


VIRGINIA TECH

Hermes: Efficient and Secure Multi-Writer Encrypted Database






Tung Le, Thang Hoang

{tungle, thanghoang}@vt.edu

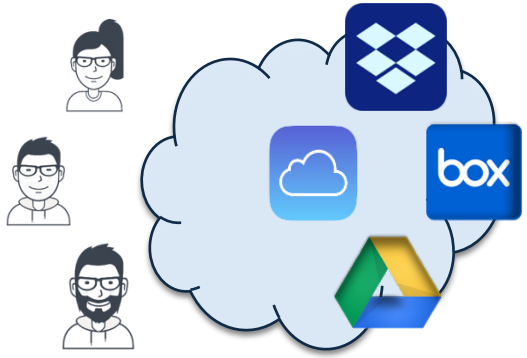
Virginia Tech

IEEE Symposium on Security and Privacy 2025

San Francisco, CA

Storing/Collecting Data with Cloud

- Cloud services reduce the burden of locally-stored ever-growing data

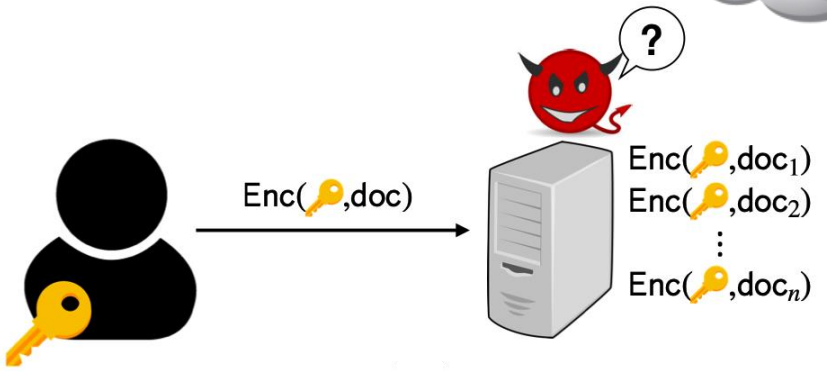


Storage-as-a service (STaaS)

- They also provide a platform for multi-writer applications:
  - Email/messaging

  - EHR systems

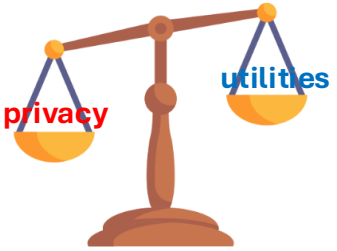
  - Collaborative applications



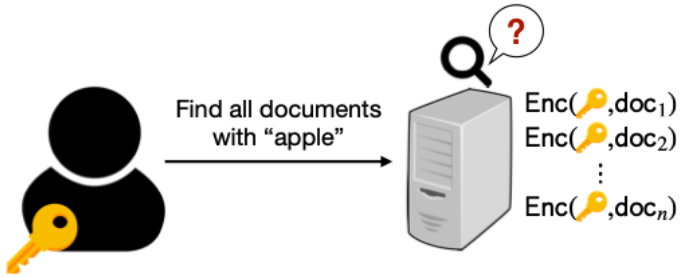
- Yet, it is doubtful whether the cloud server can be trusted. How to secure outsourced data while enabling efficient search?

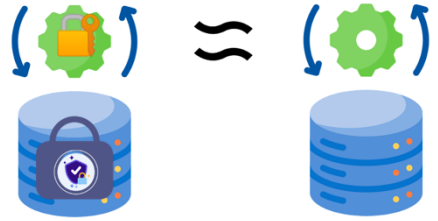
Searching Encrypted Data





- Standard encryption prevents any data querying/search

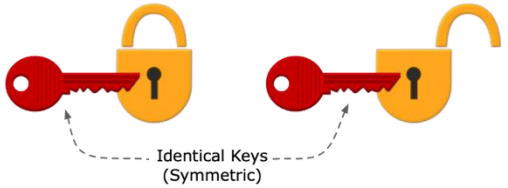



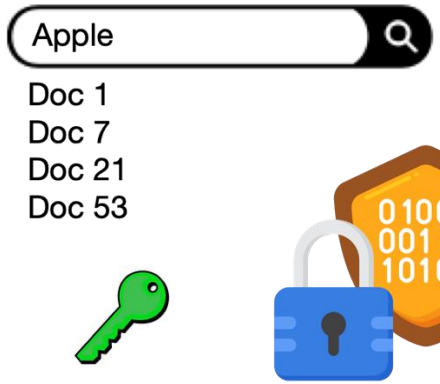


- Multi-Client ORAM or FHE are too heavyweight



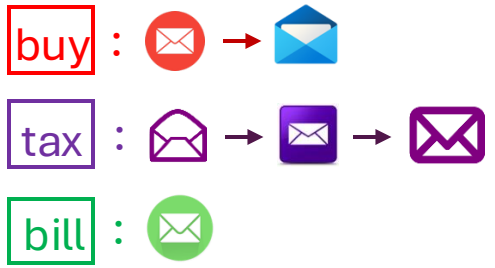

- **Searchable Encryption** (SE) is promising
  - Realize nice security-efficiency trade-offs for searching encrypted data
- Two well-studied SE notions



