Privacy-Enhancing Technologies Symposium (PETS) 2019

Hardware-Supported ORAM In Effect: Practical Oblivious Search and Update on Very Large Dataset

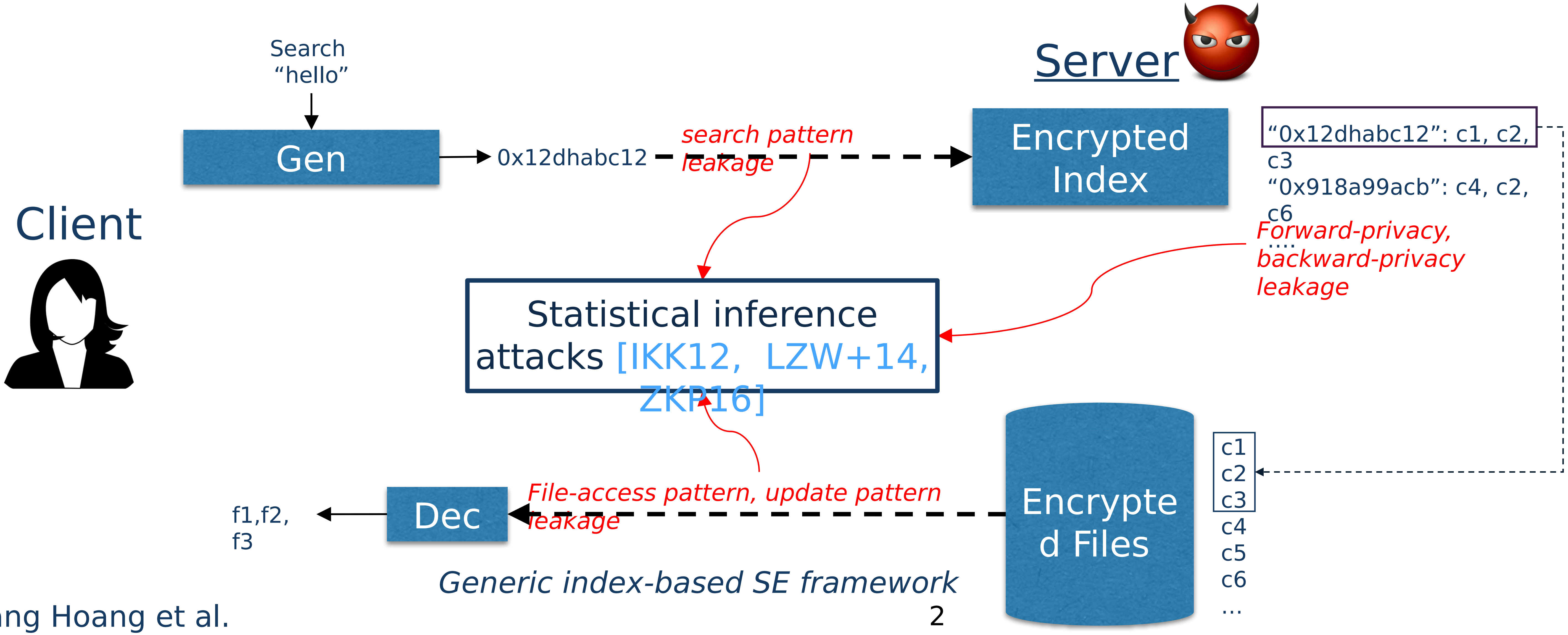
Thang Hoang[†,*], Muslum O. Ozmen[†,*], Yeongjin Jang[‡], Attila A. Yavuz[†,*]

[†] University of South Florida, Tampa, FL
[‡] Oregon State University, Corvallis, OR

** Part of work done while the first, second and last authors were at Oregon State University*

Introduction

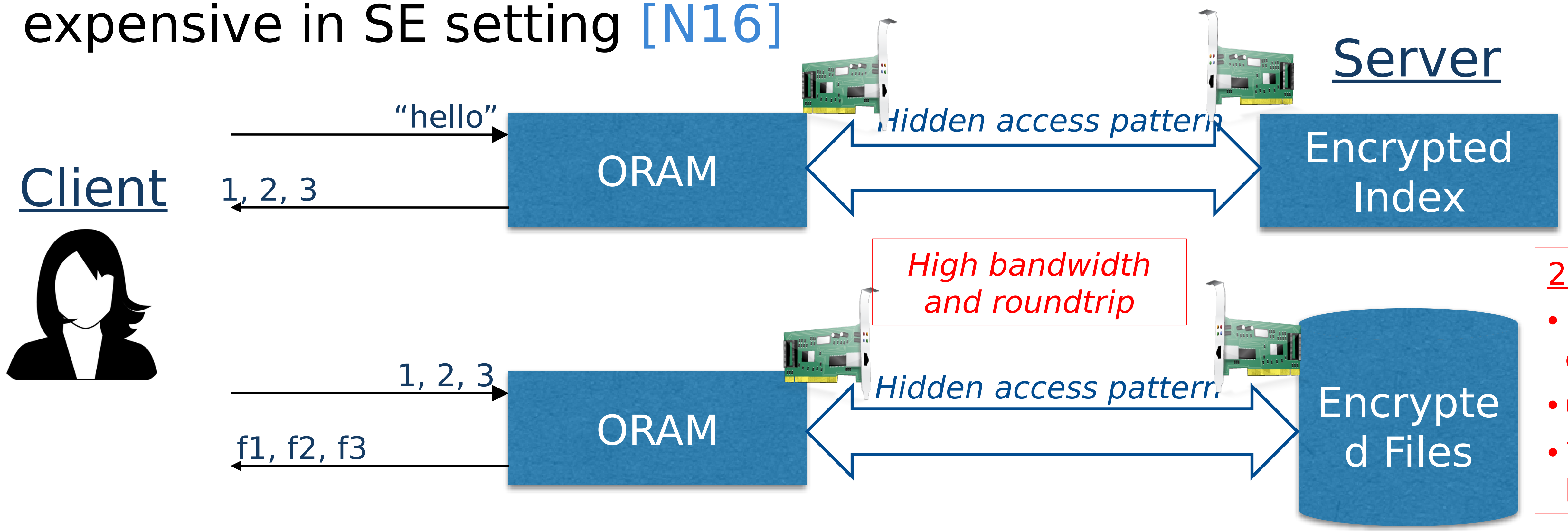
- Searchable encryption (SE) allows search/update operations on encrypted data
- State-of-the-art SE still leak significant information with many attacks shown



Generic index-based SE framework

Introduction

- Oblivious Random Access Machine (ORAM) can seal access pattern leakage but expensive in SE setting [N16]



