SecDep: A User-Aware Efficient Fine-Grained Secure Deduplication Scheme with Multi-Level Key Management

Yukun Zhou, Dan Feng, Wen Xia, Min Fu,
Fangting Huang, Yucheng Zhang, Chunguang Li

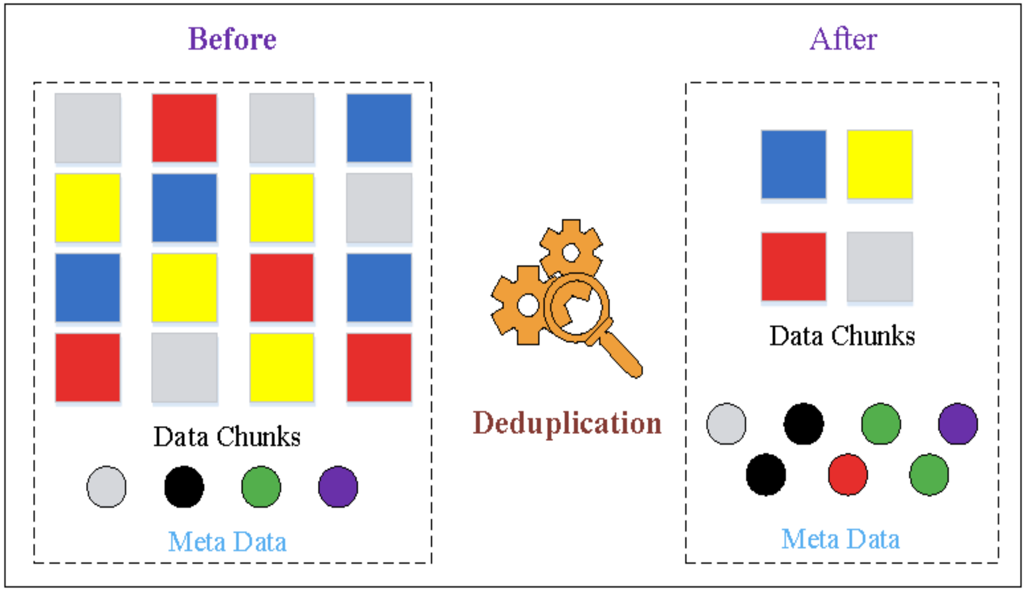
Wuhan National Laboratory for Optoelectronics
School of computer, Huazhong University of Science and Technology

Background

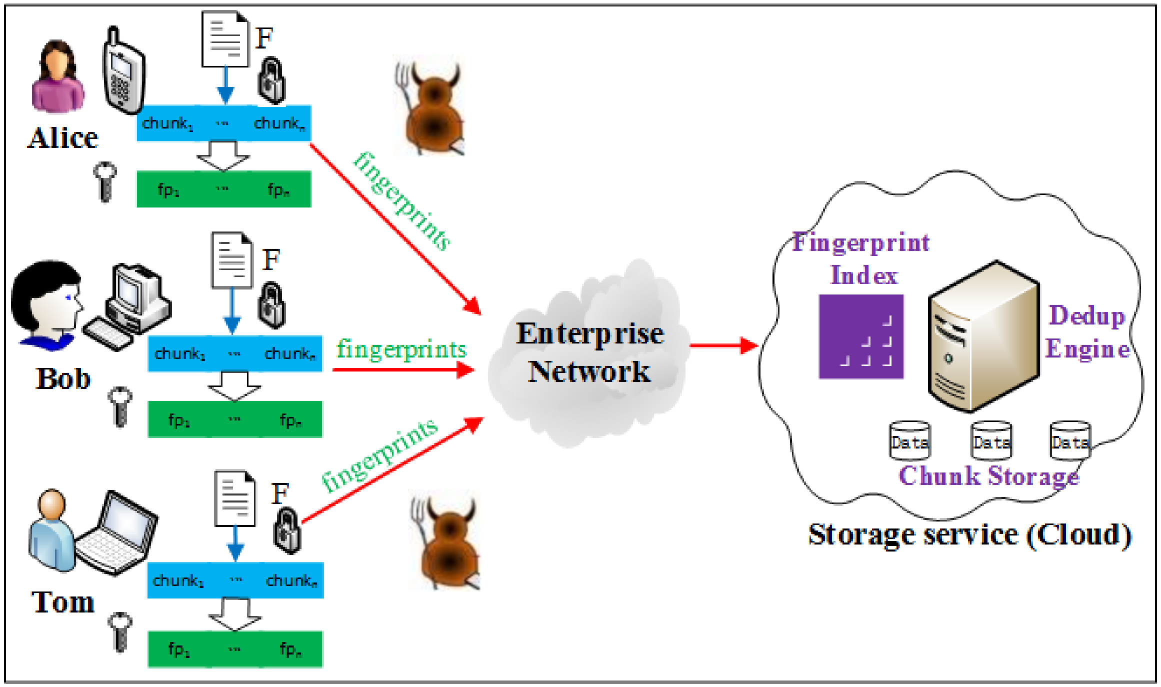
- In big data era
  - We had 4.4 ZB of data in 2013, and expectedly to grow by 10-fold in 2020 [*IDC'14*]
- Why data deduplication?
  - Faster due to file-level and chunk-level compression
  - Higher compression ratio since larger scope. (the entire system vs. a limited compression window)
  - Space- and bandwidth-efficient

