# Secure and Lightweight Deduplicated Storage via Shielded Deduplication-Before-Encryption

**Zuoru Yang**[1], Jingwei Li[2], Patrick P. C. Lee[1]

[1]The Chinese University of Hong Kong
[2]University of Electronic Science and Technology of China

USENIX ATC 2022

# Outsourced Storage

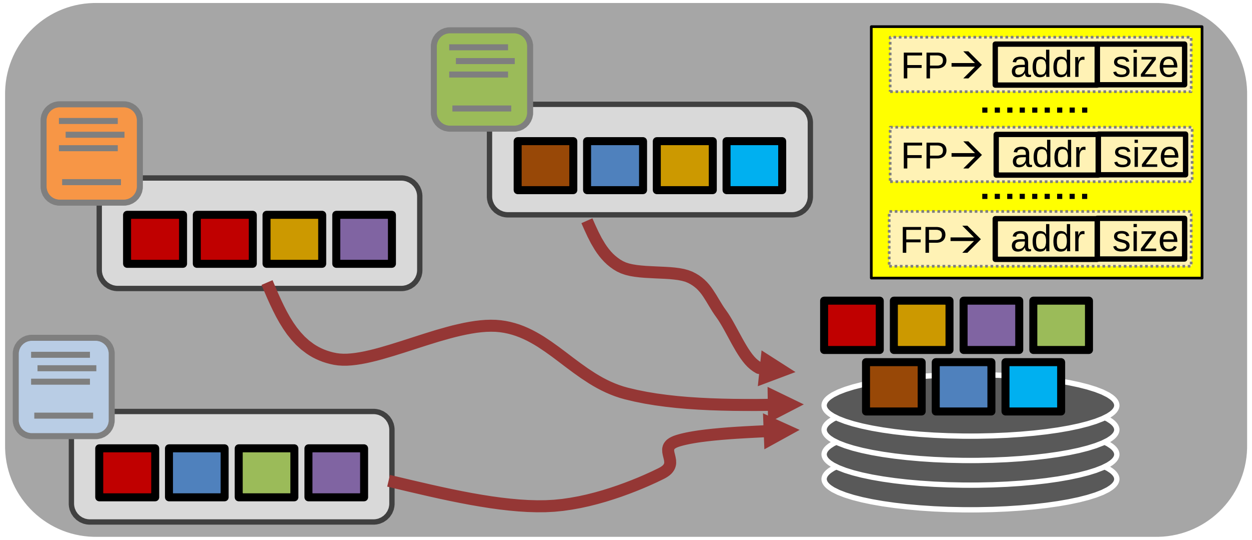
- Data outsourcing is a plausible storage solution in data explosion
  - Global datasphere grows to **175 ZB** by 2025
  - **49%** of the world's stored data will reside in public clouds [*]

- Two primary requirements
  - **Storage efficiency**: reduce storage overhead as much as possible
  - **Data confidentiality**: defend against data privacy leakage

[*] https://www.seagate.com/files/www-content/our-story/trends/files/idc-seagate-dataage-whitepaper.pdf

# Data Deduplication

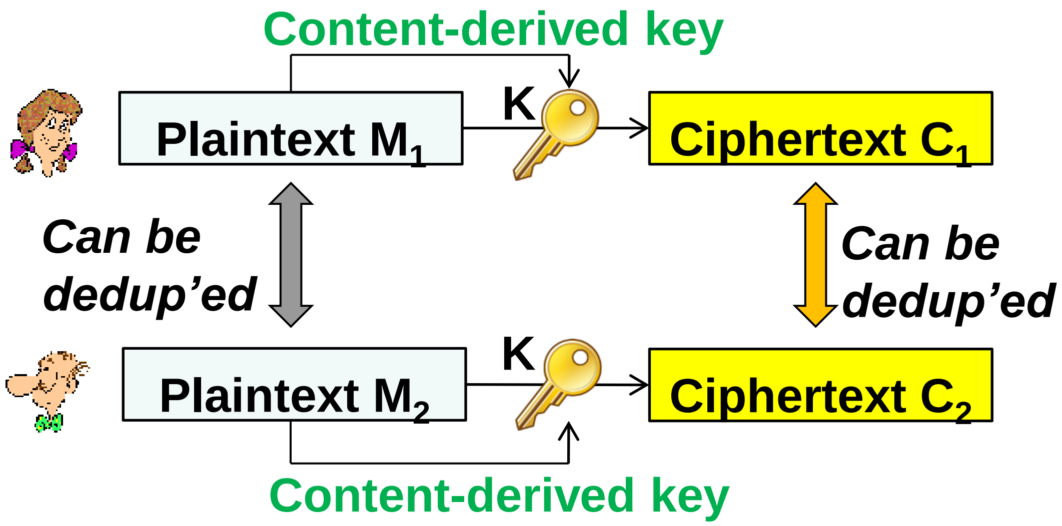
- A space-efficient storage approach
  - Unit: **chunk** (fixed-size or variable-size)
    - Compute a fingerprint for each chunk (e.g., SHA-256)
  - Manage fingerprint index to track stored chunks
    - Store only **one** copy of duplicate chunks
  - Achieve **~10x** storage space savings in backup workloads [Wallace, FAST'12]