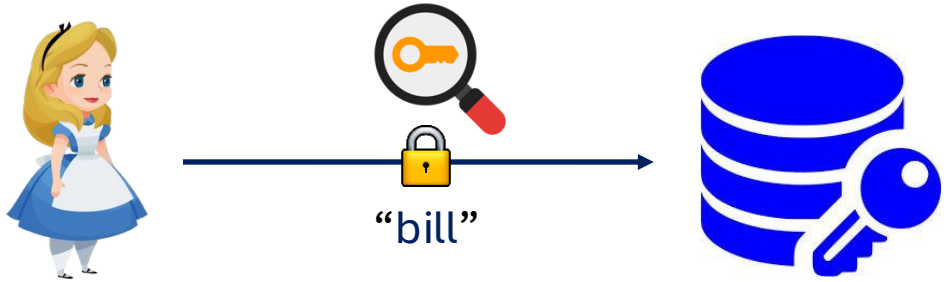
  - Dynamic Searchable Symmetric Encryption (**DSSE**) [KPR12]
  - Public-Key Searchable Encryption (**PKSE**) [BDOP04]

Representative SE Notions: DSSE



Sublinear Search Complexity

- Encrypted index allows efficient search



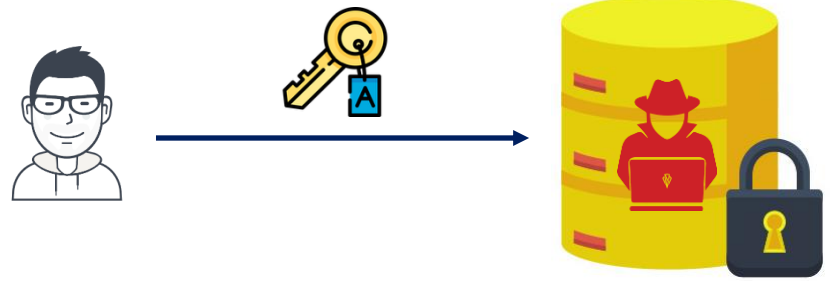
Ensure Forward Privacy

- Protect privacy of future updates by *"changing"* update tokens

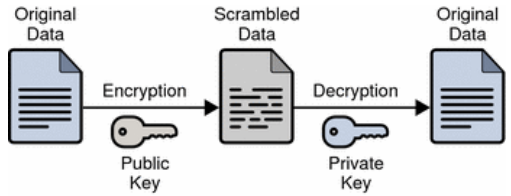
Mainly Support Single-User Settings

- Writer = Reader = Secret-Key Owner

- Fails for contributive applications

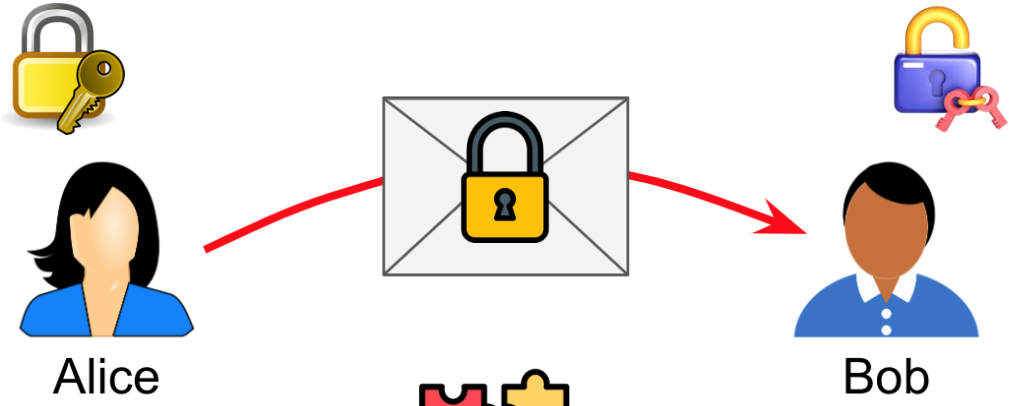


Representative SE Notions: PKSE

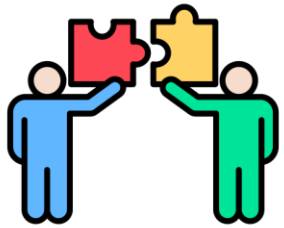


Multi-Writer Support

- No secret-key distribution
- No synchronous communication



Linear Search Complexity



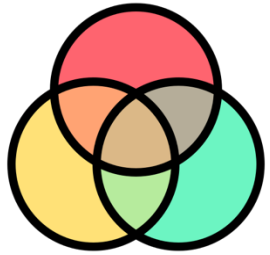
- Obstacle to build joint search index without shared secret and coordination between reader and writers