Workflow of data deduplication

- Data deduplication is a scalable compression technology
  - Keep only one physical copy
- Key steps of data deduplication
  - Chunking
  - Hashing
  - Indexing (Dedup)
  - Writing (Store)



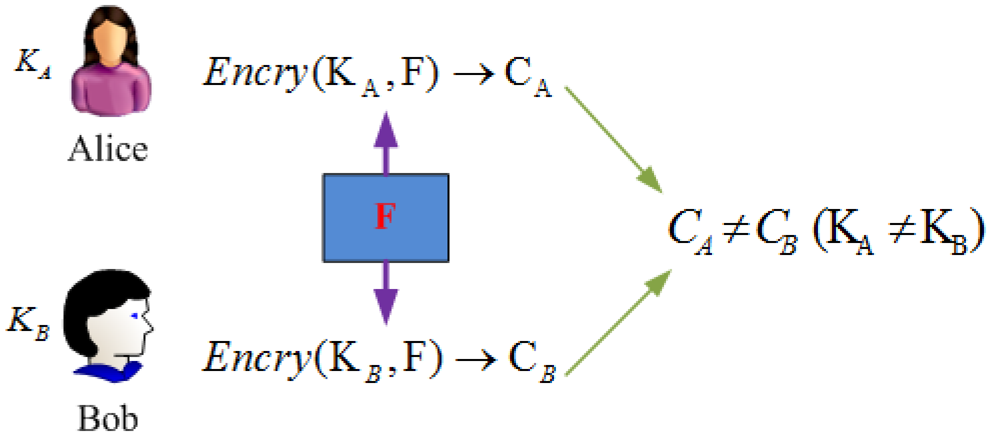
Our design goal



- Data confidentiality
  - Customers want to encrypt and protect their data, while using deduplication to save storage space
- Key management
  - Key security
  - Large key space overheads for fine-grained deduplication

The problem between dedup and encryption

- Encrypting data with users' own keys

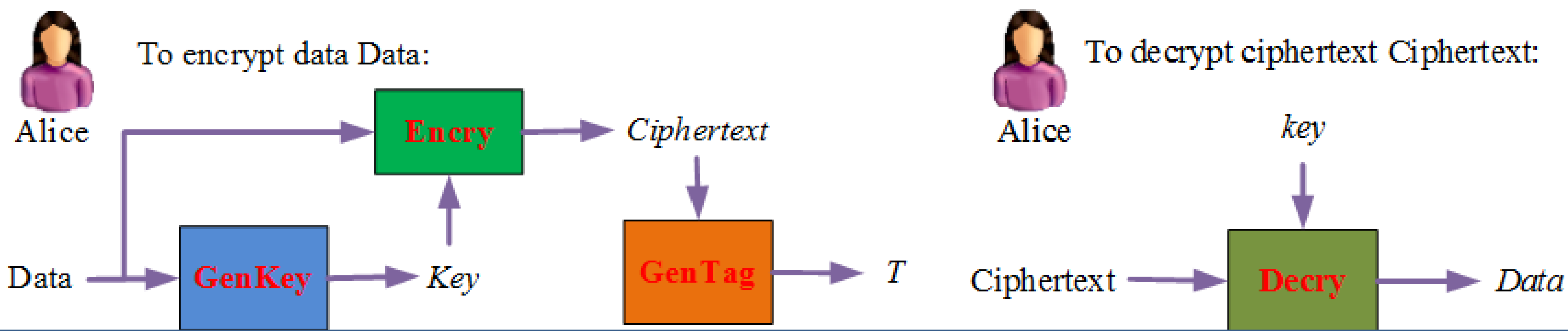


Deduplication does not work

- Sharing keys or "secret" among users
  - **Large computation (time) overheads**
  - **User compromise**. All data is insecure if one user is compromised

State of the art

1 CE: Convergent Encryption *(Farsite@ICDCS'02, MLE@Europcrypt'13)*



The main reason why CE suffers brute-force attacks is deterministic and keyless.

