A Quick Intro to Searchable Encryption

Theory & Practice - Constructions & Attacks

R. Bost - 21/04/2021

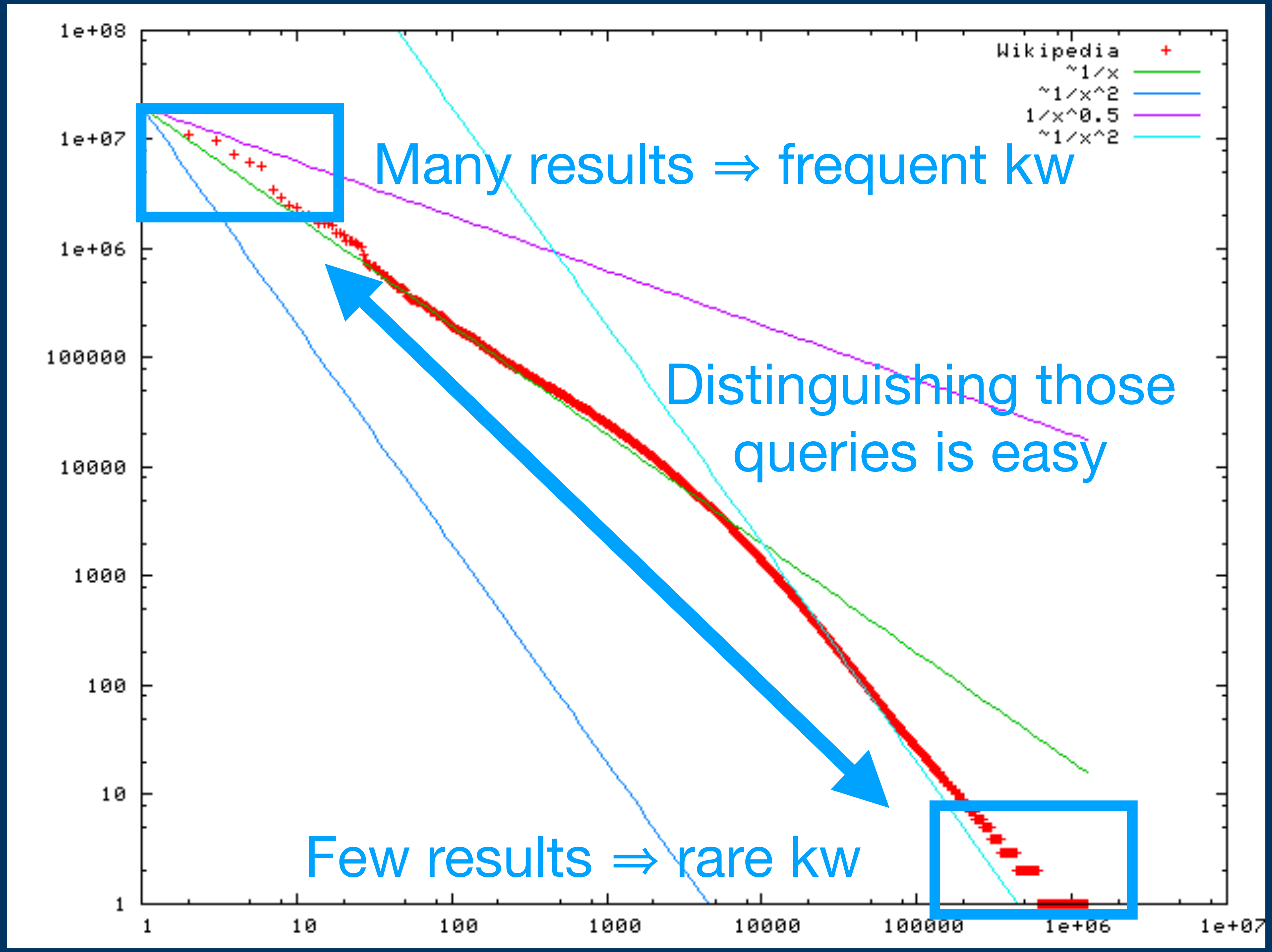
Searchable Encryption

Outsource data

- Securely
- Keep search functionalities
- Aimed at efficiency
- … we have to leak some information …
- … and this can lead to devastating attacks

Searchable Encryption

- We want to protect both data & queries from the server
  - Query only: PIR
  - Data only: does not really make sense
  - In practice, the docs are stored separately from the index, and the index is ‘encrypted’
- Example of leakage vs efficiency: keyword frequency
  - Padding or ~~O(N) comp./comm.~~



