Balancing Storage Efficiency and Data Confidentiality with Tunable Encrypted Deduplication

Jingwei Li[*], Zuoru Yang[#], Yanjing Ren[*], **Patrick P. C. Lee[#]**, Xiaosong Zhang[*]

[*]University of Electronic Science and Technology of China (UESTC)
[#]The Chinese University of Hong Kong (CUHK)

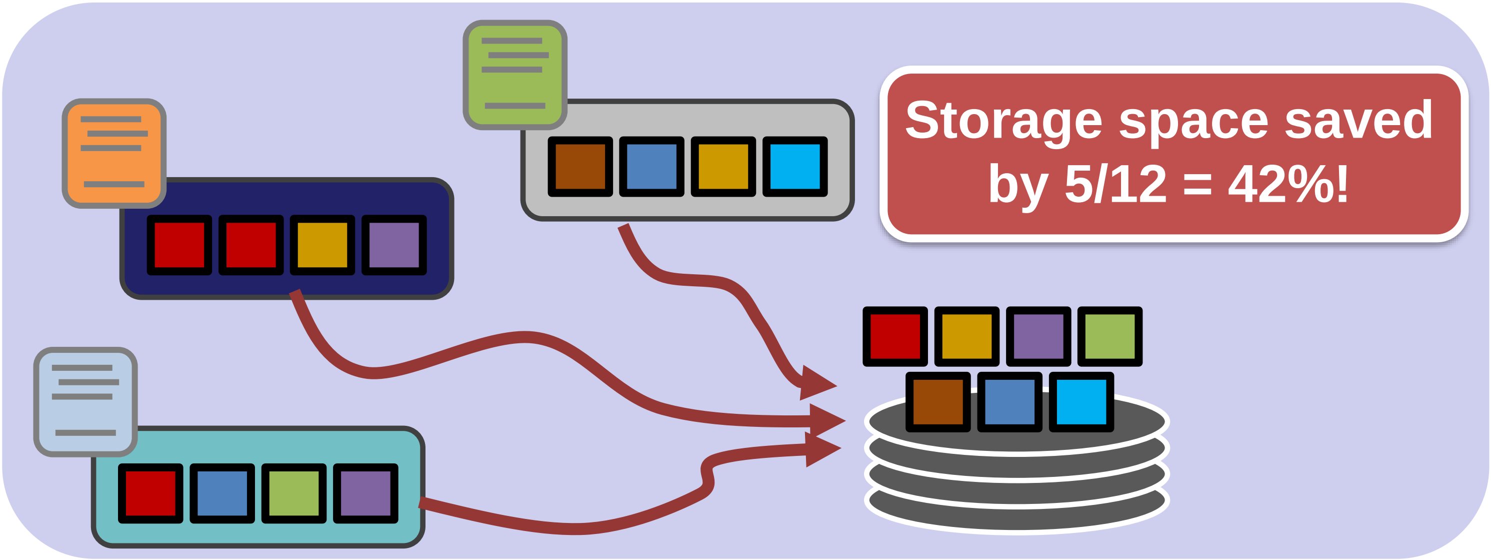
EuroSys 2020

Deduplication

➢ Deduplication → coarse-grained compression

- Units: **chunks** (fixed- or variable-size)