27 GB WikiDB:

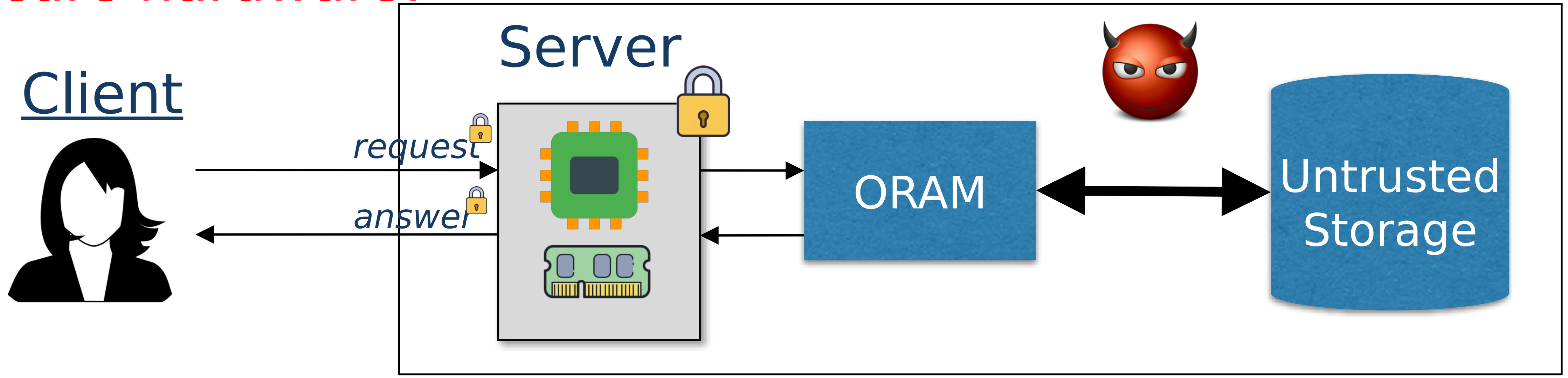
- 100 - 1000x roundtrips (20 ms latency each)
- 0.4 – 7 GB data transmitted per query
- 7s – 1.7 hours delay per query with high-speed network (150 Mbps)

- Passive ORAMs (storage-only server) is the most common and efficient ones
  - O(log N) communication overhead (proven as tight lower bound [GO96, LN18])
  - Significant delay and latency [SCE14]

Motivation

- ORAM seems the best option to hide access pattern but **very costly**
  - ORAM over the network results in significant delay due to the client's limited bandwidth

- Is there any way to execute ORAM but not over the network?
  - ➢ Use secure hardware!



ORAM with secure hardware [GO96, SGF17, RFK+17, MLS+13]

➢ Trusted Execution Environment (TEE) becomes widely available (e.g., Intel-SGX)

Our Contributions

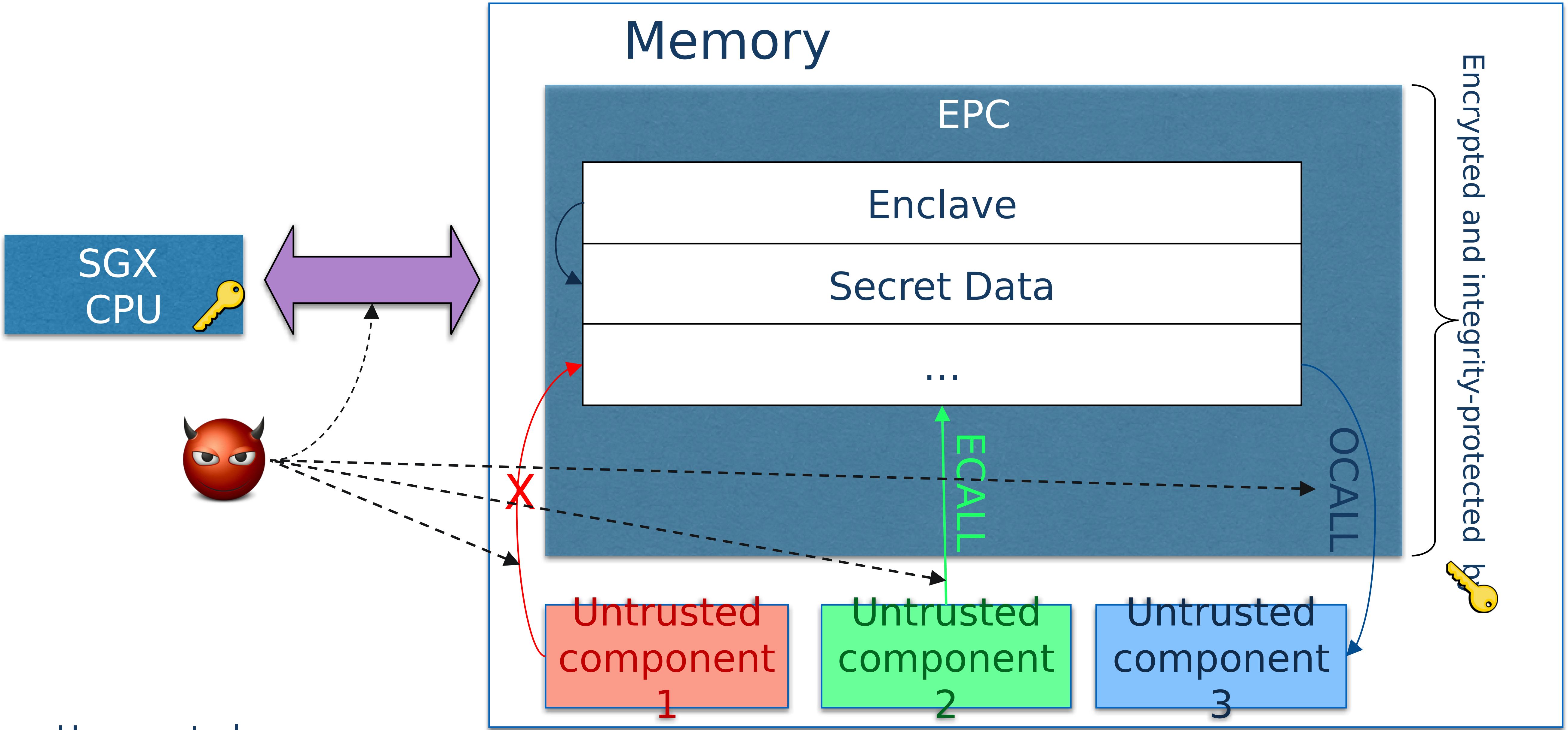
- POSUP: A new oblivious search and update platform design with Intel-SGX
  - Harness and optimize the most suitable cryptographic primitives for secure hardware
    - (Recursive) Circuit-ORAM, Oblivious Data Structures
  - Respect secure hardware constraints
    - Limited memory (95 MB for Intel-SGX)
    - Prevent side-channel access pattern leakages
- Implementation and evaluation with large DB
  - Code to be available soon (https://github.com/thanghoang/POSUP/)
  - Wikipedia Dataset: 27 GB, 7,075,917 keywords, 863,782,383 keyword-file pairs

Approach	Query latency*
Conventional ORAM+SE [N16]	7 s - 1.7 hours
Process Entire DB in SGX [FBB+17]	131 s - 157 s

