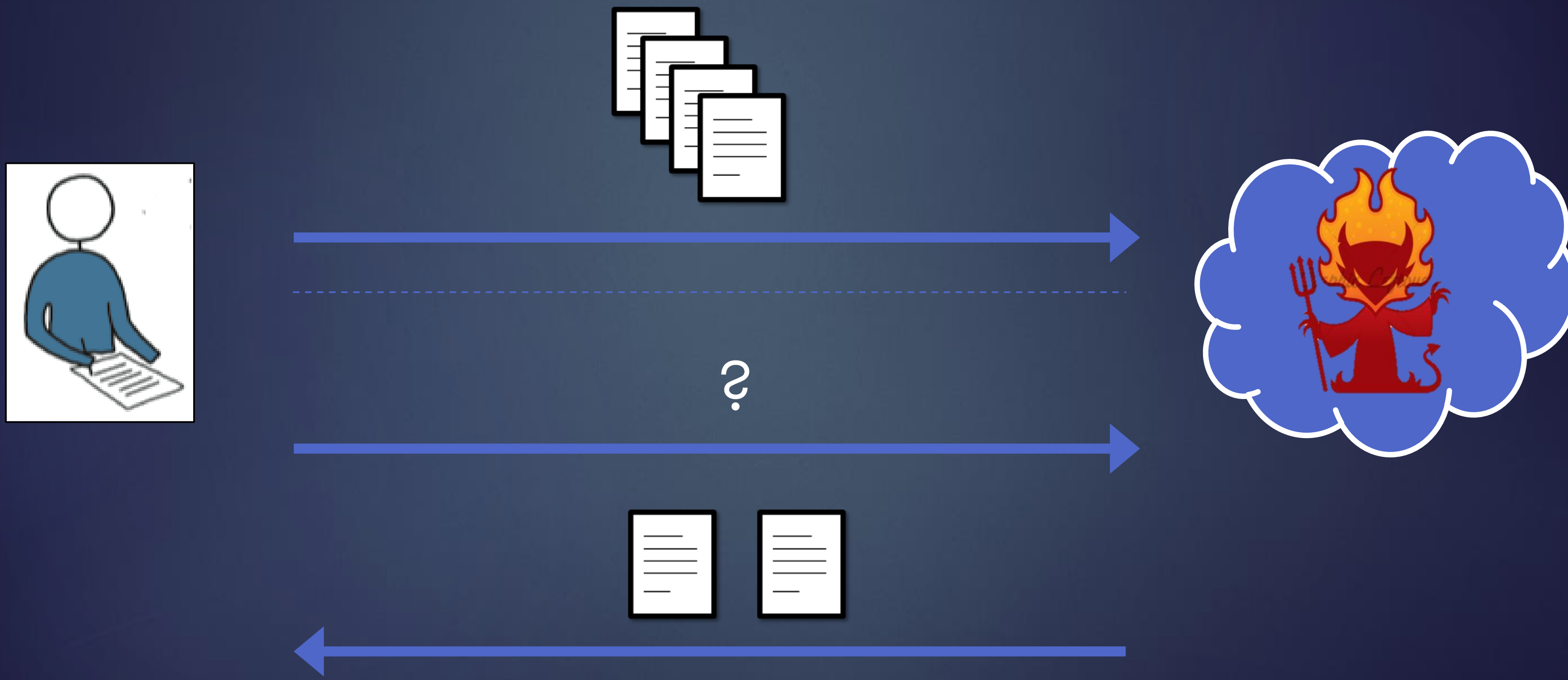
Parallel & Dynamic Searchable Symmetric Encryption