# Deduplication-after-Encryption

- Deduplication-after-Encryption (**DaE**)
  - Augment deduplication with encryption for data confidentiality
  - Carefully encrypt chunks to preserve deduplication effectiveness on ciphertext chunks **after** encryption

- **Message-locked encryption** uses a key derived from chunk content [Bellare, EuroCrypt'13]
  - Enable **cross-user deduplication** on ciphertext chunks
    - e.g., Key = hash of plaintext chunk
  - Server-aided key management
    - Deploy a **key server** to prevent brute-force attacks [Bellare, Security'13]



# Limitations of DaE

- **L1: High key management overhead**
  - **Storage**: store a key for each chunk
  - **Performance**: key generation overhead is expensive [Ren, ATC'21]
- **L2: Incompatibility with compression**
  - Ciphertext chunks cannot be further compressed
    - Compression before encryption → leak compressed chunk lengths [Chen, SYSTOR'21]
- **L3: Security risks**
  - Single point-of-attack due to centralized server-aided key management
  - DaE is deterministic → vulnerable to frequency analysis [Li, EuroSys'20]

# Deduplication-before-Encryption

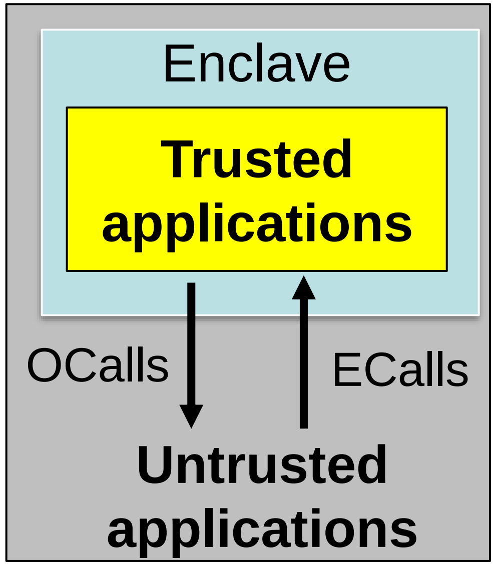
- Deduplication-before-Encryption (**DbE**)
  - We explore DbE, which performs deduplication on plaintext chunks, followed by encrypting non-duplicate chunks
- Benefits over DaE by design
  - Encryption can use **content-independent** keys (**L1** addressed)
  - Compression can be applied on **non-duplicate plaintext chunks** after deduplication (**L2** addressed)
  - Deploying a key server for key generation is unnecessary (**L3** addressed)
- **Question**: ***how should deduplication be protected?***
  - DbE's deduplication process is no longer protected by encryption

# Contributions

- **DEBE**: a shielded DbE-based deduplicated storage system based on shielded execution
  - Explore DbE with aid of **Intel SGX**
  - Apply **frequency-based deduplication** for performance and security
- Experiments show that DEBE outperforms conventional DaE approaches in **performance**, **storage savings**, and **security**
  - Up to **13.1x** upload speedup over DupLESS [Bellare, Security'13]
  - **93.8%** key metadata storage saving over DaE
  - Reduce information leakage without compromising storage efficiency