* for 99.5% fraction of keywords

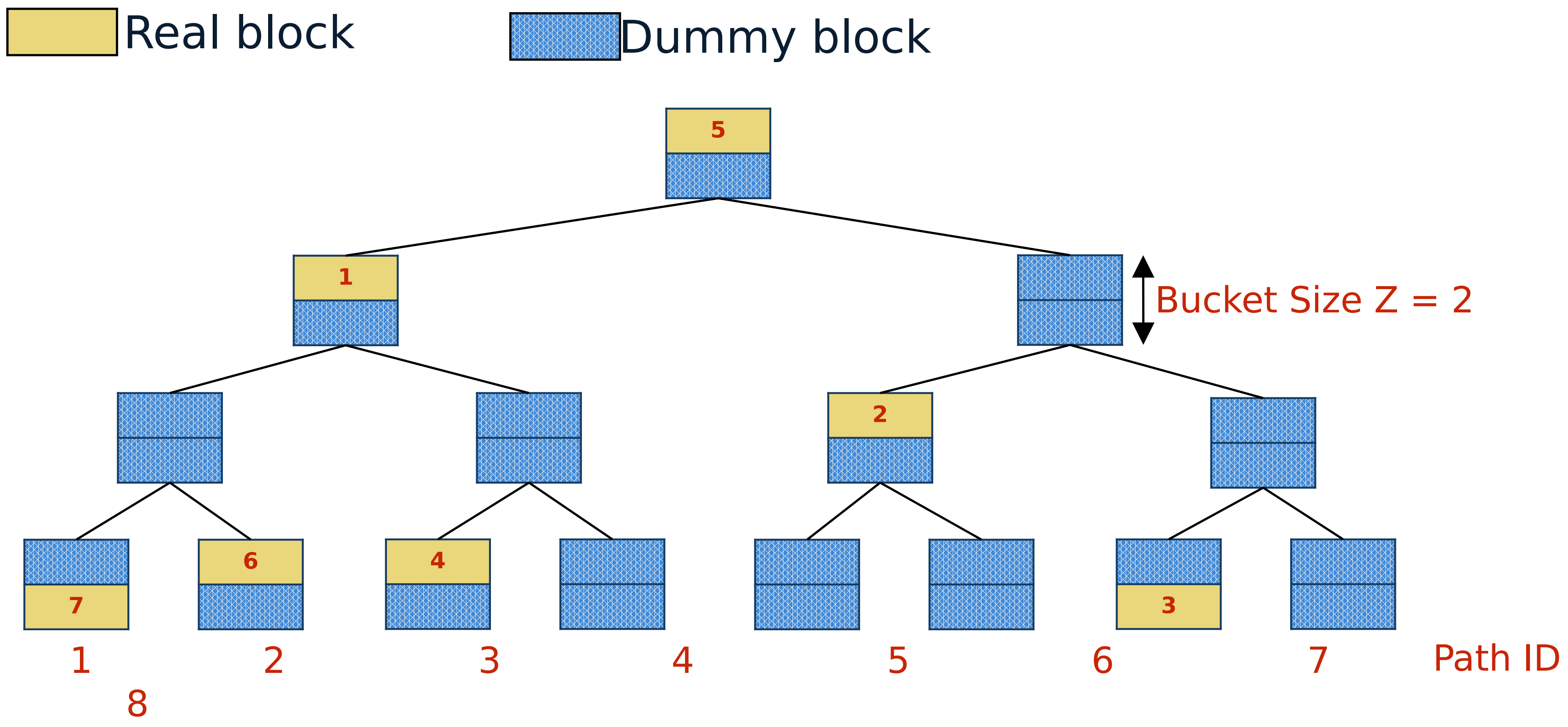
Secure Enclaves [JSR+16]

- Intel-SGX provides an *enclave* with hardware-based isolated, encrypted and integrity-protected memory
- Prevent any execution outside the enclave from accessing enclave's data

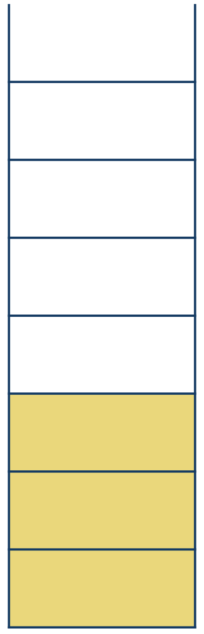


Circuit-ORAM [WCS15]

- Follows tree paradigm [SCS+11] with two main phases
  1. Read: Entire path but only keep 1 block into the stash
  2. Eviction: Push blocks to deeper levels as much as possible in a single scan
- Evict path: Deterministic, reverse lexicographic order



Stash

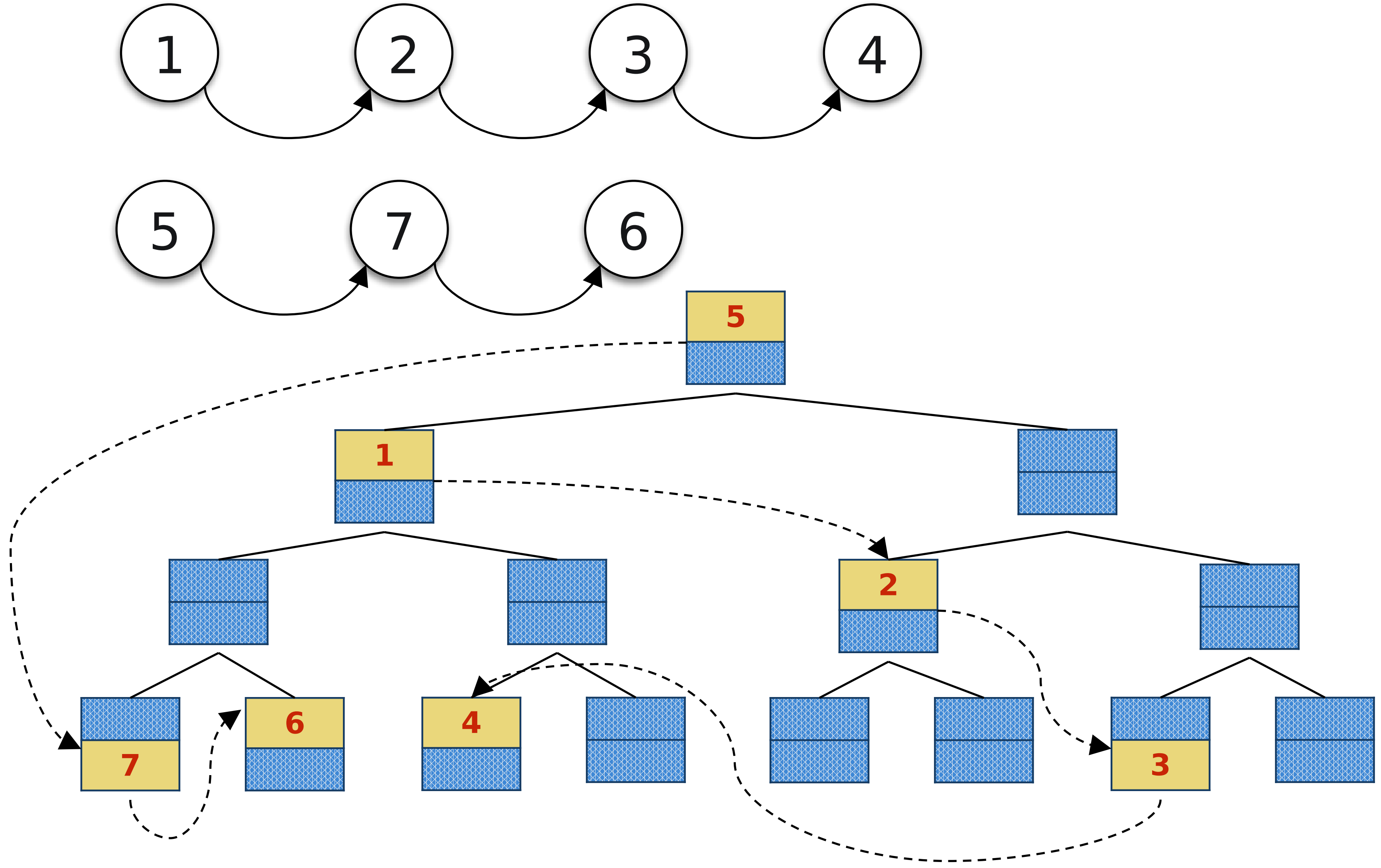


Position map

Block ID	Path ID
1	2
2	5
3	7
4	3
5	4
6	2
7	1

Oblivious Data Structures [WNL+14]