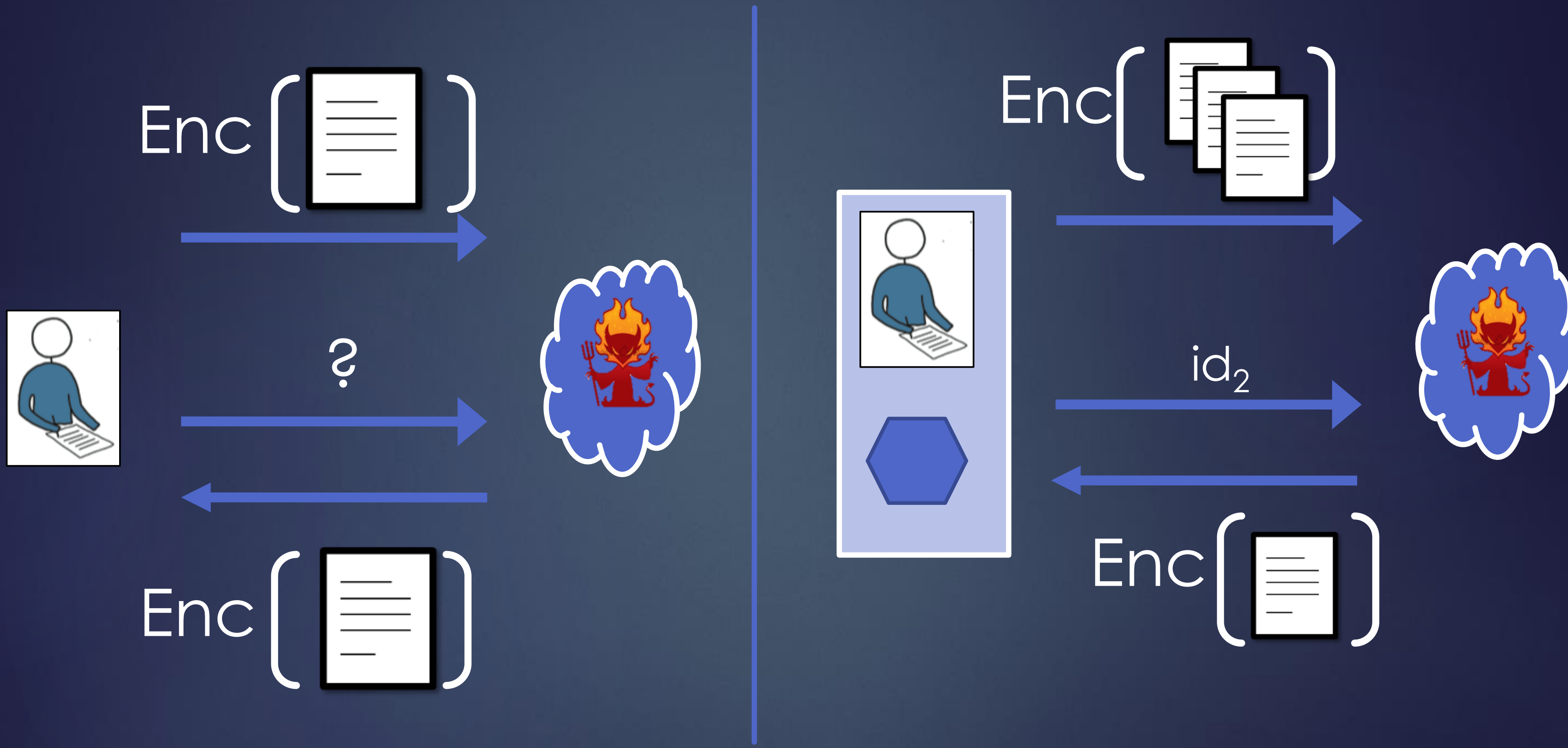
SENY KAMARA - MICROSOFT RESEARCH

CHARALAMPOS PAPAMANTHOU – UC BERKELEY

Cloud Storage

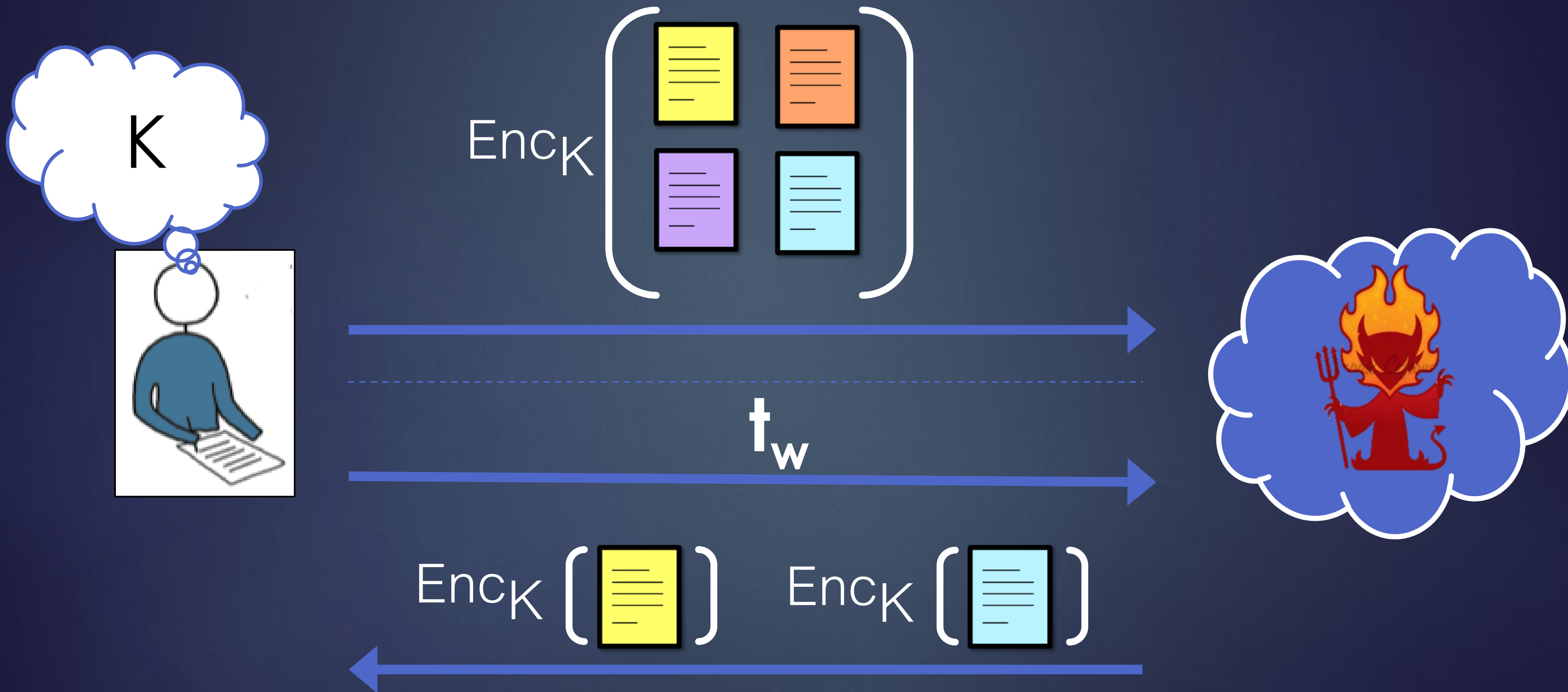


Two Simple Solutions to Search



Q: can we achieve the best of both?

Searchable Symm. Encryption



Security Definitions

- Security against chosen-keyword attack
  [Goh03, Chang-Mitzenmacher05, Curtmola-Garay-K.-Ostrovsky06]

> **CKA1:** "Protects files and keywords even if chosen by adversary"

- Security against *adaptive* chosen-keywords attacks
  [Curtmola-Garay-K.-Ostrovsky06]

> **CKA2:** "Protects files and keywords even if chosen by adversary, and even if chosen as a function of ciphertexts, index, and previous results"

Security Definitions

- UC [Kurosawa-Ohtaki12]
  - Universal composability [Canetti01]

UC: "Remains CKA2-secure even if composed arbitrarily"

CKA2-Security

[Curtmola-Garay-K.-Ostrovsky06]

- *Simulation*-based definition