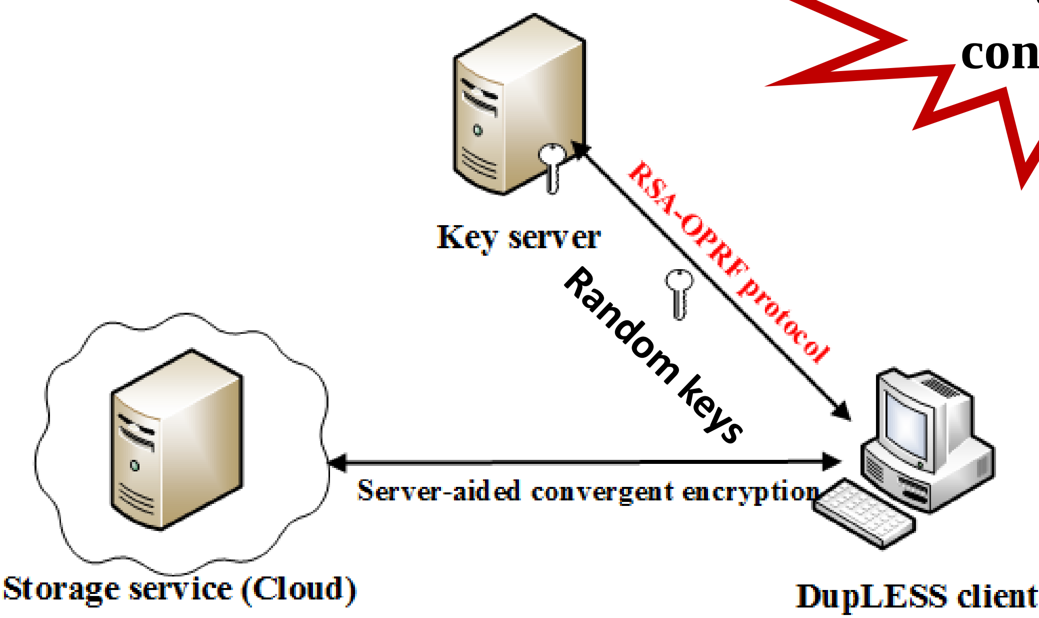
Brute-force attacks *(DupLESS@USENIX Security'13)*

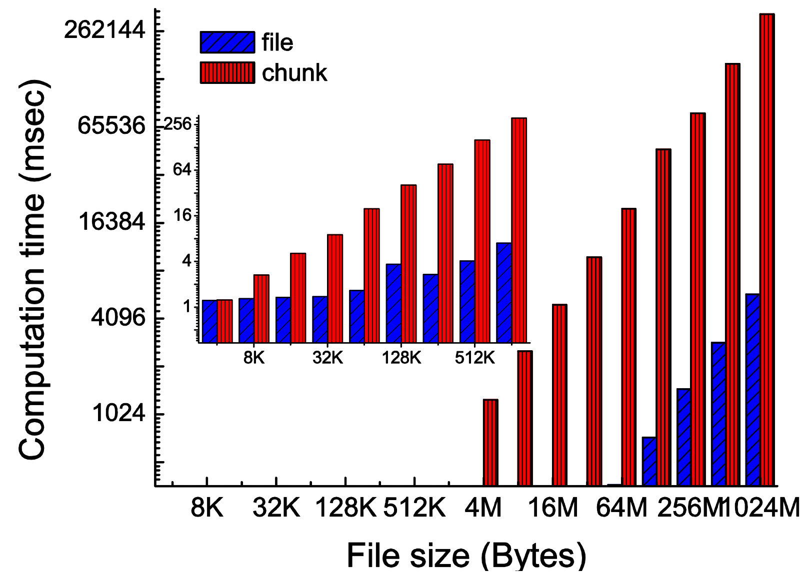
If D comes from $S = \{D_1, \ldots, D_n\}$ The adversary knows the ciphertext C of D can recover D from $C \leftarrow Encry(\text{GenKey}(D), D)$	For $D_i \in S$ $C_i \leftarrow Encry(\text{GenKey}(D_i), D_i)$ if $C_i = C$, then return D_i

State of the art

2 DupLESS: Server-aided CE *(DupLESS@USENIX Security'13)*

time consuming





DupLESS is not suitable for fine-grained deduplication.

State of the art

1 Master Key *(Pastiche@SOSP'02, DupLESS@USENIX Security'13)*

- Encrypting chunk-level CE keys with users' master key

- **The master key suffers from single point of failure**
- **Key space overheads will increase with the number of sharing users**

2 Dekey *(Dekey@TPDS'14)*

- Protecting chunk-level CE keys by splitting them into key shares via RSSS [Ramp@CRYPTO'84]

- **Large key space overheads for fine-grained deduplication due to storage blowup of secret sharing scheme**

A summary of state of the art

Security goals	Approaches	Granularity	Limitations
Data Confidentiality	CE	File & Chunk	Brute-force attacks
	HCE	File & Chunk	Brute-force attacks, duplicate-faking attacks
	DupLESS	File	Large computation overheads (chunk-level)
Key Security	Single Key Server	File & Chunk	Single-point-of-failure
	Master Key	File & Chunk	Key space overheads
	Secret Splitting	Chunk	Key space overheads

- Main problems:
  - **Brute-force attacks**
  - **Large computation (time) overheads**
  - **Large key space overheads**

Observations

- Cross-user redundant data are mainly from the duplicate files. (*Similar with Sam@ICPP'10,microsoft@FAST'11*)