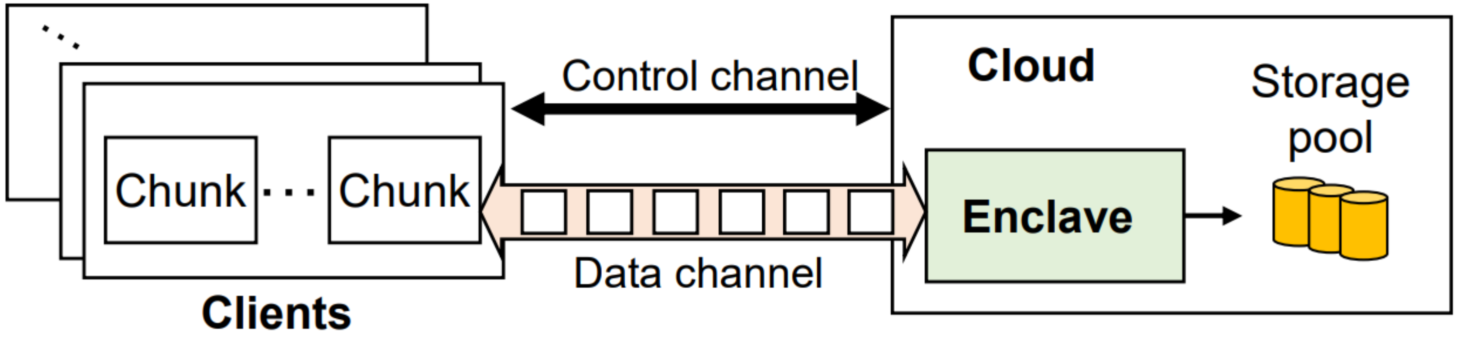
# Intel SGX Basics

- Enclave: secure memory region realized by Intel SGX
  - **OCalls** and **ECalls** to interact with untrusted applications
- SGX limitations in performance
  - Enclave page cache (EPC) has limited size (e.g., 128 MiB)
    - Exceeding EPC size → expensive EPC paging overhead
  - ECalls and OCalls lead to context-switching overhead




- **Challenge:** ***How to mitigate SGX overhead in DEBE?***

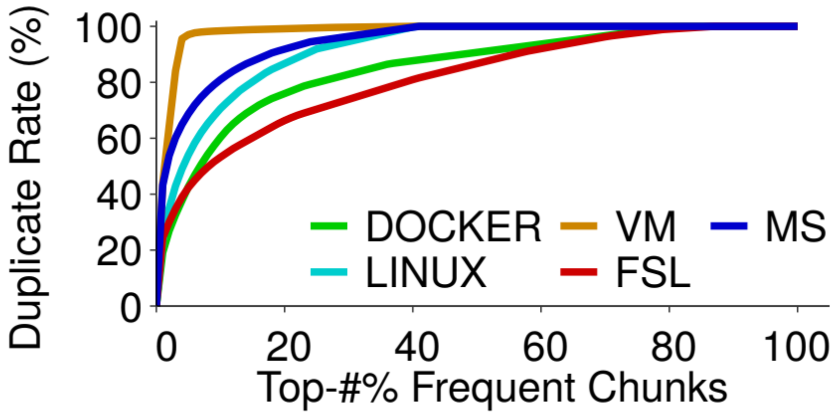
# Overview




- Target-based deduplication
  - Protect DbE via Intel SGX
  - Perform deduplication and compression over plaintext chunks **in enclave**
- Communication
  - **Control channel:** transmit commands for storage operations
  - **Data channel:** transmit plaintext chunks to enclave
    - Protected by a short-term session key shared by a client and enclave