  - ``given the encrypted index, encrypted files and search tokens, no adversary can learn any information about the files and the search keywords other than what can be deduced from the *access* and *search* patterns…"
  - "…even if queries are made adaptively"
  - access pattern: pointers to (encrypted) files that satisfy search query
  - search pattern: whether a search query is repeated

SSE Parameters

- Parameters
  - n: number of files in collection
  - |**f**|: size of file collection
  - m: number of keywords
- Client-side
  - Security: CKA1, CKA2, UC
  - Token size: O(1) to O(n)
- Server-side
  - Search time: OPT, O(n), O(|**f**|)

Searchable Symmetric Encryption

Scheme	Dynamism	Security	Search	Parallel
[SWP00]	No	CPA	O(|**f**|)	O(n/p)
[Goh03]	**Yes**	CKA1	O(n)	O(n/p)
[CM05]	No	CKA1	O(n)	O(n/p)
[CGKO06] #1	No	CKA1	**O(OPT)**	N/A
[CGKO06] #2	No	**CKA2**	**O(OPT)**	**N/A**
[CK10]	No	**CKA2**	**O(OPT)**	**N/A**
[vLSDHJ10]	**Yes**	**CKA2**	O(log m)	N/A
[KO12]	No	**UC**	O(n)	N/A
[KPR12]	**Yes**	**CKA2**	**O(OPT)**	**N/A**
[this work]	**Yes**	**CKA2**	**O(OPT·log(n))**	**$O(\frac{OPT}{p} \cdot \log(n))$**