- Reduce the size of position map stored at the client
  - Each node store the position map of its logical next node and so forth
  - Only need to store the position map of the root(s)



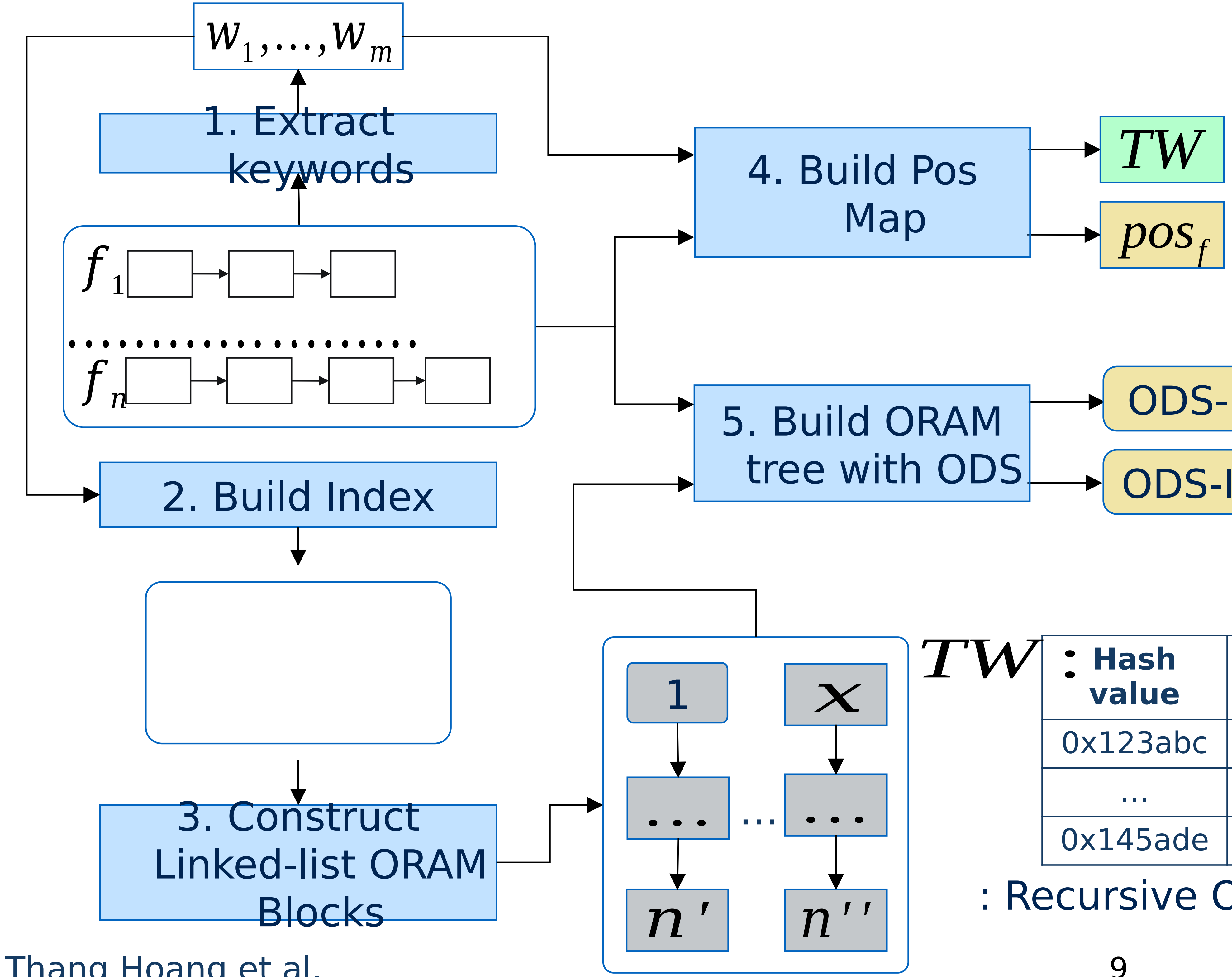
ORAM block structure:

bid	Payload	Next bid	Next pid
1	…	2	5
2	…	3	7
3	…	4	3
4	…	-	-
5	…	7	1
6	…	-	-
7	…	6	2