# Main Idea

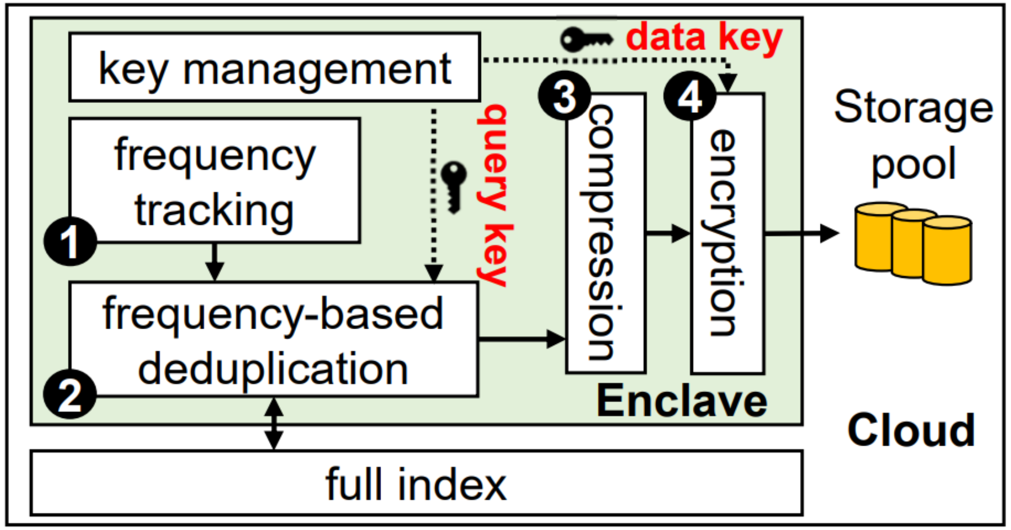
- A small fraction of top frequent chunks contribute a large fraction of duplicates
  - In VM, **top-5%** of frequent chunks contribute to a duplicate rate of **97%**




- **Frequency-based deduplication**: separate deduplication process in two phases based on chunk **frequencies**
  - **First phase**: Manage small fingerprint index in enclave to remove most duplicates → mitigate EPC paging overhead
  - **Second phase**: Manage full index out of enclave to remove remaining few duplicates → reduce context-switching overhead

# Architecture

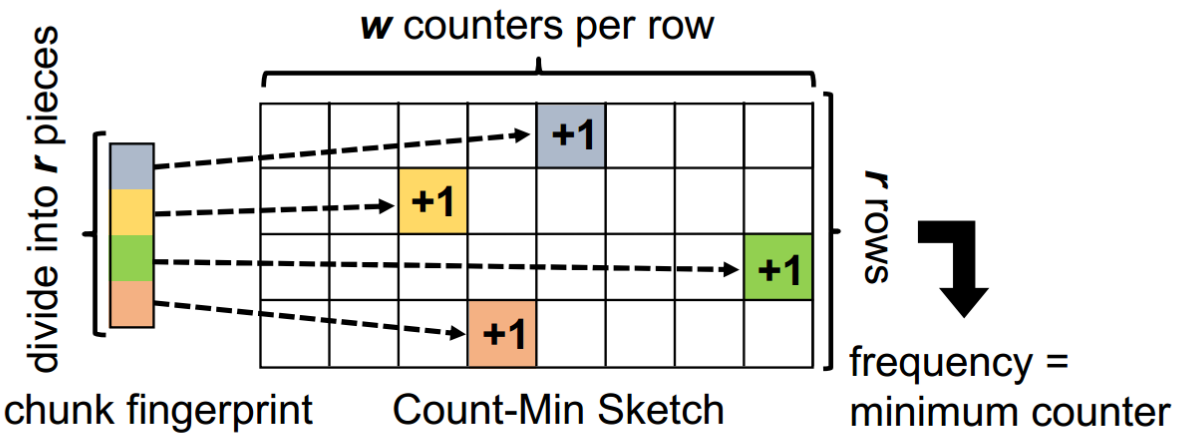
- ➢ Track frequencies of plaintext chunks
- ➢ Frequency-based deduplication
  - Remove duplicates of **most frequent** chunks
  - Query full index to remove remaining duplicates of **less frequent** chunks
    - Protect query information via **query key**




- ➢ Compress non-duplicate chunks and encrypt compressed chunks via **data key**

# Frequency Tracking

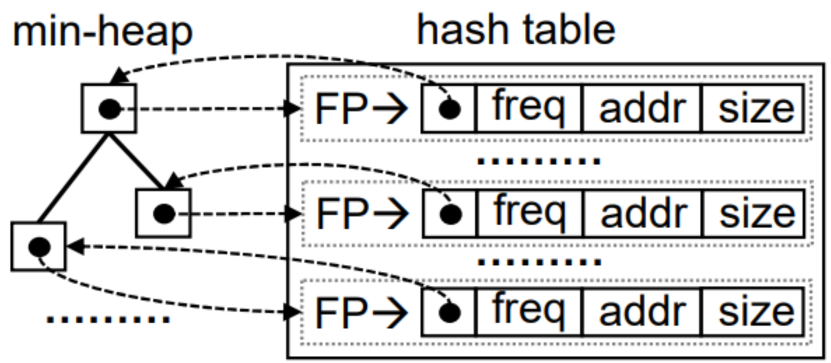
- Use **Count-Min Sketch** (CM-Sketch) to track **approximate** frequency of each chunk
  - Fixed memory usage with provable error bounds
  - Divide fingerprint into **r** pieces for counting
  - Nearly no extra performance overhead