Limitations of Inverted Index Approach

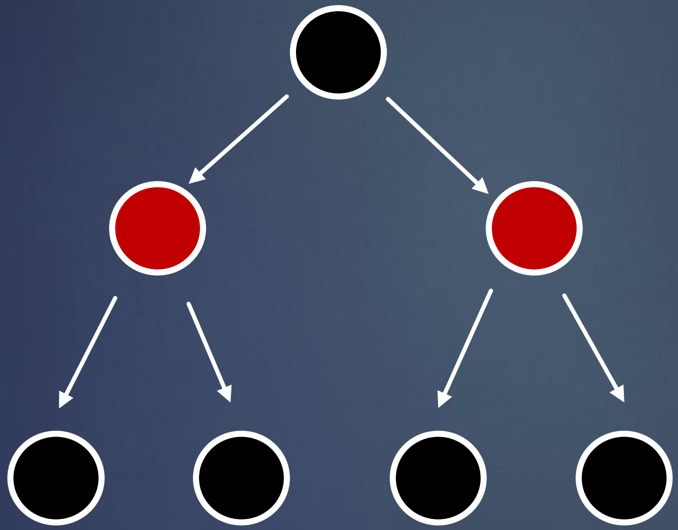
- Static
- Sequential
- [K.-Papamanthou-Roeder12]
  - ☺ Optimal search time
  - ☺ Handles updates
  - ☹ Overly complex
  - ☹ Sequential

A New Approach

Tree-Based Approach

- Advantages
  - Sub-linear
  - Dynamic
  - Parallelizable
  - Simple
- Disadvantages
  - not optimal
  - interactive updates

