that never really existed.

- **Data-at-rest security** – stealing a disk is not a risk because it never contains two versions of the same sector
  - The above attacks are only relevant when eavesdropping to I/O traffic

Rethinking Encryption for Virtual Disks

1. Due to **snapshot** support several versions of the same sector can appear on the same disk – data-at-rest security no longer guaranteed

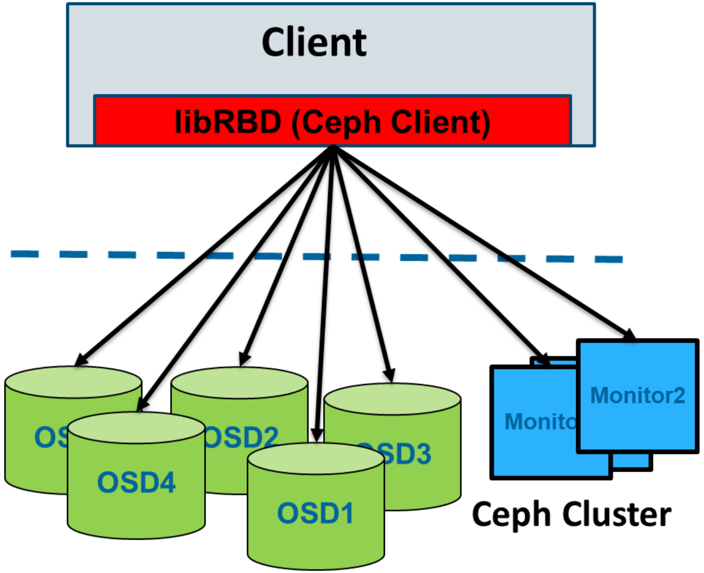
2. Virtual to Physical mapping is inherent
   - Can piggyback this to add per-sector metadata

Our Work

- Investigate how to integrate per-sector encryption metadata in a distributed block storage system – Ceph RBD
- Use it to add a random IV per sector – explore the security vs. performance tradeoff

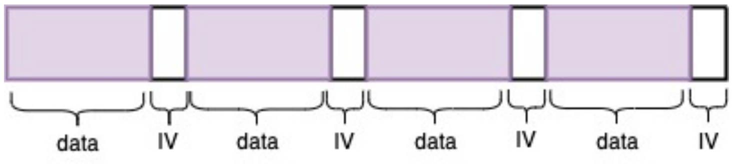
Ceph RBD and Encryption

- Ceph is a popular open-source distributed storage platform
  - Supports block, object and file storage
  - We focus on block – Ceph RBD
- Recently, disk encryption was added at the Ceph client (libRBD)
  - Compatible with standard LUKS encryption
  - Uses AES-XTS

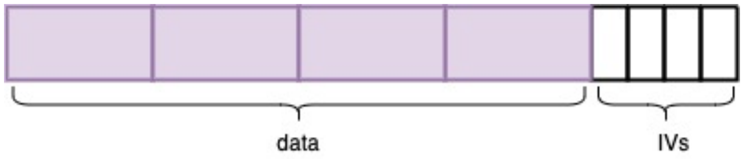


Implemented 3 Alternatives for Storing IVs

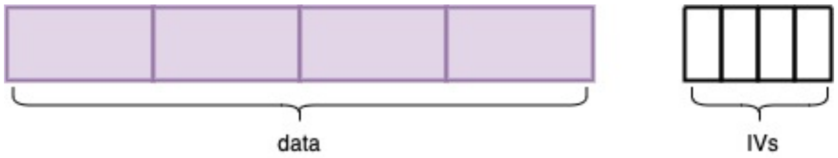
Unaligned – write IV sequentially after the sector