Lack of Forward Privacy

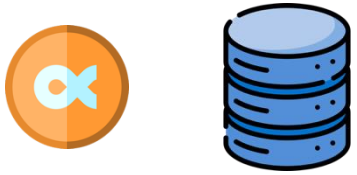
- Writers don't know any search states to change update tokens

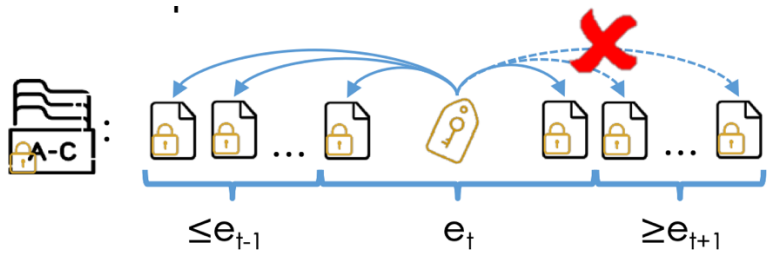
Towards the Best of Both Worlds: Hybrid SE

Advantages of Hybrid Searchable Encryption (HSE):



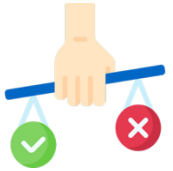
- **Sublinear Search Complexity:** $O(|W| + a_w)$
  - W: active keywords set, a_w: number of updates on target keyword w
  - Sublinear in database size



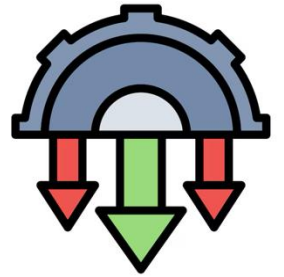


- **Forward Privacy:** Epoch-Based Approach
  - A search token is invalid for updates at future epochs

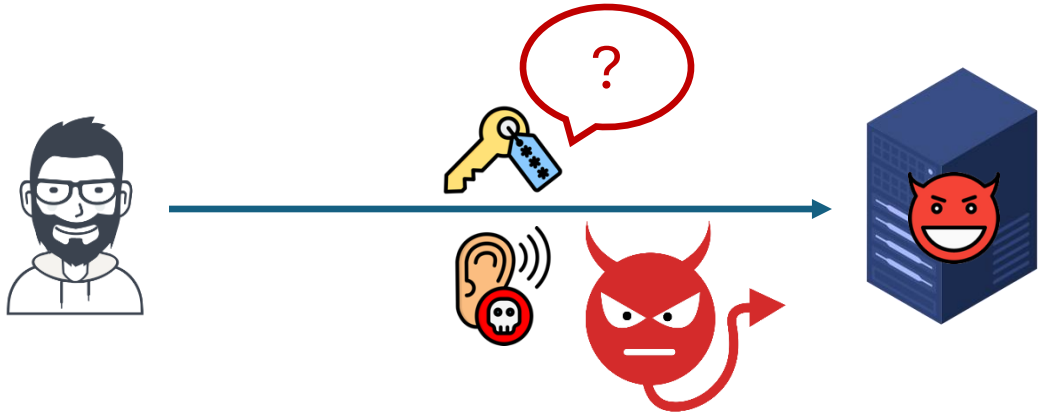
- **Multi-Writer Support:** ID-Coupling Key-Aggregate Encryption (ICKAE)
  - Only need reader public key to encrypt, only cloud server is online
  - Each writer independently updates, no synchronization with reader