Data sets	Cross-user dup files	Inside-user dup chunks	Cross-user dup chunks
One-set (GB)	104.28 (51.6%)	78.62(38.9%)	19.2(9.5%)
Inc-set (GB)	108.8 (77%)	27.13(19.2%)	5.37(3.8%)
Full-set (GB)	2393.7 (97.4%)	60(2.4%)	1.95(0.02%)
FSLhomes (GB)	13764.7 (95.17%)	687(4.85%)	11.6(0.08%)

- Cross-user and inside-user deduplication schemes face different security challenges
  - Inside-user deduplication could ensure security easily
  - Cross-user deduplication need a more secure approach

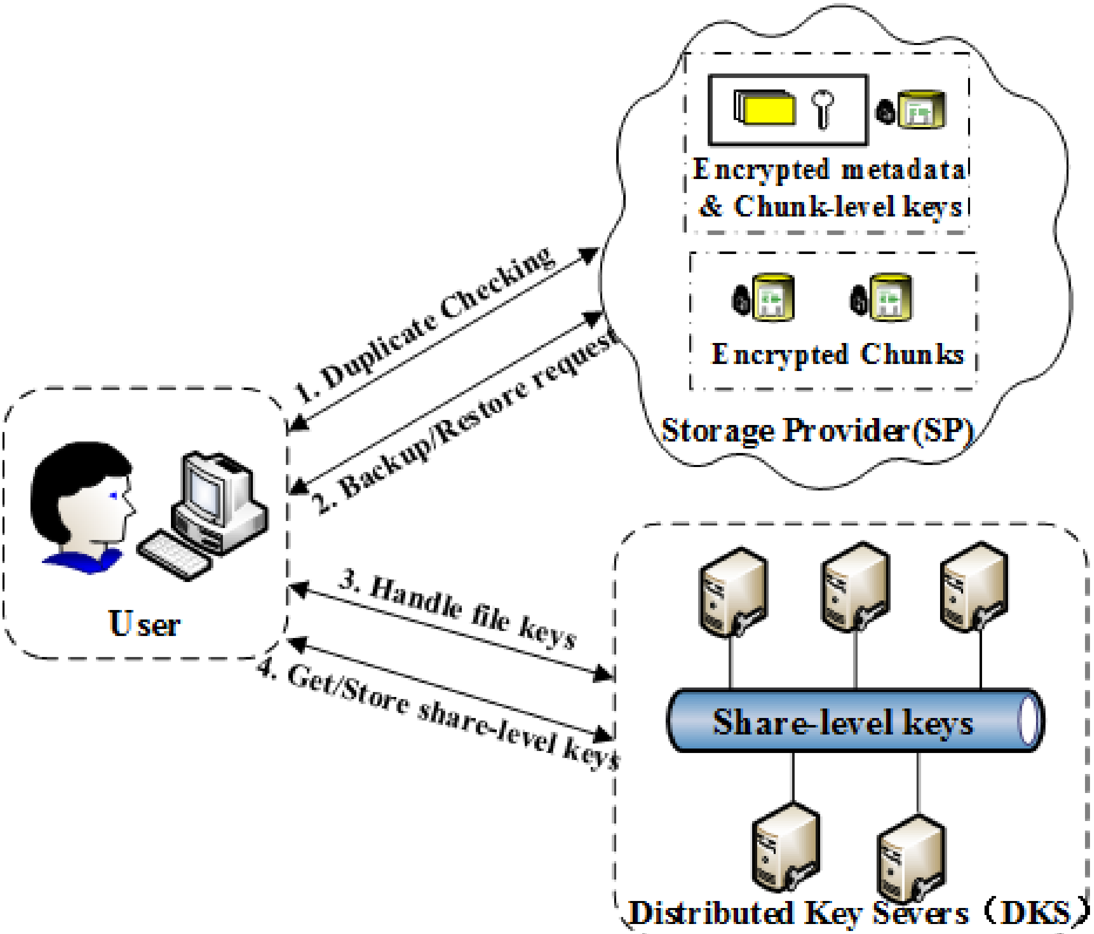
Motivations

Dedup Scheme	High dedup factor	Low security time overheads
Global dedup at chunk-level	√	×
Cross-user dedup at file-level	×	√
Inside-user dedup at chunk-level	×	√
SecDep	√	√

- **SecDep:** Combining cross-user file-level and inside-user chunk-level secure deduplication. Employing different secure policies to make a trade-off between security and deduplication performance.

- Overview of SecDep
- User-Aware Convergent Encryption (UACE)
- Multi-Level Key management (MLK)
- Evaluation
- Conclusion

Overview of SecDep



SP: file-level index

chunk-level index

chunk storage

DKS: Handle file-level keys

Store share-level keys

- Connections are protected by password or credentials.
- Data stored on SP and DKS are protected via access control.