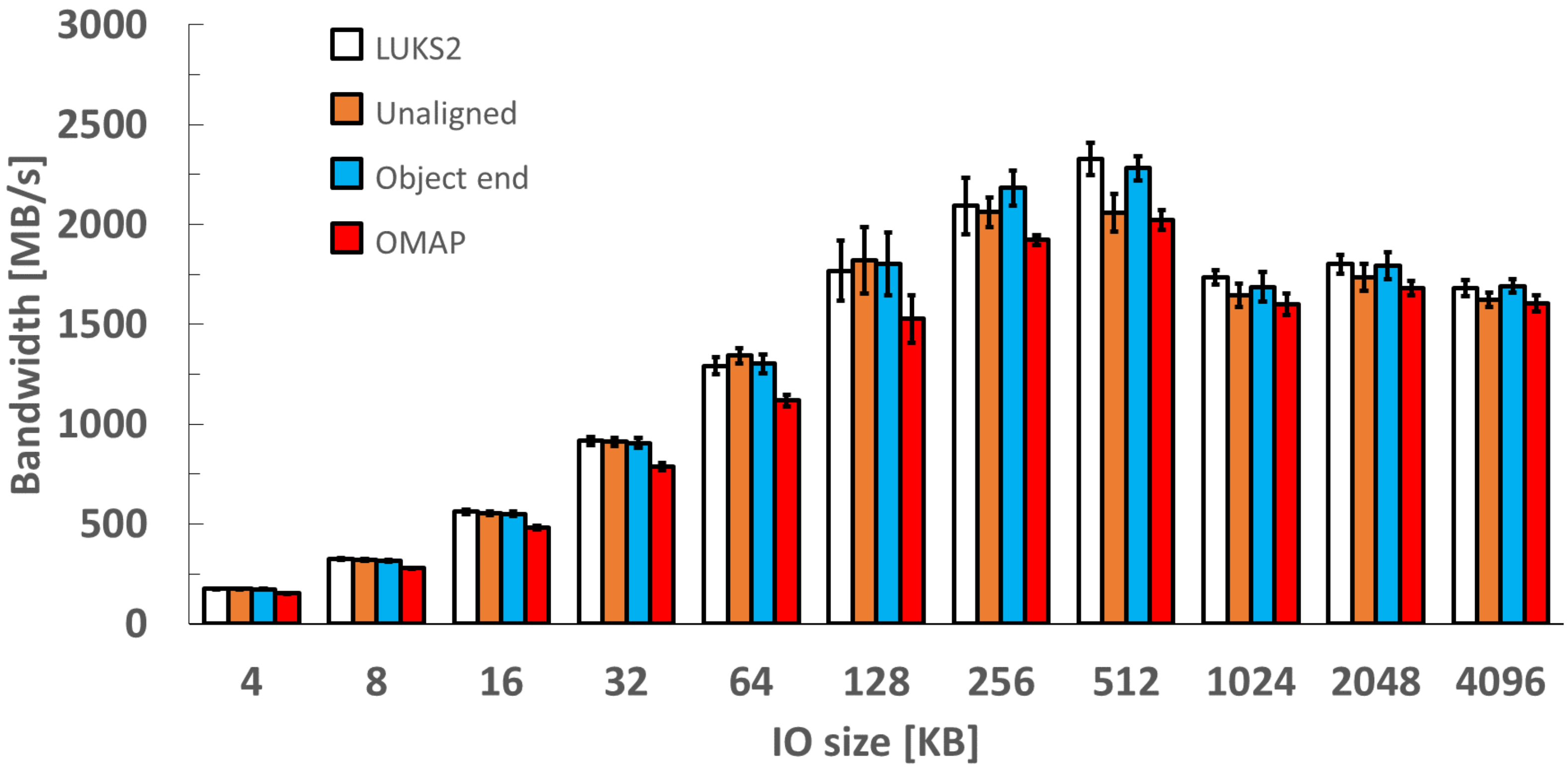
Object End – batch IVs to keep sector alignment. Use Ceph *object* granularity to batch all IV of an object at its end