Red-Black Trees

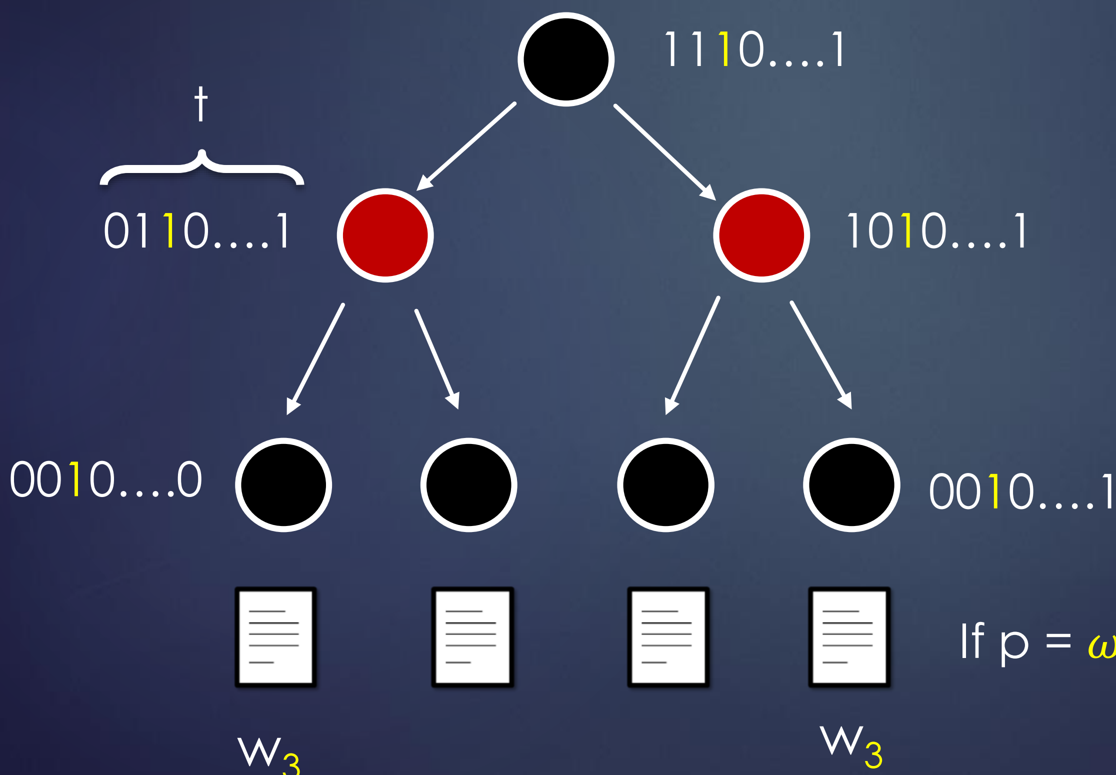


Worst-case

1. Search: O(log(n))
2. Add: O(log(n))
3. Delete: O(log(n))

A New Data Structure

- Keyword Red-Black (KRB) Trees



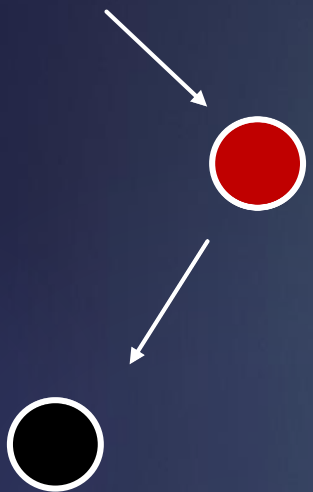
$\mathcal{W} = \{w_1, \ldots, w_t\}$

Search: $O(\frac{OPT}{p} \cdot \log(n))$

Add/delete: $O(\frac{\#f}{p} \cdot \log(n))$

If $p = \omega(\log(n))$ then search is $o(OPT)$

Encrypting KRB Trees



1010….1 $\xrightarrow{P_{K2}(w_3)}$ 010…1….1

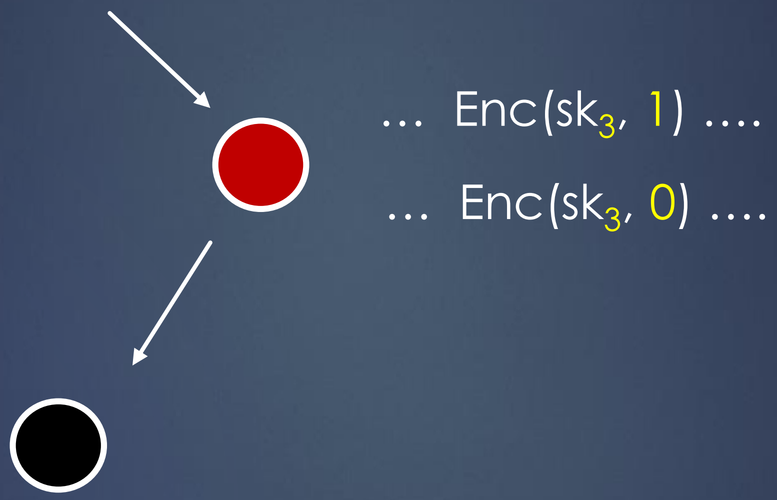
$sk_3 = Gen(1^k; F(w_3))$

… $Enc(sk_3, 1)$ ….

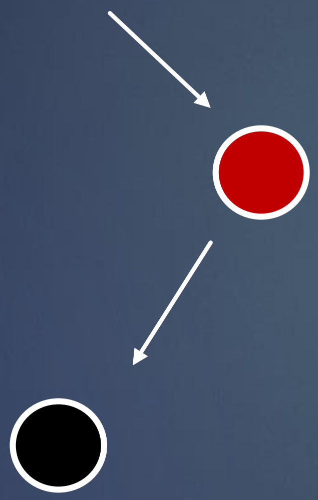
… $Enc(sk_3, 0)$ ….