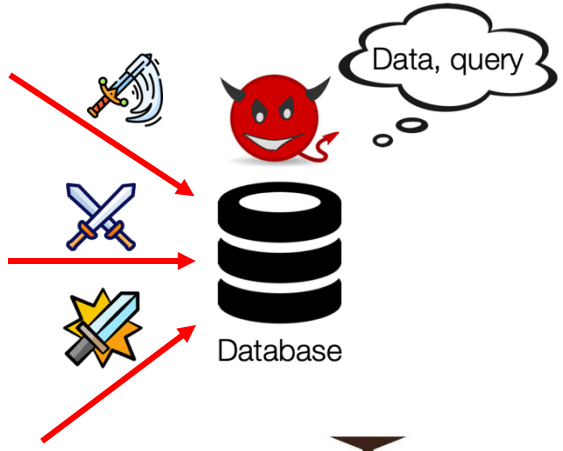


Shortcomings of State-of-the-Art HSE

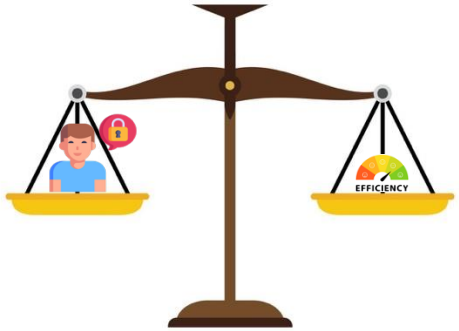


Vulnerable to **keyword-guessing** (dictionary) attacks

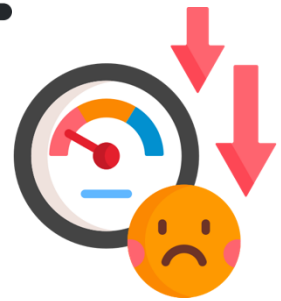




Search complexity is **linear** in keyword set size



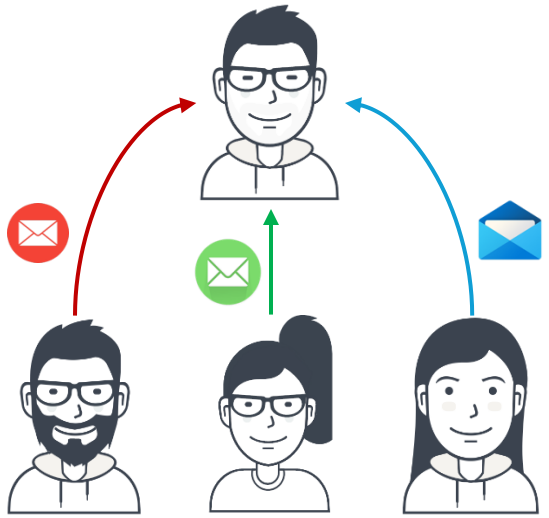
Significant user **overhead** to ensure forward privacy



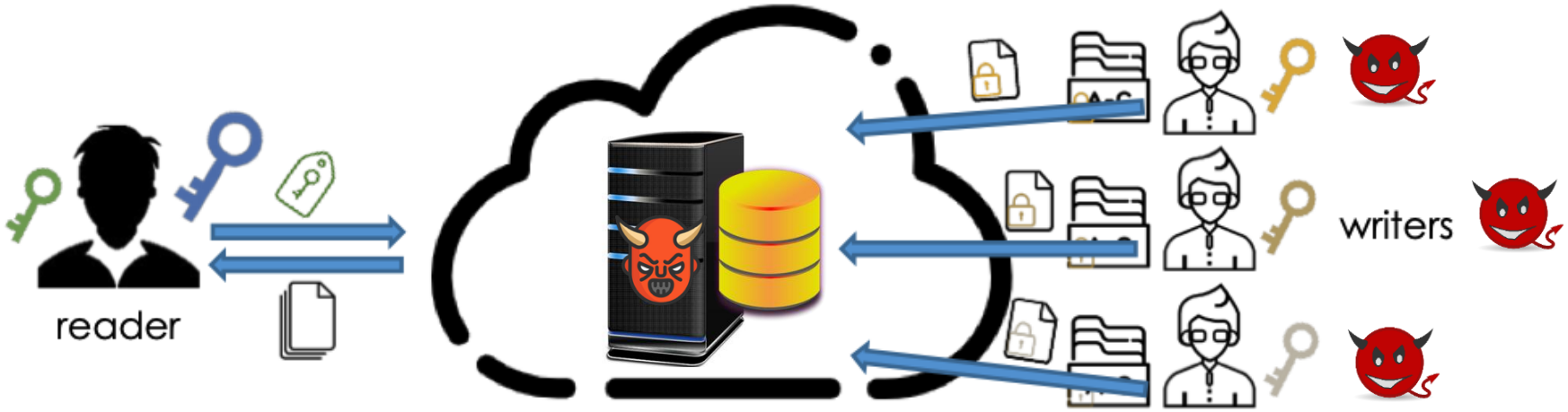
System, Theat and Security Models

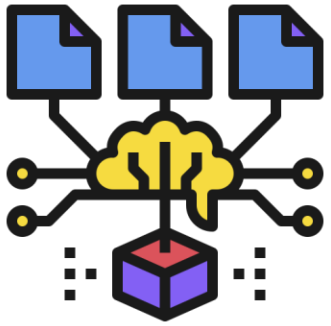
An *honest* reader who holds a public/private key pair