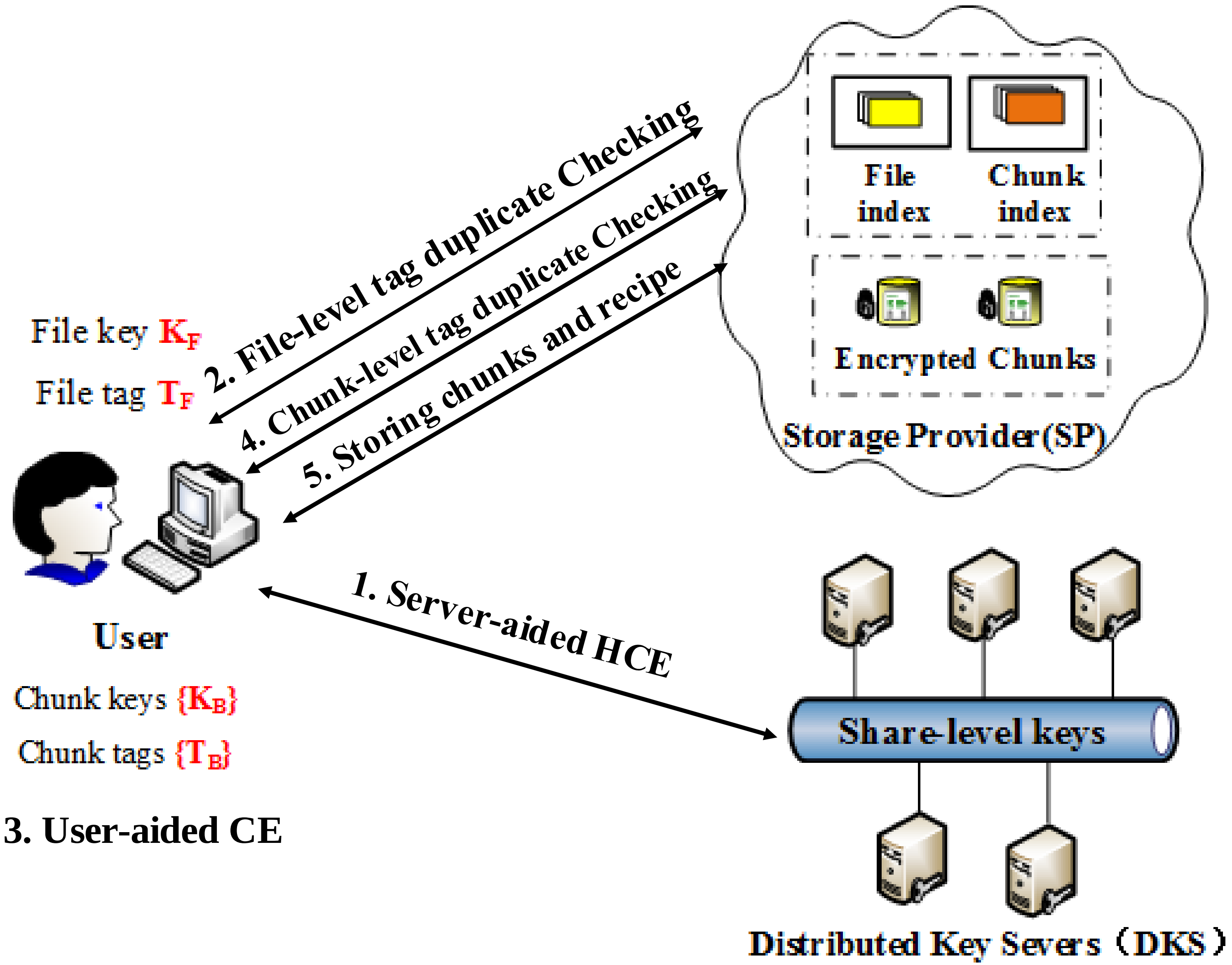
Design Goals of SecDep

- Data confidentiality
  - User-Aware Convergent Encryption(UACE)
    - **Cross-user file-level** **Sever-Aided HCE**
      - ✓ **Resisting brute-force attacks with higher security policy**
    - **Inside-user chunk-level** **User-Aided CE**
      - ✓ **Resisting brute-force attacks with lower security overhead**
- Key security
  - Multi-Level Key management(MLK)
    - Using file-level key to encrypt chunk-level keys
      - ✓ **Avoiding key space increasing with the number of sharing users**
    - Using Shamir Secret Sharing to protect file-level key
      - ✓ **Ensuring security and reliability of file-level key**