# First-Phase Deduplication

- Remove duplicates from *k* most frequent plaintext chunks
  - Expect to remove a large fraction of duplicates

- Manage **top-*k* index** in enclave
  - Limited EPC usage → O(*k*)
  - Min-heap to differentiate the top-*k*-frequent and less frequent chunks
  - Hash table to track chunk information for duplicate detection



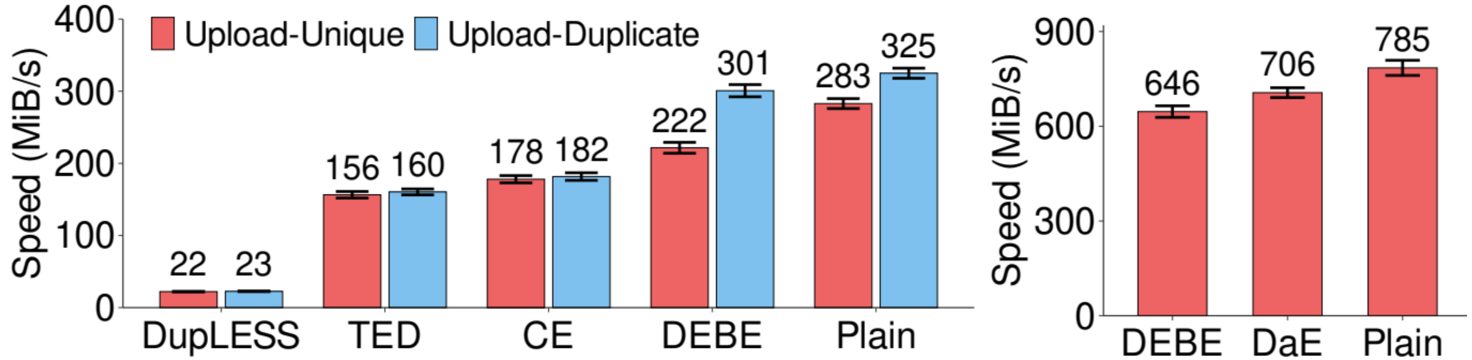
# Second-Phase Deduplication

- Remove duplicates from remaining less frequent chunks
- Manage **full index** outside enclave
  - Protected by query key
  - Hash table: encrypted fingerprint → encrypted chunk information
- Enclave **deterministically** encrypts the fingerprint of each remaining plaintext chunk with query key
  - Query full index via Ocalls

# Experimental Setup

- Implement DEBE in C++ on Linux
  - Intel SGX SDK Linux 2.7, OpenSSL 1.1.1, and Intel SGX SSL
  - FastCDC, LZ4
  - ~17.5 K LoC
- Datasets
  - Five real-world backup workloads: DOCKER, LINUX, FSL, MS, and VM
- Testbed
  - Multiple machines connected with 10GbE
  - Each machine has Intel Core i5-7500 3.4GHz and 32GiB RAM

# Overall Performance



(a) Upload (b) Download

- ➢ Baselines
  - DupLESS [Bellare, Security'13]
  - TED [Li, EuroSys'20]
  - CE [Bellare, EuroCrypt13]
  - Plain (without encryption)

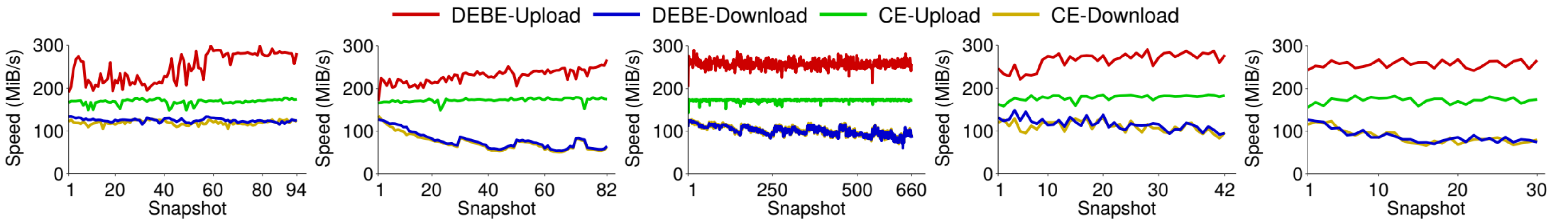
➢ DEBE outperforms all DaE approaches in uploads