$\Big\}\ H(N_{id} \mid P_{K2}(w_3))$

Encrypting KRB Trees



Searching KRB Trees



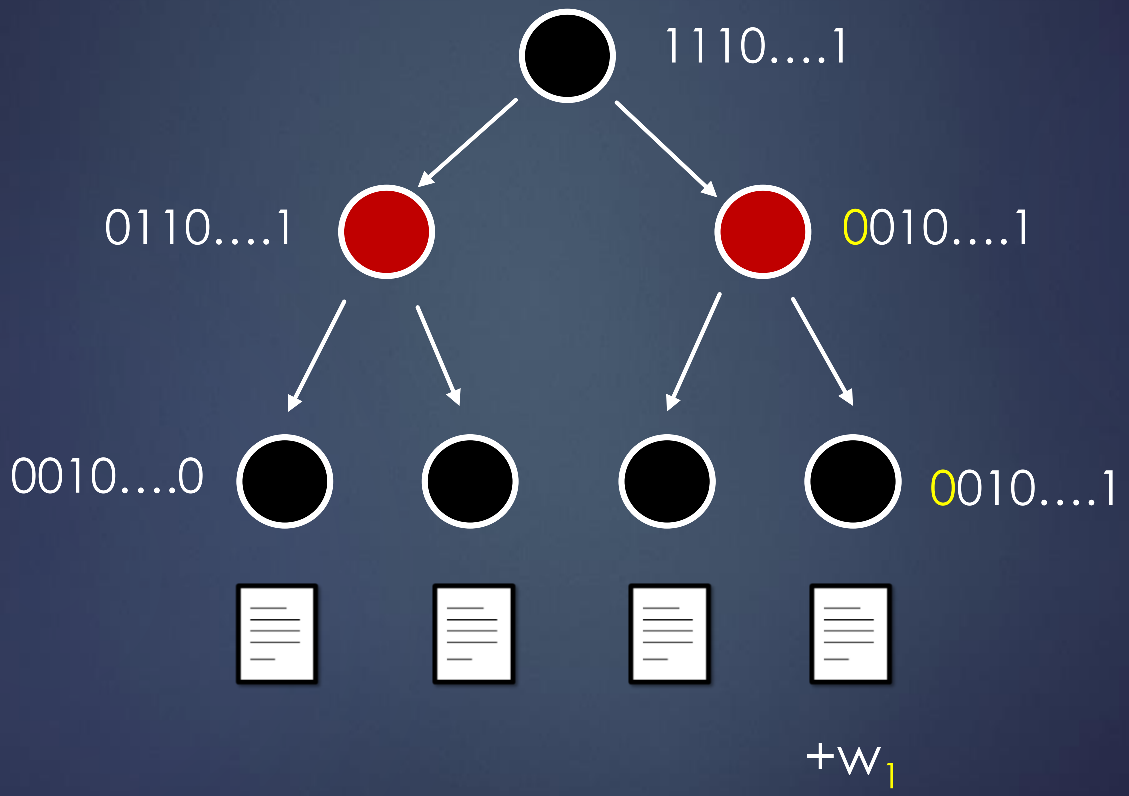
… $Enc(sk_3, 1)$ ….

… $Enc(sk_3, 0)$ ….