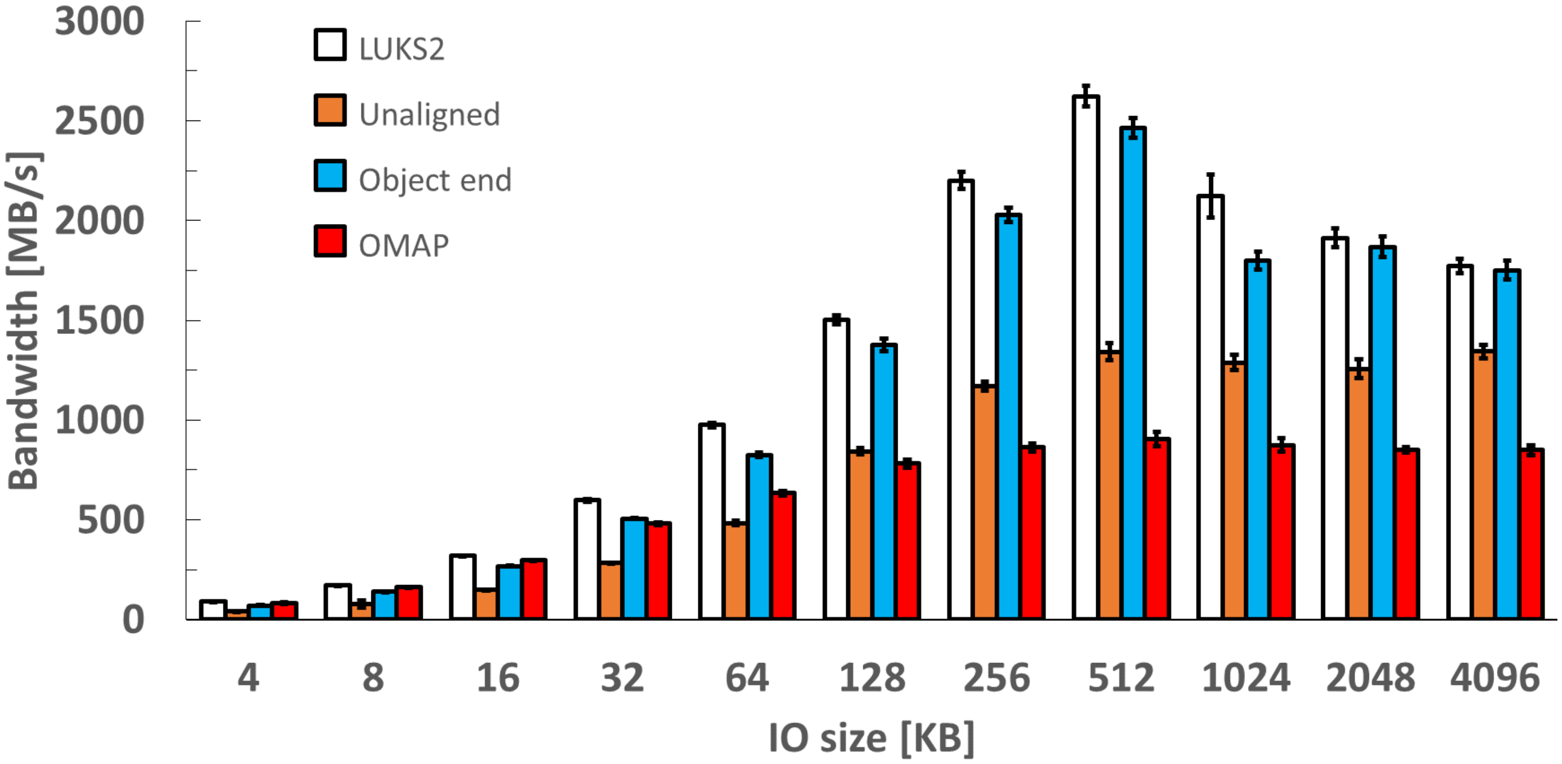
OMAP – Store the IVs in an External DB. Use Ceph OMAP DB which is based on RocksDB



Read Performance



Write Performance



Summary

- Today's commonly used block storage encryption compromises some security aspects
- We demonstrated that we can tradeoff some performance for better security in a distributed storage system (Ceph RBD)

- There were other attempts to tackle this:
  - High level - at dm-crypt using dm-integrity (incurred high performance hit)
  - Low level – at the FTL of an SSD

- Looking forward, storage with a native per-sector meta data support and API can allow encryption at a high level with negligible performance overhead