Multiple writers, where each owns its independent database



A server that stores database of writers and executes queries





The server and some of the writers can be corrupt. The adversary is semi-honest

Idea 1: Hidden-ID Key-Aggregate Encryption

Decryption in ICKAE [WM22]:

$$[u]_t = \underbrace{\sum_{j \in S} [\alpha^{n+1-j}]_1}_{\text{public}} . \overbrace{r . ([\gamma]_2 + [\alpha^i]_2)}^{\text{ciphertext}} - \left(\underbrace{\gamma \sum_{j \in S} [\alpha^{n+1-j}]_1 + \overbrace{\delta [h]_1}^{H(w)}}_{\text{aggregated key}} + \underbrace{\sum_{j \in S \setminus \{i\}} [\alpha^{n+1+i-j}]_1}_{\text{public}} \right) \overbrace{[r]_2}^{\text{ciphertext}}$$

correlation

- We design a new primitive called Hidden-ID Coupling KAE to prevent keyword-guessing attacks
- **High-level ideas:**

  - Each writer has a secret and embeds it into ciphertext

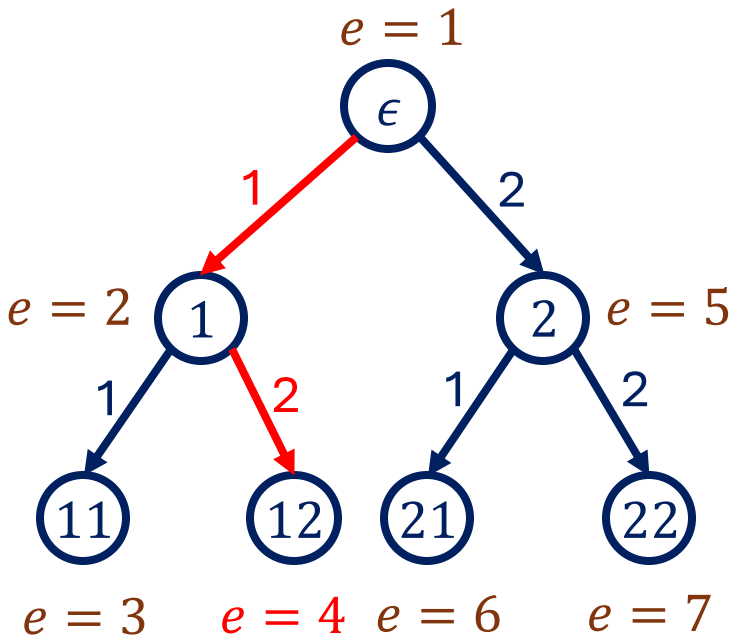
  → Prevent trying matching arbitrary IDs with given aggregated keys

  - Set up correlations between any two writers

  → Achieve key-aggregate property for confined search

- **Features**: confidentiality, anonymity, and *hidden-ID*

Idea 2: Forward Privacy via Epoch Encoding



Tree-based Epoch Encoding

$\longrightarrow$ $\mathbf{\Gamma_{t(e)}}$

Encryption/Ciphertext

$\longrightarrow$ $\mathbf{P_{(e)}}$

Decryption/Aggregated Key

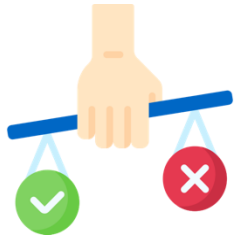
$\mathbf{\Gamma_\epsilon}$	$\mathbf{\Gamma_1}$	$\mathbf{\Gamma_{11}}$	$\mathbf{\Gamma_{12}}$	$\mathbf{\Gamma_2}$	$\mathbf{\Gamma_{21}}$	$\mathbf{\Gamma_{22}}$
$\{\epsilon\}$	{1, 2}	{11, 12, 2}	{12, 2}	**{2}**	**{21, 22}**	**{22}**

$\mathbf{\Gamma_{t(e)}}$: the smallest set of nodes that includes a prefix of all encoded epoch $\mathbf{t}(e')$ such that $e' \geq e$

Epoch $e = 4$ is encoded as $\mathbf{t}(4) = 12$, and $\mathbf{P}_{12} = \{\epsilon, 1, 12\}$