- Up to **13.1x** speedups over DupLESS
  - Avoid key generation performance overhead
  - Avoid encryption and compression for duplicate data

➢ **8.5%** download speed drops compared with DaE

- Load data into enclave for decryption and decompression

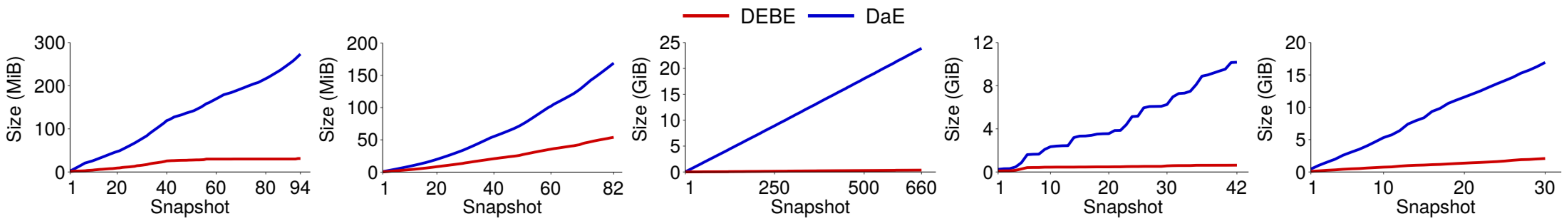
# Trace-Driven Performance



(a) DOCKER (b) LINUX (c) VM (d) FSL (e) MS

- DEBE outperforms CE in uploads
  - FSL: **246.5-277.5** MiB/s in DEBE; **163.5-179.1** MiB/s in CE
- Download speeds of both DEBE and CE are **almost identical**
  - Throttled by disk I/O

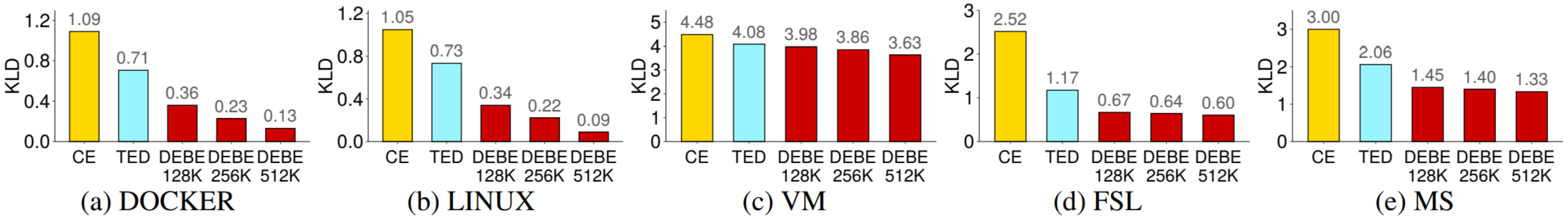
# Storage Efficiency



(a) DOCKER (b) LINUX (c) VM (d) FSL (e) MS

- In FSL, DEBE saves **93.8%** of key metadata compared with DaE
  - DaE: a 32-byte key for each chunk (in AES-256)
  - DEBE: two long-term keys (data key and query key); a 16-byte IV for each **non-duplicate** chunk
    - As in traditional symmetric encryption

# Security




- Quantify frequency leakage by KLD (a.k.a., relative entropy to uniform distribution)
  - Low KLD implies high security
- Reduce KLD of TED [Li, EuroSys'20] by up to **87.7%** in LINUX
  - TED needs to store 15% more data to enhance security

# Conclusion

- **DEBE** realizes DbE via Intel SGX
  - Perform deduplication and compression in enclave
  - Apply frequency-based deduplication
  - Outperform DaE approaches in performance, storage, and security
- See our paper and technical report for more details
- Source code: ***https://github.com/yzr95924/DEBE***
  - Received all three artifact badges