Position map

Block ID	Path ID
1	2
5	4

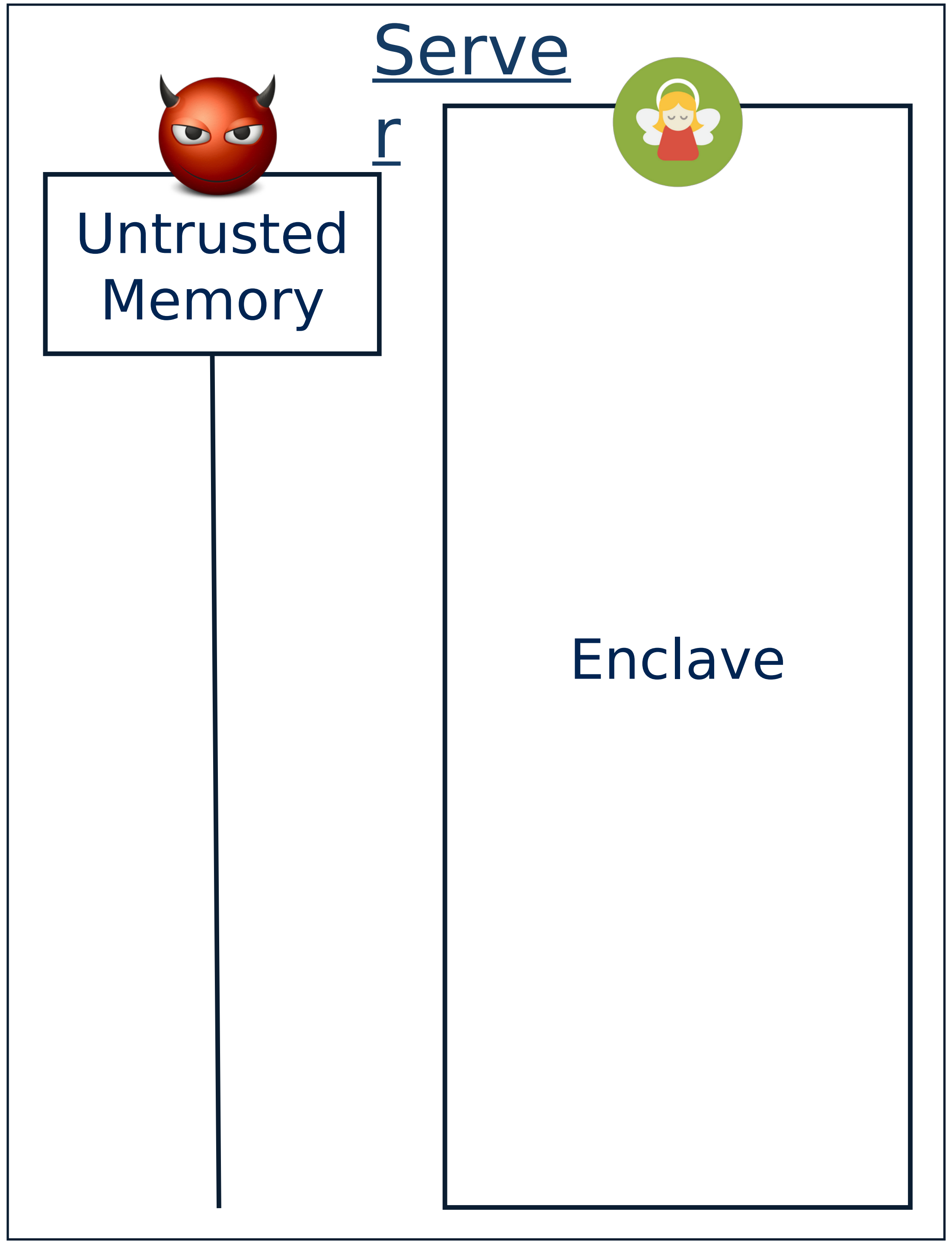
POSUP Setup



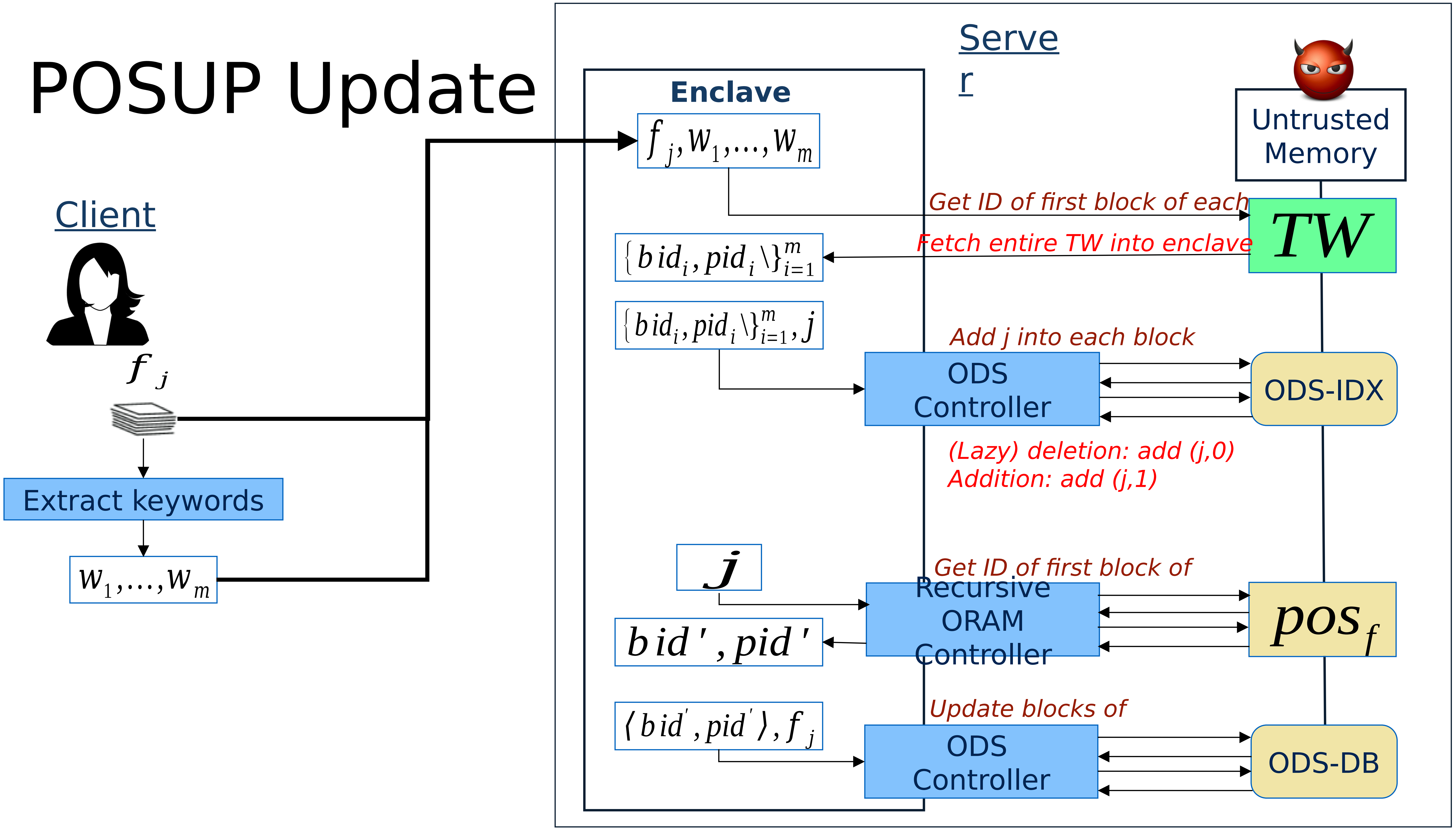
TW:

Hash value	bid	pid
0x123abc	1	2
…	…	…
0x145ade	5	4

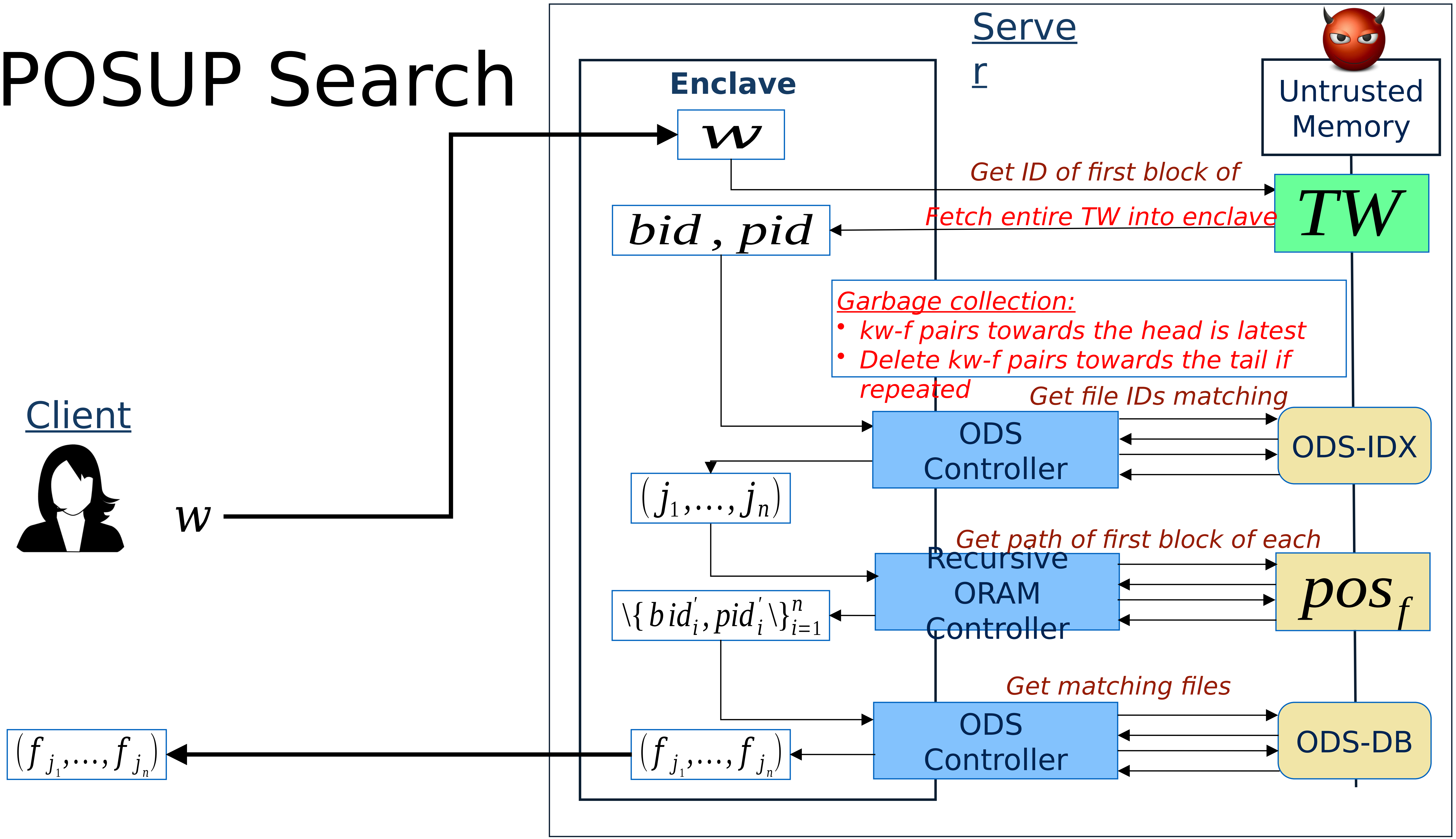
pos_f: Recursive ORAM structure



POSUP Update



POSUP Search



Hiding side-channel access pattern

- TW is a hash table
  - Linear scan (and loaded into SGX by 95MB chunks) to prevent which slots are accessed
- Stash is stored in untrusted memory region
  - Linear scan per block pushed/fetched to prevent which slot is accessed

- Conditional execution (if/else st.): Distinguishable access pattern due to execution branches
  - Use CMOV and SETE/SETG/SETGE for oblivious comparison and update [OSF+16]

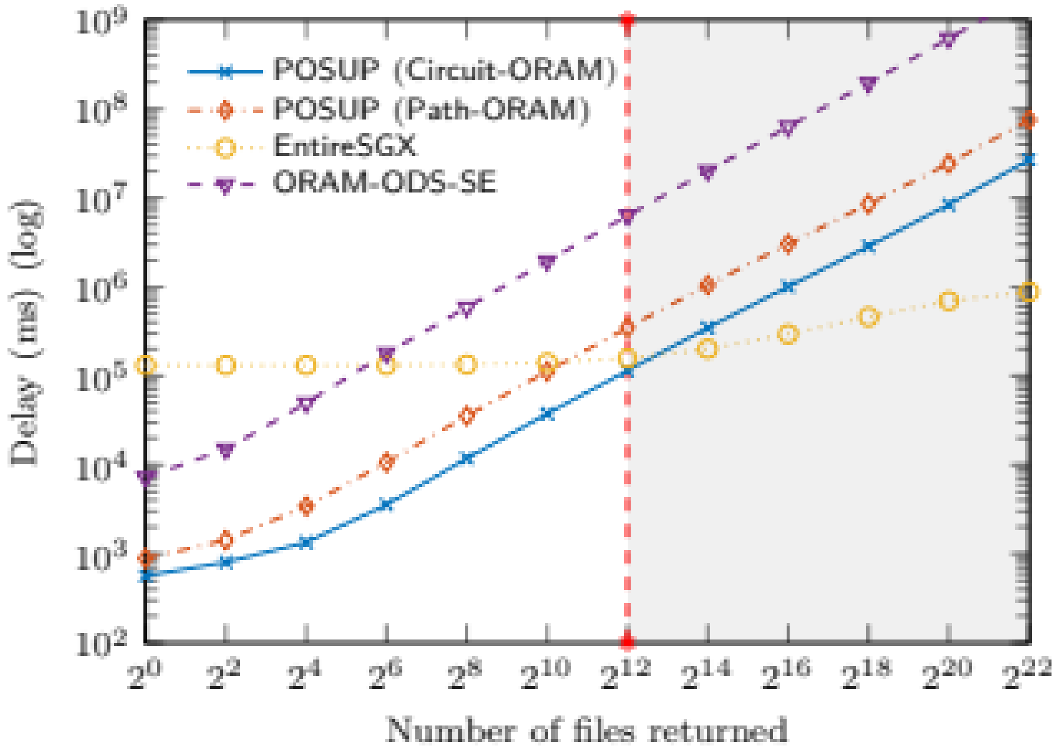
```
ocmp(x, y):
MOV rcx, x
MOV rdx, y
CMP rcx, rdx
SETE al
RETN
```

```
oupdt(b, x, y):
MOV rcx, b
MOV rdx, x
MOV rax, y
TEST rcx, rcx
CMOVZ rax, rdx
RETN
```

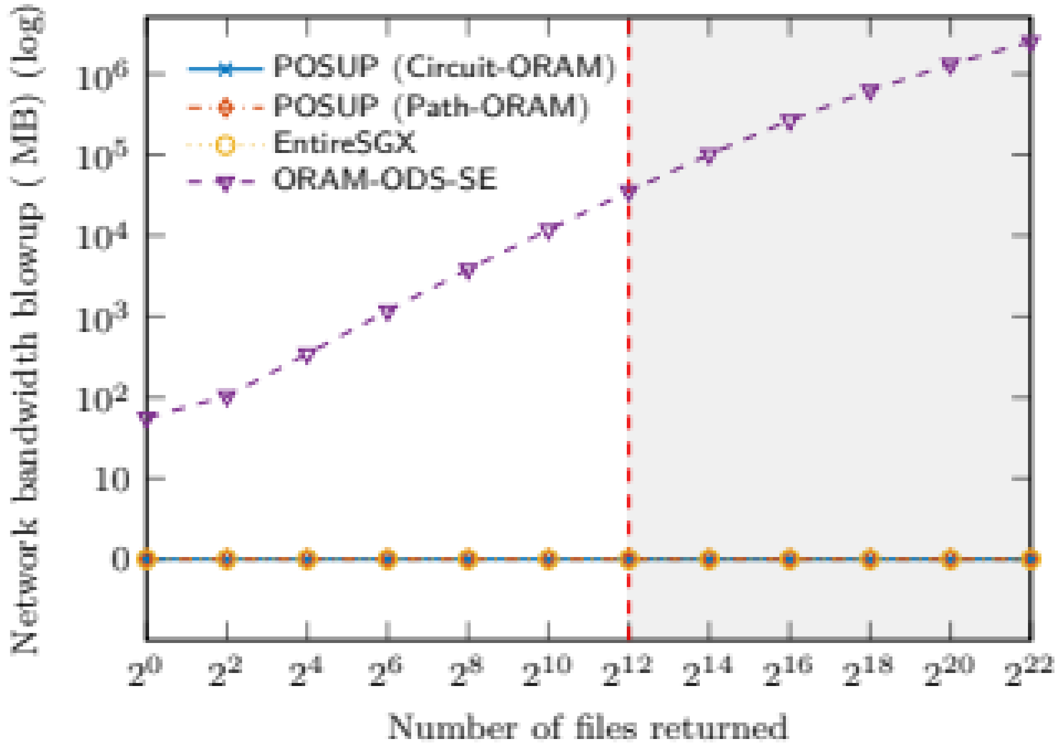
Experiment

- **Hardware:** Intel E3-1230 CPU (SGX-supported), 16 GB RAM, 512 GB SSD.
- **Dataset:** 27 GB Wikipedia English corpus with 5,554,594 files; 7,075,917 keywords; 863,782,383 keyword-file pairs; Index size: 6.9 GB
- **Network:** 18 ms latency, 150 Mbps throughput
- **POSUP Parameters:**
  - Path-ORAM and Circuit-ORAM with stash size |S| = 80
  - Block size: 3 KB for file blocks, 512 B for index blocks
- **POSUP counterparts for comparison:**
  - Path-ORAM+SE+ODS in client-server network setting
  - Process entire IDX and DB inside SGX (95-MB chunks loaded sequentially)