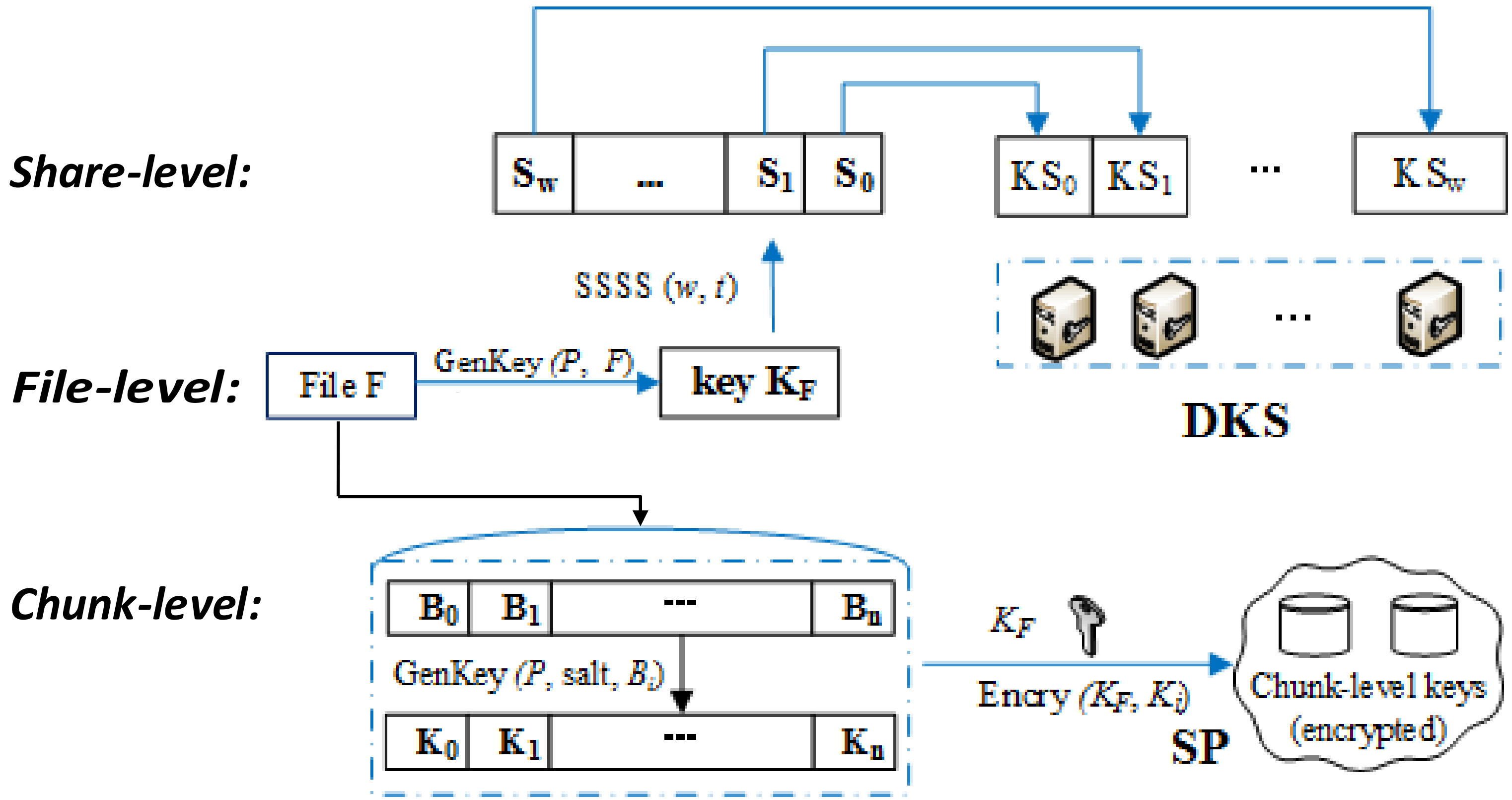
User-Aware Convergent Encryption(UACE)

- What's User-Aware?
  - Exploring variants of Convergent Encryption based on **users' data distribution and attributes**.
  - Variants of Convergent Encryption consists of **cross-user file level hash convergent encryption** and **inside-user chunk-level convergent encryption**.
- Cross-user file-level hash convergent encryption (**Server-aided HCE**)
  - Generating **random file-level keys** aided by the key server to enable global deduplication and encryption. Resist brute-force attacks at file-level.
- Inside-user chunk-level convergent encryption (**User-aided CE**)
  - Employing "secret" to **make chunk-level keys random**.
  - Resist brute-force attacks at chunk-level and reduce computation (time) overheads for fine-grained deduplication.

User-Aware Convergent Encryption



Multi-Level Key Management (MLK)