$\mathbf{\Gamma_\epsilon, \Gamma_1, \Gamma_{11}, \Gamma_{12}}$ (epochs $e' \leq 4$) all contain some value in $\mathbf{P}_{12}$

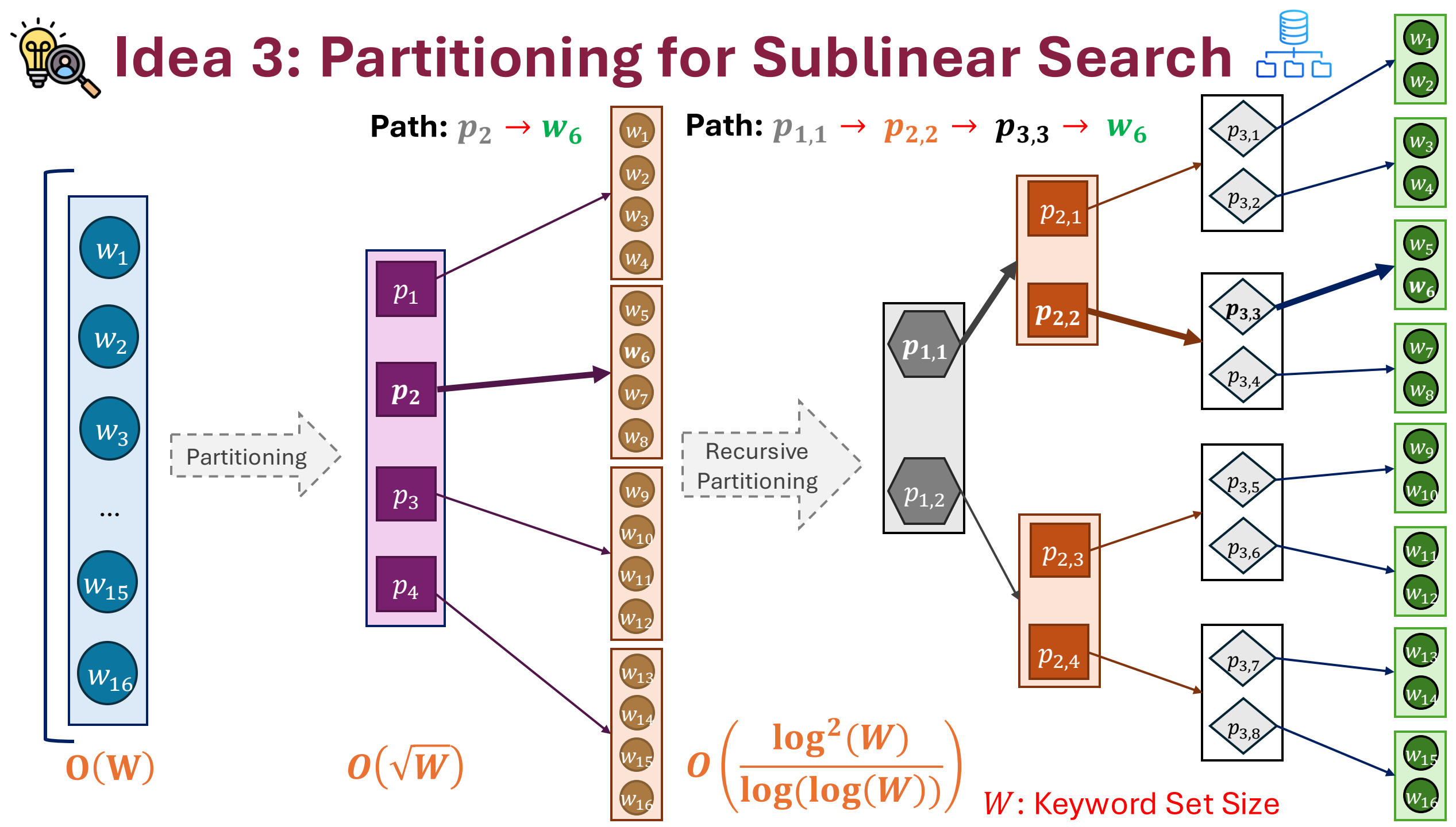
However, $\mathbf{\Gamma_2, \Gamma_{21}, \Gamma_{22}}$ (epochs $e' > 4$) *does not*