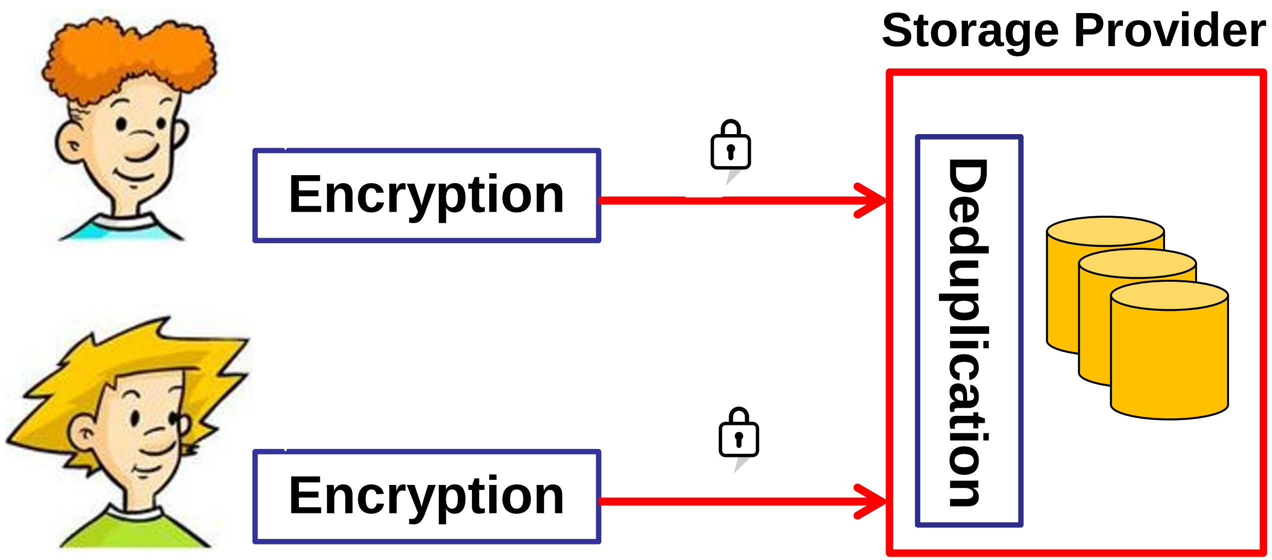
➢ Stores only one copy of duplicate chunks



Encrypted Deduplication

- Augments deduplication with encryption for data confidentiality
- Application: outsourced storage

Which crypto primitive should be used?



Encryption Primitives

- **Symmetric-key encryption (SKE)**
  - Derives a random key for chunk encryption/decryption
  - Ensures confidentiality, but **prohibits deduplication** of duplicate chunks
- **Message-locked encryption (MLE)** [Bellare et al., Eurocrypt'13]
  - Derives a deterministic key from chunk content
  - Supports deduplication, but **leaks frequency distribution** of plaintext chunks [Li et al., DSN'17]

Pose a dilemma of choosing the right cryptographic primitive

Our Contributions

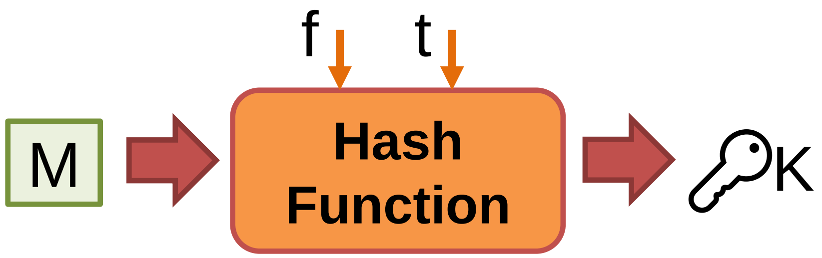
- **TED**: a tunable encrypted deduplication primitive for balancing trade-off between storage efficiency and data confidentiality
  - Includes three new designs
  - Minimizes frequency leakage via a configurable storage blowup factor

- **TEDStore**: encrypted deduplication prototype based on TED
  - TED incurs only limited performance overhead

- Extensive trace-driven analysis and prototype experiments

Main Idea

➢ Key derivation with three inputs: chunk **M**, current frequency **f**, and balance parameter **t**