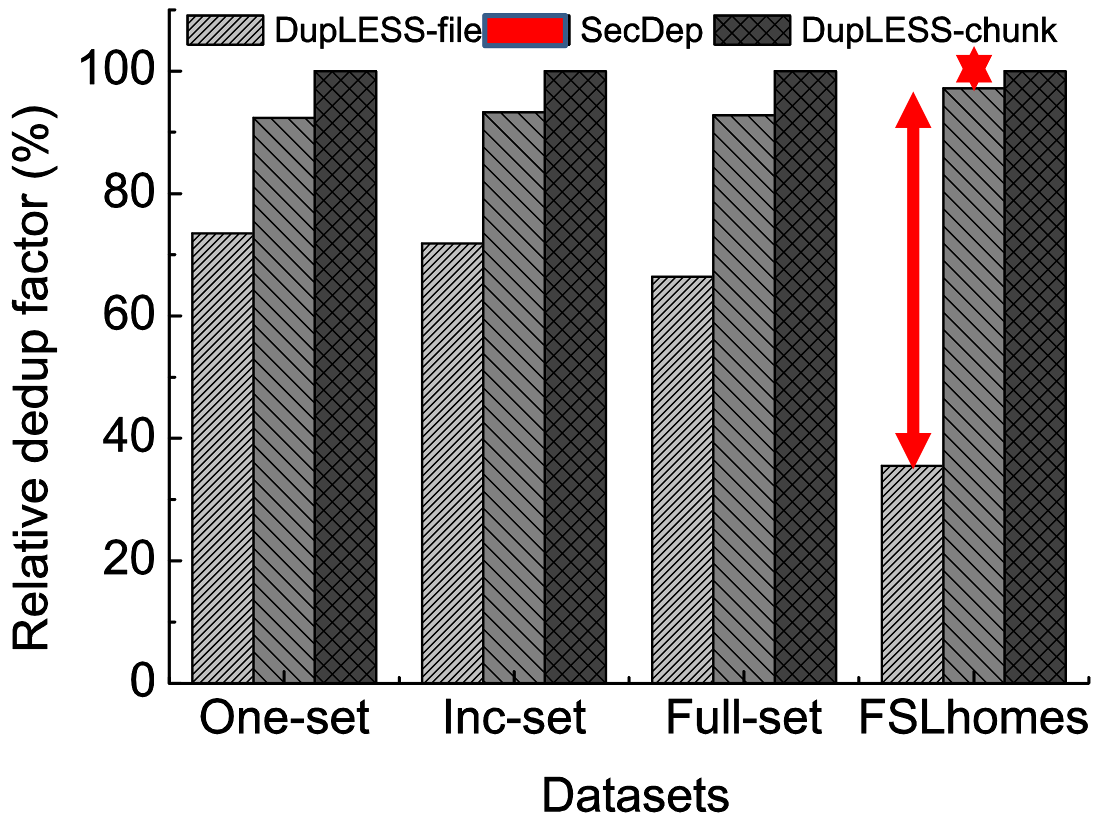
- **Using file-level key to manage chunk-level keys**
  - avoiding key space overheads increasing with the number of sharing users
- **Using secret sharing scheme to ensure security and reliability of file-level key**

Performance evaluation

- Metrics
  - Dedup Factor $DF = \frac{\text{the size of data before deduplication}}{\text{the size of data after deduplication}}$
  - Backup time
  - Key space overhead
- Datasets

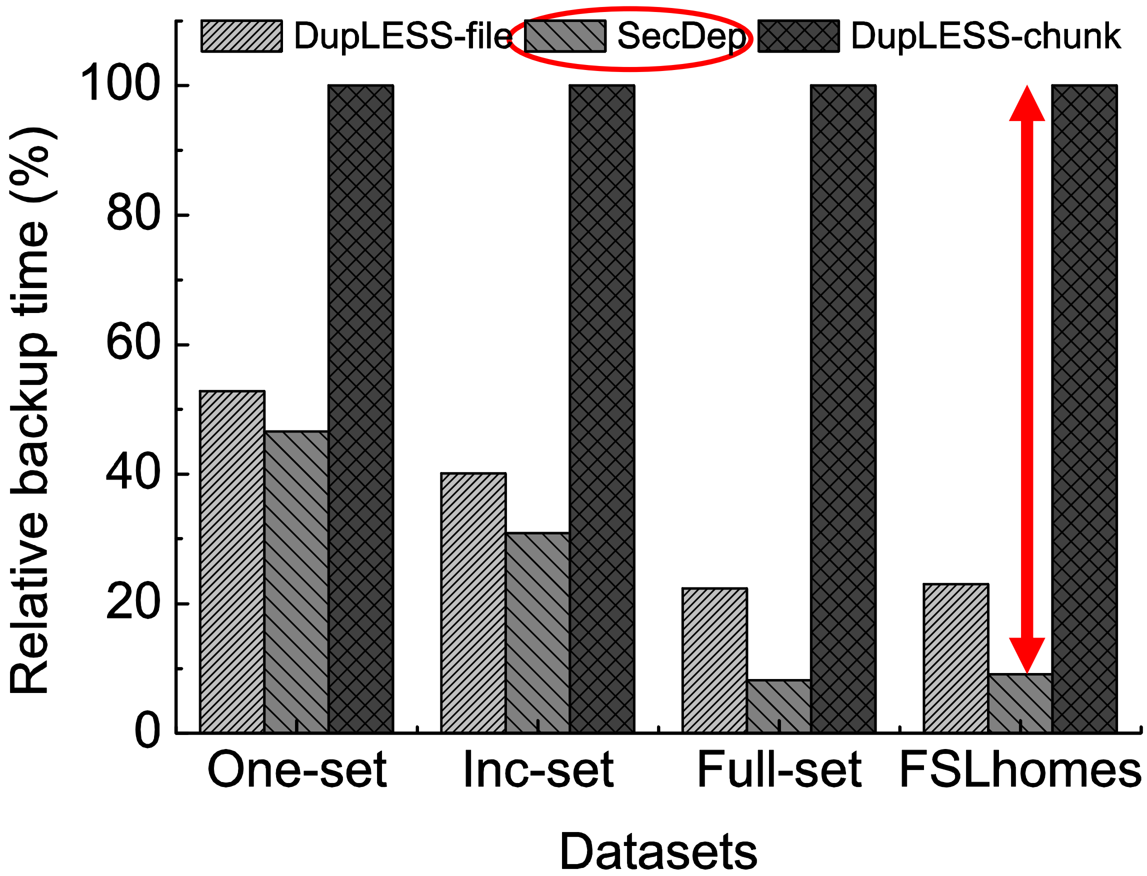
Characteristics	One-set	Inc-set	Full-set	FSLhomes
Number of users	11	6	19	7
Total size	491GB	224.4GB	2.5TB	14.5TB
Total files	2.5M	0.59M	11.3M	64.6M
Total chunks	50.5M	29.4M	417M	1703.3M
Avg. chunk size	10KB	8KB	6.5KB	8KB
Dedup factor	1.7	2.7	25	38.6