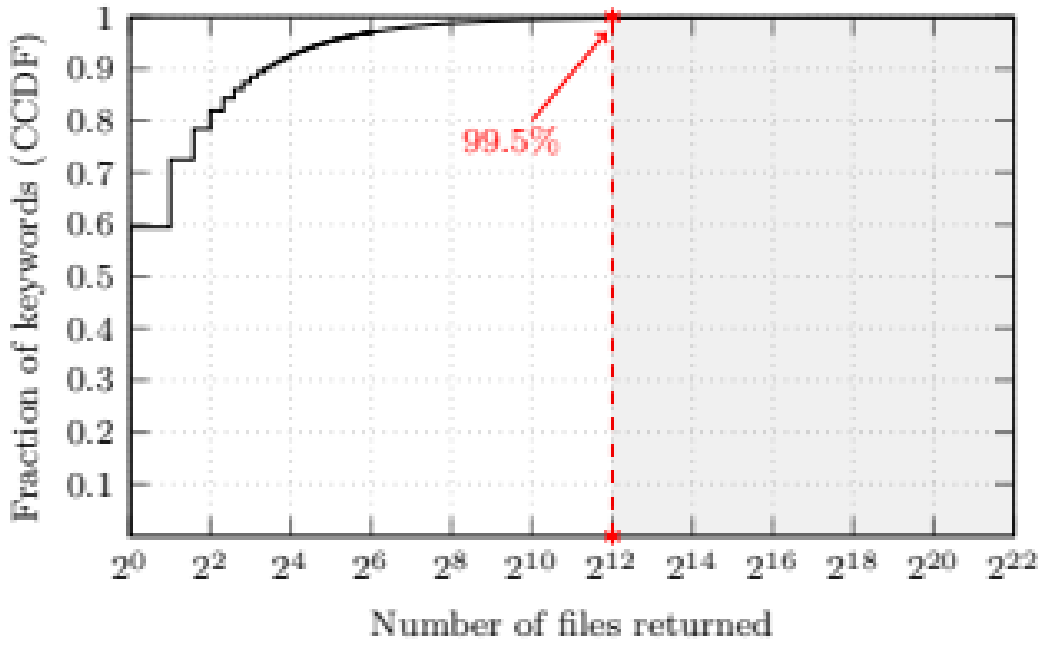
Experiment: Search



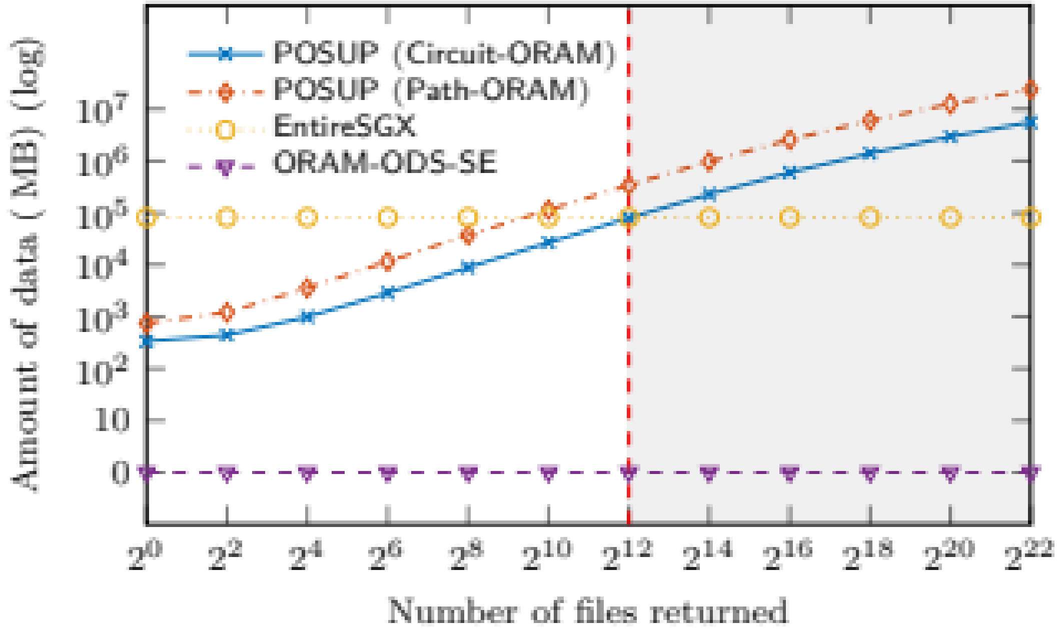
(a) End-to-end delay in POSUP and its counterparts regarding how many files returned in any single-keyword/boolean search.



(b) Network bandwidth increase of POSUP and its counterparts. Hardware-assisted techniques do not incur network overhead.



(c) Keyword distribution in enwiki dataset. An (x, y) point denotes that y fraction of keywords appear in less than x files.



(d) Amount of data being accessed and processed by SGX of POSUP and its counterparts.

- POSUP is 74× - 232× faster than its counterparts for 99.5% fraction of keywords
  - Minimal BW Usage
  - 4.5× - 245× less computation delay than EntireSGX

Experiment: Update (single file)