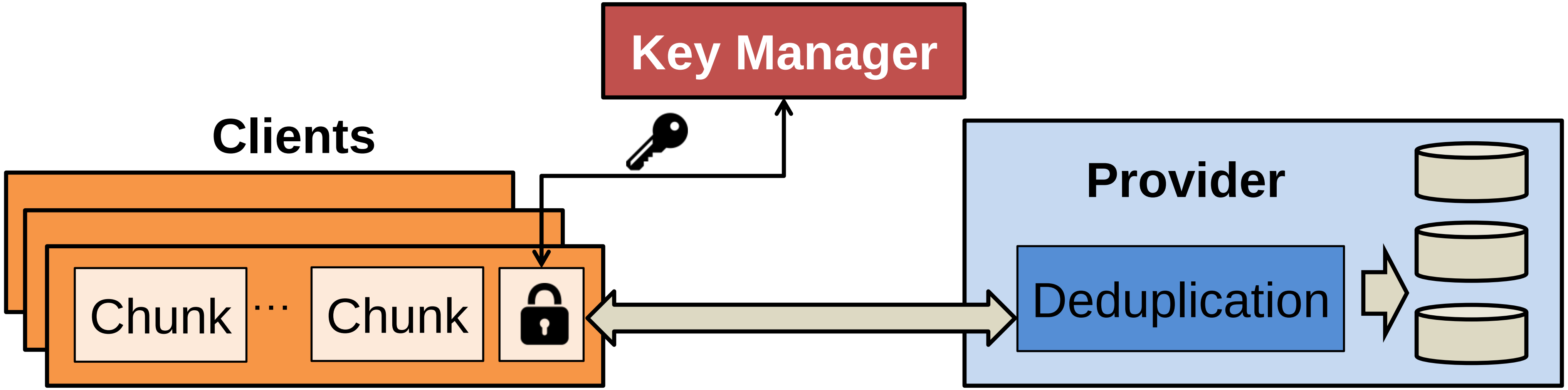
$K = \mathbf{H}(M \,||\, \lfloor f/t \rfloor)$

- f: cumulative and increases with number of duplicates of M
- t: controls maximum allowed number of duplicate copies for a ciphertext chunk

➢ Special cases:

- $t = 1$ → SKE
- $t \rightarrow \infty$ → MLE

Design Overview



- TED builds on **server-aided MLE architecture** in DupLESS [Bellare et al., Security’13]
  - Key generation by **key manager** to prevent offline brute-force attacks

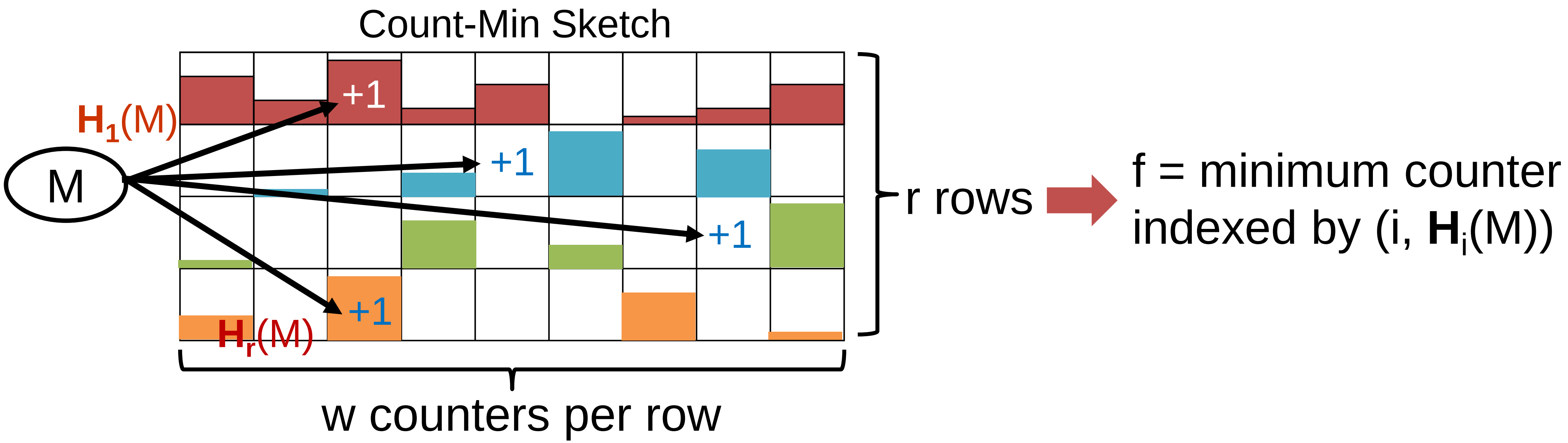
Questions

- Q1: How does the key manager learn chunk frequencies?
  - Low overhead required even for many chunks