Deduplication factor



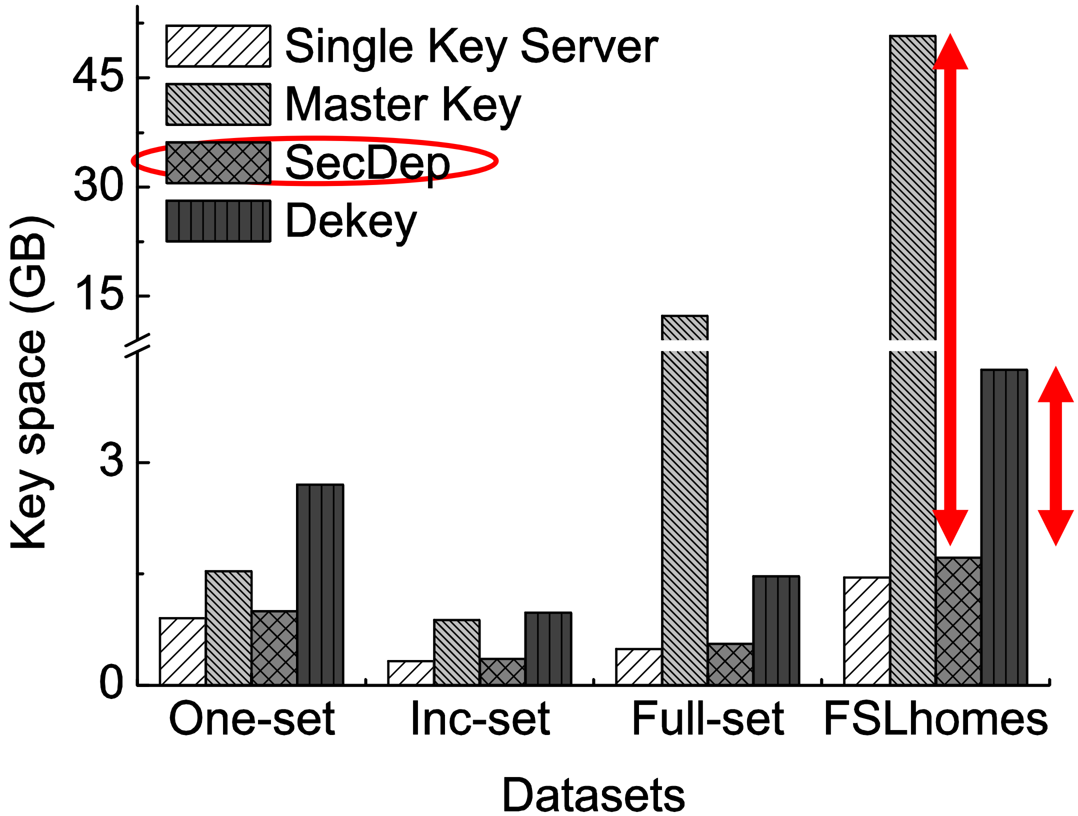
SecDep eliminates the majority of duplicate data, only resulting in a **2.8-7.35% loss of dedup factor** compared with the DupLESS-chunk.

Computation (time) overheads



- SecDep reduces **52-92%** of backup time overheads compared with DupLESS-chunk.

Key space overheads



RSSS(*w, t, r*) : *6, 4, 2*

SSSS(w, t)： *6, 4*

- SecDep reduces **59.5-63.6% and 34.8-96.6%** of key space overheads on the four real-world datasets compared with Dekey and Master Key approach respectively.

Conclusions

- We propose SecDep to ensure data and key security, which is a cross-user fine-grained deduplication-based system for cloud backups.
  - SecDep proposes UACE to resist brute-force attacks and reduce computation (time) overheads.
  - SecDep proposes MLK to ensure key security and reduce key space overheads.
- Our experiment results based on real-world datasets show that SecDep is more time-efficient and key-space-efficient than the state-of-the-art secure deduplication approaches.

Thank You !

Questions?

ykzhou@hust.edu.cn