Forward Privacy with Efficiency

Trade-Off: An Increase $O(\lambda)$ in Update Complexity

Idea 3: Partitioning for Sublinear Search

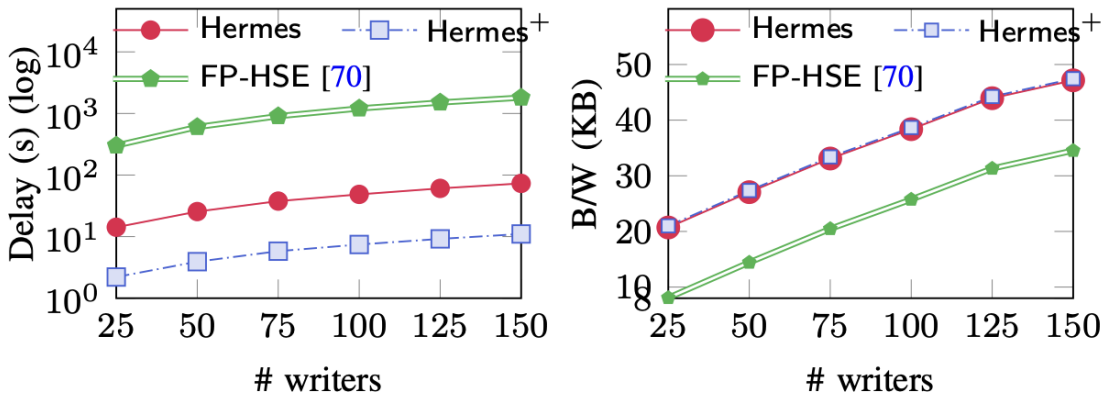


Experimental Evaluation

Keyword Search

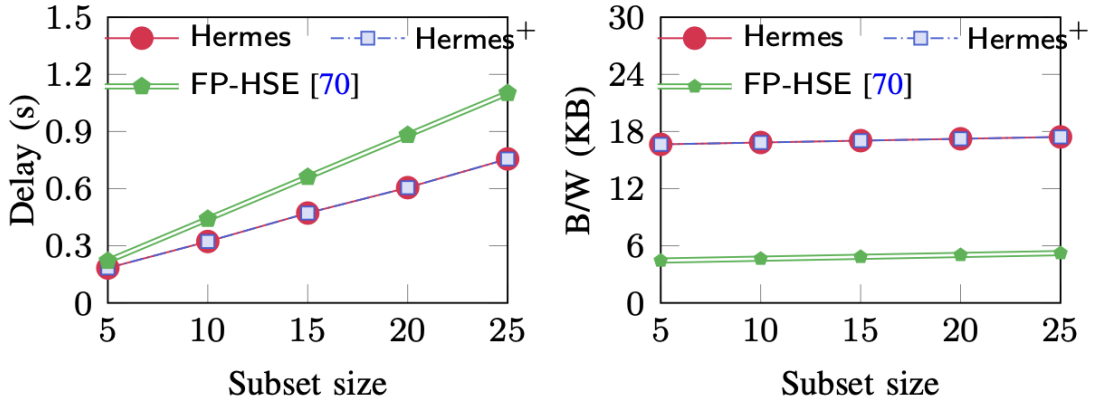
Hermes executes search up to **two orders of magnitude** faster than FP-HSE [WM22]

while **preventing keyword-guessing attacks**



(a) E2E delay **(b)** Reader's bandwidth

Figure 7: E2E keyword search delay and bandwidth (Enron).

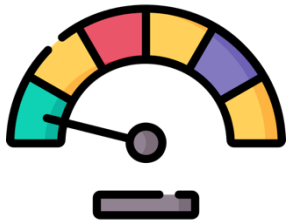


(a) E2E delay **(b)** Reader's bandwidth

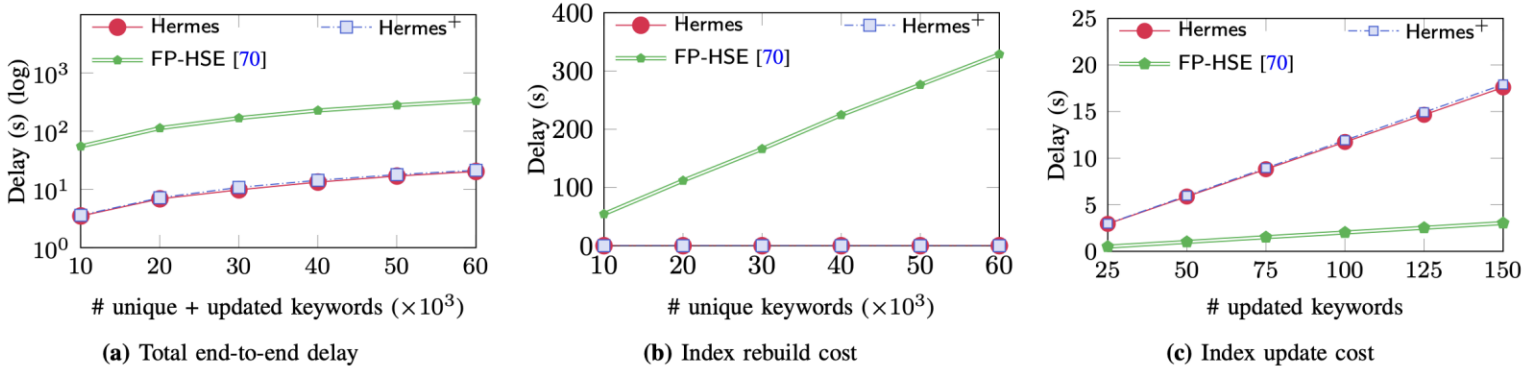
Figure 8: E2E keyword search delay and bandwidth (EHR).