- Q2: How does the key manager generate keys for chunks?
  - Distinct sequences of ciphertext chunks required for identical files

- Q3: How should the balance parameter t be configured in practice?
  - Adaptive for different workloads

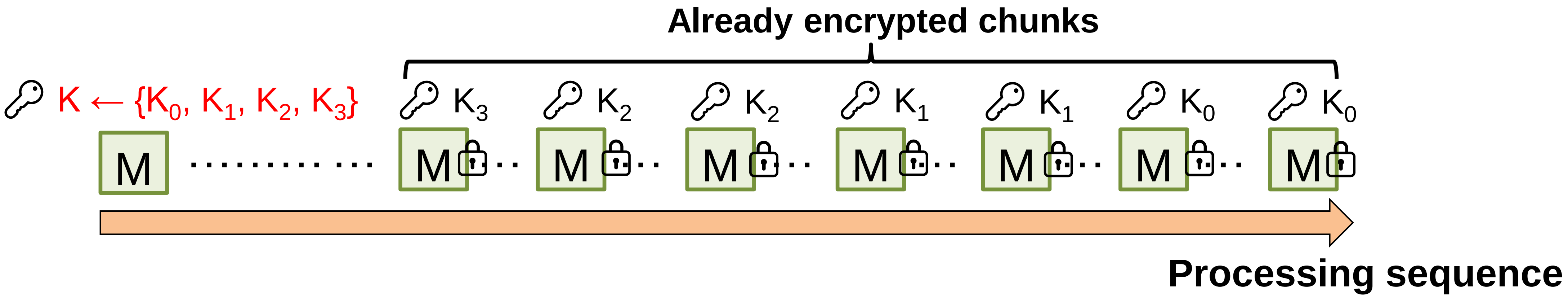
Sketch-based Frequency Counting



- Key manager estimates f via Count-Min Sketch [Cormode 2005]
  - Fixed memory usage with provable error bounds
- Client sends **short hashes** {$\mathbf{H}_i$(M)} to key manager
  - Key manager cannot readily infer M from short hashes

Probabilistic Key Generation

- Selects K uniformly from candidate keys derived from 0, 1,…, $\lfloor f/t \rfloor$
  - Enables probabilistic encryption on identical files
  - **Maintains deduplication effectiveness**
    - **Reason**: f is cumulative; keys derived from 0, 1,…, $\lfloor f/t \rfloor$-1 have been used to encrypt some old copies of M