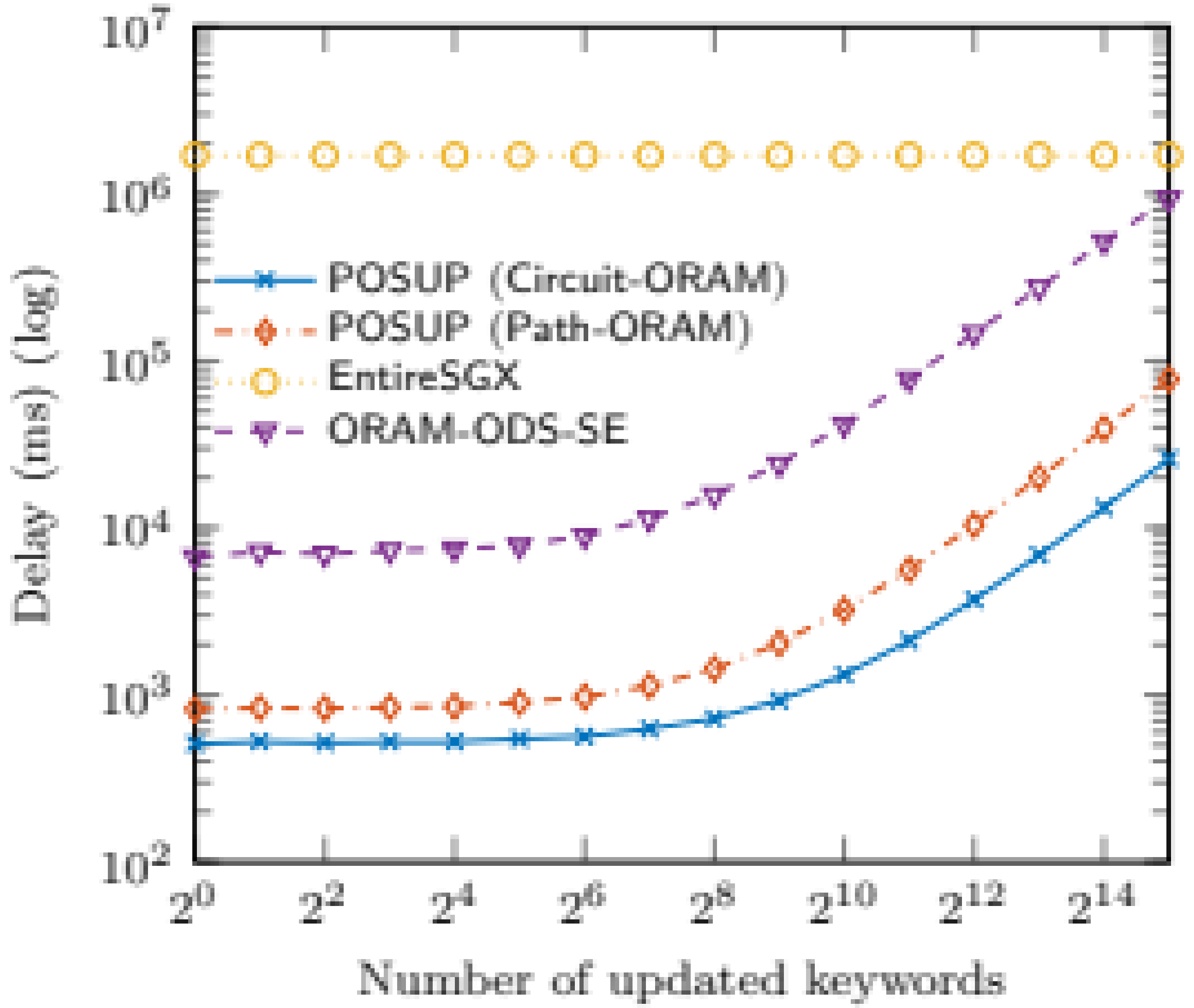


Fig. 9. End-to-end delay of updating a 290 KB file with different number of updated keywords involved in.

- For update, POSUP is 40× faster than ORAM-ODS-SE and up to approx. 1,000× faster than EntireSGX
- Remember Lazy add/delete: Access 1 block
  - Both ORAM-ODS-SE and POSUP
- EntireSGX decrypts and re-encrypts the entire DB and IDX per update
- Storage: |TW| + |ODS-IDX| + |ODS-DB|+| pos_f|
  - *Total: 175 GB (using Circuit-ORAM)*
  - *27 GB Wikiset*

Conclusion and Further Direction

- POSUSP: An SGX-supported oblivious search and update platform
  - Efficient composition of crypto primitives in the context of secure hardware
- With the support of secure hardware, oblivious search/update become much more practical

Limitation:

- Support only basic single-keyword search, multi-keyword can be done but with high cost
- Linear scan of Keyword hash table (210 msec, 188 MB)

Open Research Question:

  - More efficient and diverse oblivious queries (e.g., conjunctive/boolean/ranged)
  - Efficient oblivious hash table for keyword

Thank you for your attention!

?

Contacts: (hoangm@mail.usf.edu, attilaayavuz@usf.edu)

The code will be available soon at: https://github.com/thanghoang/POSUP/

References

[N16] Naveed, Muhammad. "The Fallacy of Composition of Oblivious RAM and Searchable Encryption." IACR Cryptology ePrint Archive 2015 (2015): 668.

[SCE14] Stefanov, Emil, Charalampos Papamanthou, and Elaine Shi. "Practical Dynamic Searchable Encryption with Small Leakage." In NDSS, vol. 71, pp. 72-75. 2014.

[GO96] Goldreich, Oded, and Rafail Ostrovsky. "Software protection and simulation on oblivious RAMs." Journal of the ACM (JACM) 43, no. 3 (1996): 431-473.

[LN18] Larsen, Kasper Green, and Jesper Buus Nielsen. "Yes, there is an oblivious RAM lower bound!." In Annual International Cryptology Conference, pp. 523-542. Springer, Cham, 2018.

[WCS15] Wang, Xiao, Hubert Chan, and Elaine Shi. "Circuit oram: On tightness of the goldreich-ostrovsky lower bound." In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, pp. 850-861. ACM, 2015.

[SCS+11] Shi, Elaine, T-H. Hubert Chan, Emil Stefanov, and Mingfei Li. "Oblivious RAM with O ((logN) 3) worst-case cost." In International Conference on The Theory and Application of Cryptology and Information Security, pp. 197-214. Springer, Berlin, Heidelberg, 2011.

[WNL+14] Wang, Xiao Shaun, Kartik Nayak, Chang Liu, T. H. Chan, Elaine Shi, Emil Stefanov, and Yan Huang. "Oblivious data structures." In Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, pp. 215-226. ACM, 2014.

[ZKP16] Zhang, Yupeng, Jonathan Katz, and Charalampos Papamanthou. "All your queries are belong to us: The power of file-injection attacks on searchable encryption." In 25th {USENIX} Security Symposium ({USENIX} Security 16), pp. 707-720. 2016.

[LZW+14] Liu, Chang, Liehuang Zhu, Mingzhong Wang, and Yu-An Tan. "Search pattern leakage in searchable encryption: Attacks and new construction." Information Sciences 265 (2014): 176-188.

[IKK12] Islam, Mohammad Saiful, Mehmet Kuzu, and Murat Kantarcioglu. "Access Pattern disclosure on Searchable Encryption: Ramification, Attack and Mitigation." In Ndss, vol. 20, p. 12. 2012.

References

[SGF17] Sasy, Sajin, Sergey Gorbunov, and Christopher W. Fletcher. "ZeroTrace: Oblivious Memory Primitives from Intel SGX." IACR Cryptology ePrint Archive 2017 (2017): 549.

[RFK+17] Ren, Ling, Christopher W. Fletcher, Albert Kwon, Marten Van Dijk, and Srinivas Devadas. "Design and implementation of the ascend secure processor." IEEE Transactions on Dependable and Secure Computing 16, no. 2 (2017): 204-216.

[MLS+13] Maas, Martin, Eric Love, Emil Stefanov, Mohit Tiwari, Elaine Shi, Krste Asanovic, John Kubiatowicz, and Dawn Song. "Phantom: Practical oblivious computation in a secure processor." In Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security, pp. 311-324. ACM, 2013.

[FBB+17] Fuhry, Benny, Raad Bahmani, Ferdinand Brasser, Florian Hahn, Florian Kerschbaum, and Ahmad-Reza Sadeghi. "HardIDX: Practical and secure index with SGX." In IFIP Annual Conference on Data and Applications Security and Privacy, pp. 386-408. Springer, Cham, 2017.

[JSR+16] Johnson, Simon, Vinnie Scarlata, Carlos Rozas, Ernie Brickell, and Frank Mckeen. "Intel® software guard extensions: Epid provisioning and attestation services." White Paper 1 (2016): 1-10.

[OSF+16] Ohrimenko, Olga, Felix Schuster, Cédric Fournet, Aastha Mehta, Sebastian Nowozin, Kapil Vaswani, and Manuel Costa. "Oblivious multi-party machine learning on trusted processors." In 25th {USENIX} Security Symposium ({USENIX} Security 16), pp. 619-636. 2016.

Experiment – Microbenchmark

Operation	Execution Time (μs)	
	Path-ORAM	Circuit-ORAM
ODS access on ODS-IDX		
I/O Access	134	144
Enclave Process	2,362	686
Total	2,496	830
ODS access on ODS-DB		
I/O Access	156	285
Enclave Process	3,909	746
Total	4,065	1,031
Recursive ORAM on file position map		
I/O Access	34	41
Enclave Process	13,246	4,631
Total	13,280	4,672

- With Circuit-ORAM, POSUP takes 1 ms to access a 3 KB block in 107 GB DB
- Path-ORAM is slower than Circuit-ORAM for SGX since entire stash is loaded multiple times
- File position map access:
  - I/O access is low because it is stored on RAM memory
  - Enclave process is high because it decrypts/re-encrypt multiple recursive levels