Experimental Evaluation

Keyword Update

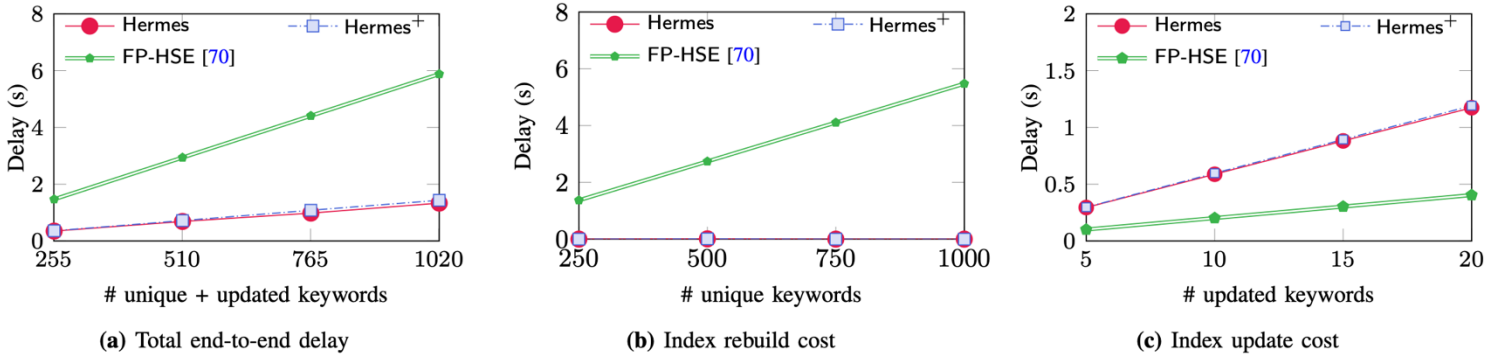


Hermes executes update **3.8× - 17.1×** faster than FP-HSE



(a) Total end-to-end delay (b) Index rebuild cost (c) Index update cost

Figure 10: Keyword update delay (Enron). The total update cost with forward privacy (FP) **(a)** equals the cost to rebuild the index **(b)** plus the cost to update the index **(c)**. Unlike FP-HSE, Hermes/Hermes$^+$ does not need a rebuild process to ensure FP of subsequent updates.



(a) Total end-to-end delay (b) Index rebuild cost (c) Index update cost

Figure 11: Keyword update delay (EHR).

Summary

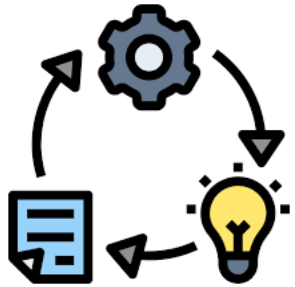




Our Hermes simultaneously achieves:


Security Against Keyword-Guessing Attacks

- A New Primitive: Hidden-ID Key-Aggregate Encryption



Sublinear Search Complexity (in Keyword Set Size)

- Partitioning/Recursive Partitioning

User-Efficient Forward Privacy

- Tree-based Epoch-Encoding Technique




THANK YOU FOR YOUR ATTENTION


References

[BDOP04] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. Cryptology ePrint Archive, Paper 2003/195, 2003.

[KPR12] Seny Kamara, Charalampos Papamanthou, and Tom Roeder. 2012. Dynamic searchable symmetric encryption. In Proceedings of the 2012 ACM conference on Computer and communications security (CCS '12).

[WM22] Jiafan Wang and Sherman S. M. Chow. Omnes pro uno: Practical Multi-Writer encrypted database. In 31st USENIX Security Symposium (USENIX Security 22), pages 2371–2388, Boston, MA, August 2022. USENIX Association.