Automated Parameter Configuration

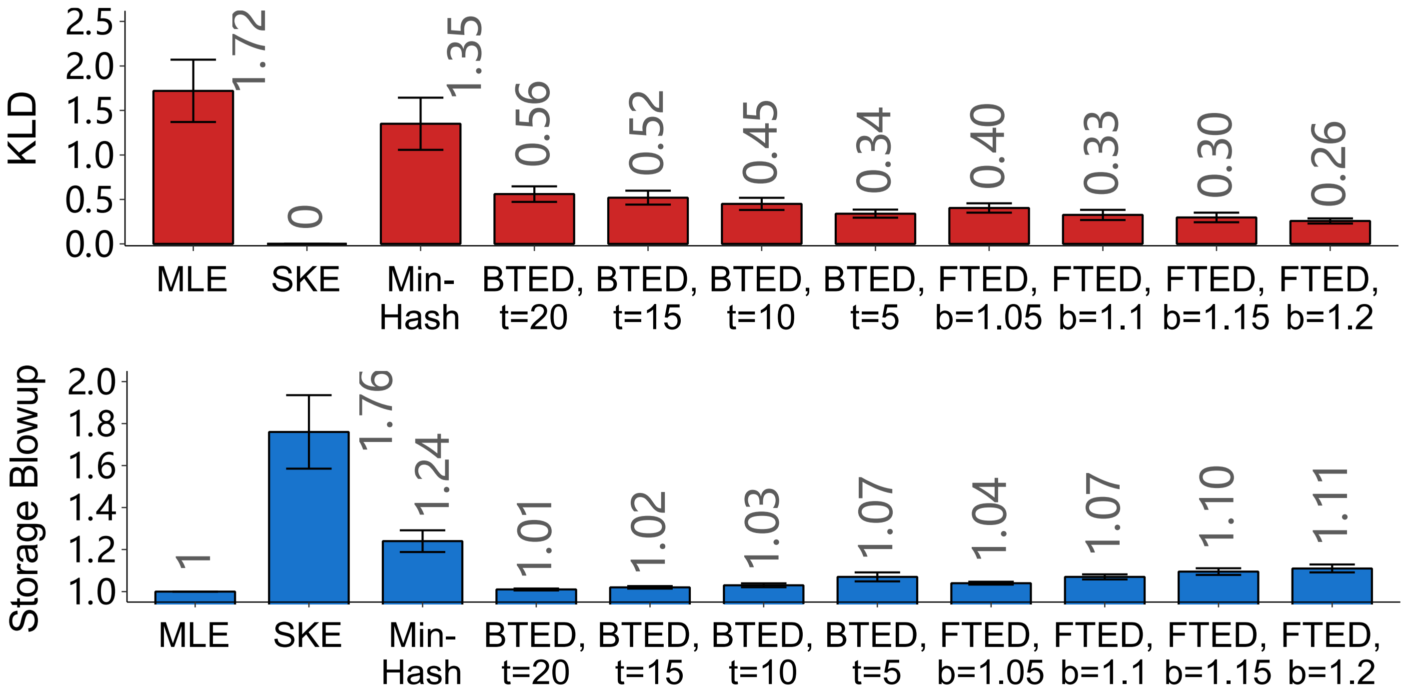
- Configure t by **solving optimization problem**, given:
  - Frequency distribution for a batch of plaintext chunks
  - Affordable storage blowup b over exact deduplication

- Goal: **minimize frequency leakage**
  - Quantify frequency leakage by Kullback-Leibler distance (KLD)
    - KLD: relative entropy to uniform distribution
  - A lower KLD implies higher robustness against frequency analysis
  - Configure t from the returned optimal frequency distribution of ciphertext chunks

Evaluation

- TEDStore realizes TED in encrypted deduplication storage
  - ~4.5K line of C++ code in Linux
- Trace analysis
  - FSL: file system snapshots (42 backups; 3.08TB raw data)
  - MS: windows file system snapshots (30 backups; 3.91TB raw data)
- Prototype experiments
  - Local 10 GbE cluster

Trade-off Analysis (FSL Dataset)



- Schemes
  - MLE
  - SKE
  - MinHash [Li et al., DSN'17]
  - Basic TED (varying t)
  - Full TED (varying b)

- Basic TED and Full TED effectively balance trade-off
- Full TED readily configures actual storage blowup

Prototype Experiments

Steps	Fast (MD5, AES-128)	Secure (SHA-256, AES-256)
Chunking	0.8ms	
Fingerprinting	1.7ms	2.6ms
Hashing	0.4ms	
Key Seeding	0.01ms	0.04ms
Key Derivation	0.07ms	0.1ms
Encryption	3.7ms	4.9ms

TED operations (Hashing, Key Seeding, Key Derivation)

Computational time per 1MB of uploads

- ➢ TED incurs limited overhead (7.2% for Fast; 6.1% for Secure)
- ➢ More results in paper:
  - TED achieves ~**30X** key generation speedup over existing approaches
  - Multi-client upload/download performance

Conclusion

- TED: encrypted deduplication primitive that enables controllable trade-off between storage efficiency and data confidentiality
  - Sketch-based frequency counting
  - Probabilistic key generation
  - Automated parameter configuration

- Source code: **http://adslab.cse.cuhk.edu.hk/software/ted**