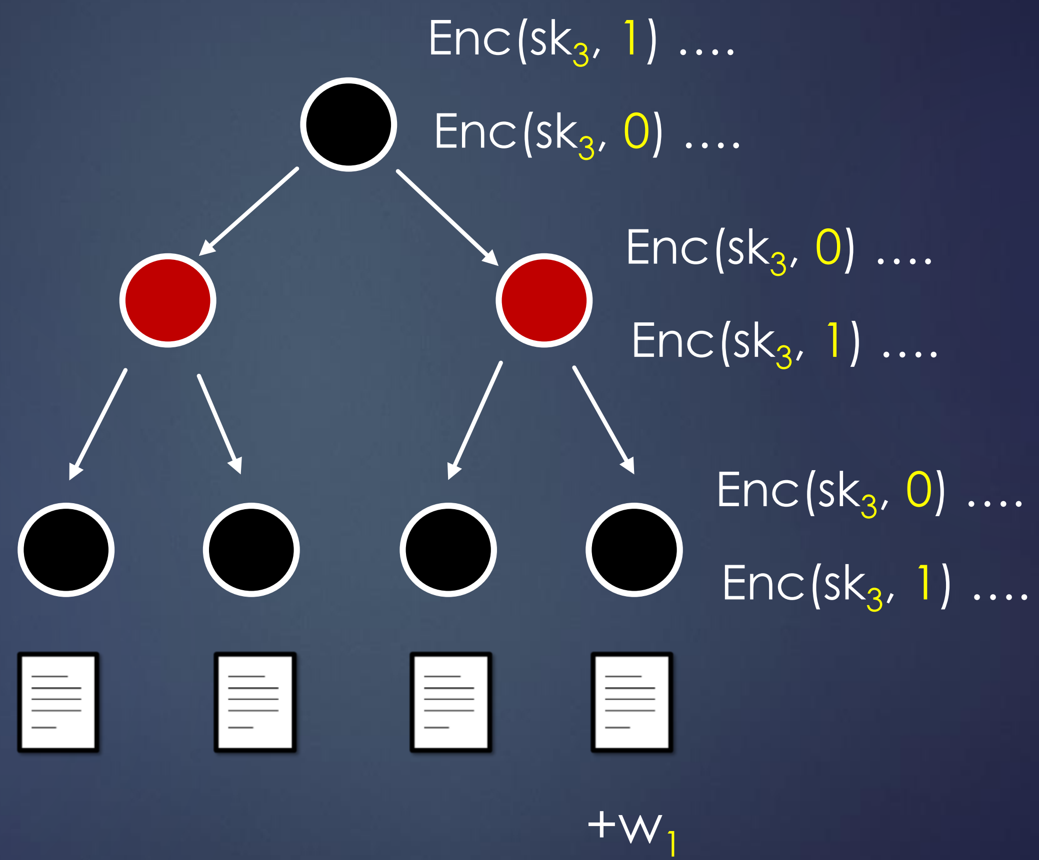
$\} H(N_{id} \mid P_{K2}(w_3))$

$Token_K(w_3) = P_{K2}(w_3), sk_3$

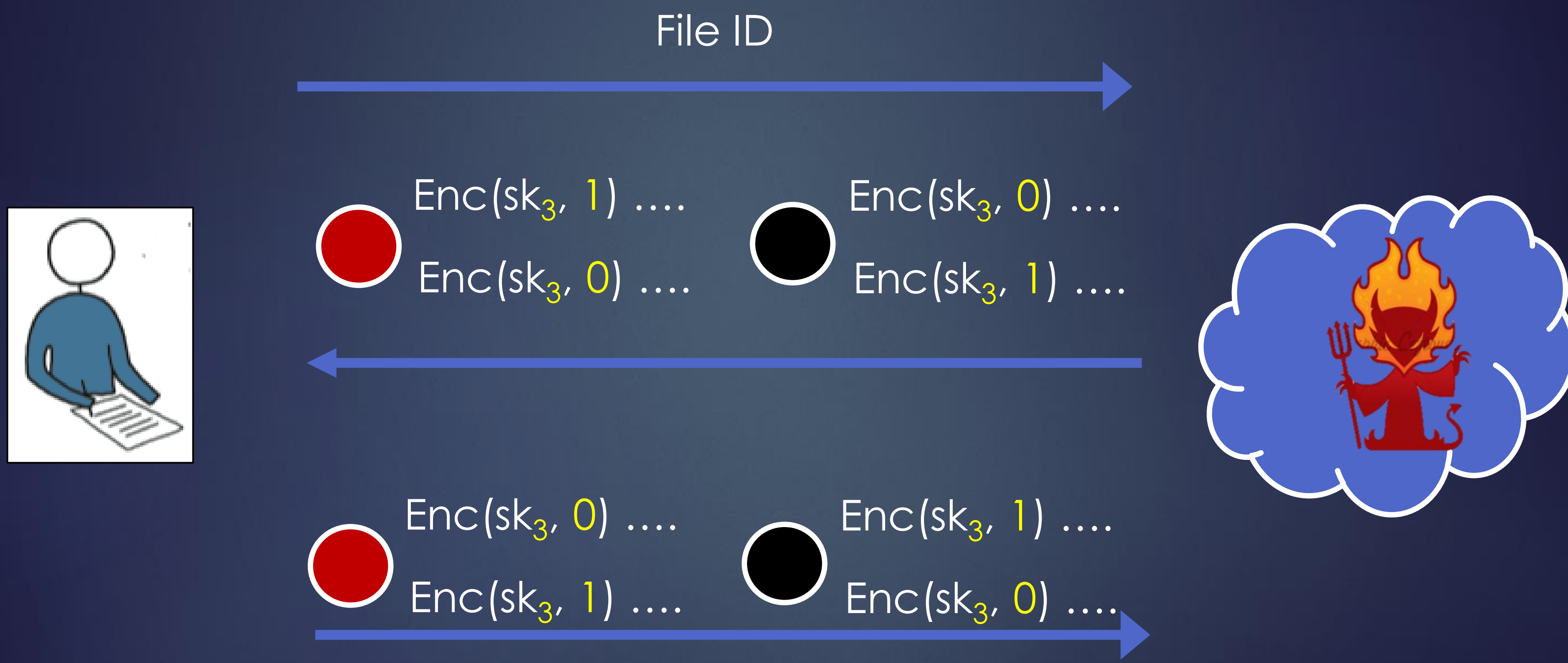
Updating KRB Trees



Updating Encrypted KRB Trees



Updating Encrypted KRB Trees



Thanks!