Property Preserving Encryption

Deterministic encryption, Order Preserving Encryption

✓ Legacy compatible (works on top of unencrypted DB)

✓ Very efficient

✗ Not secure in practice (frequency analysis)

FHE & ORAM

Fully Homomorphic Encryption

✓ Support arbitrary queries

✓ Fully secure

✗ Not efficient at all

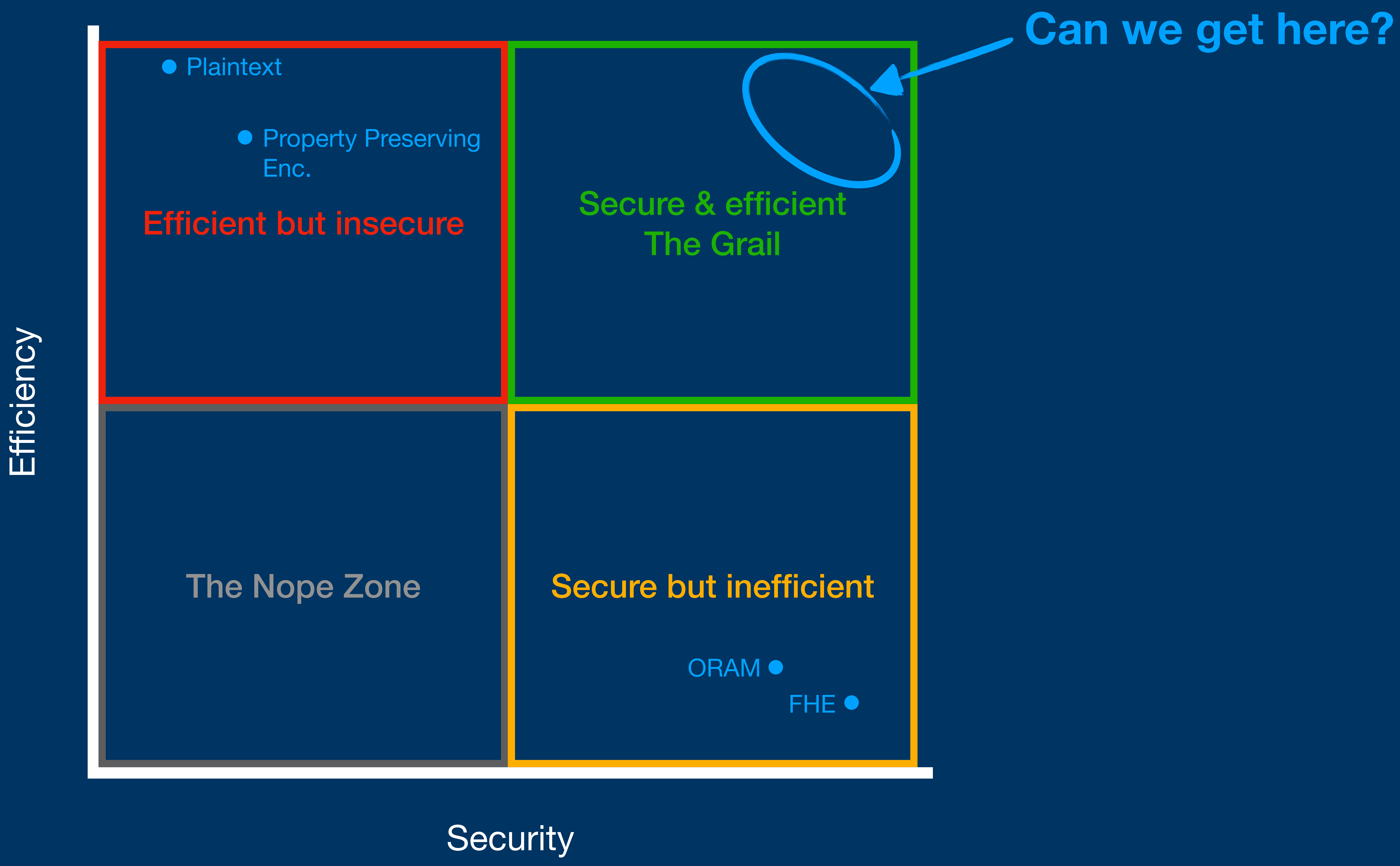
Oblivious RAM

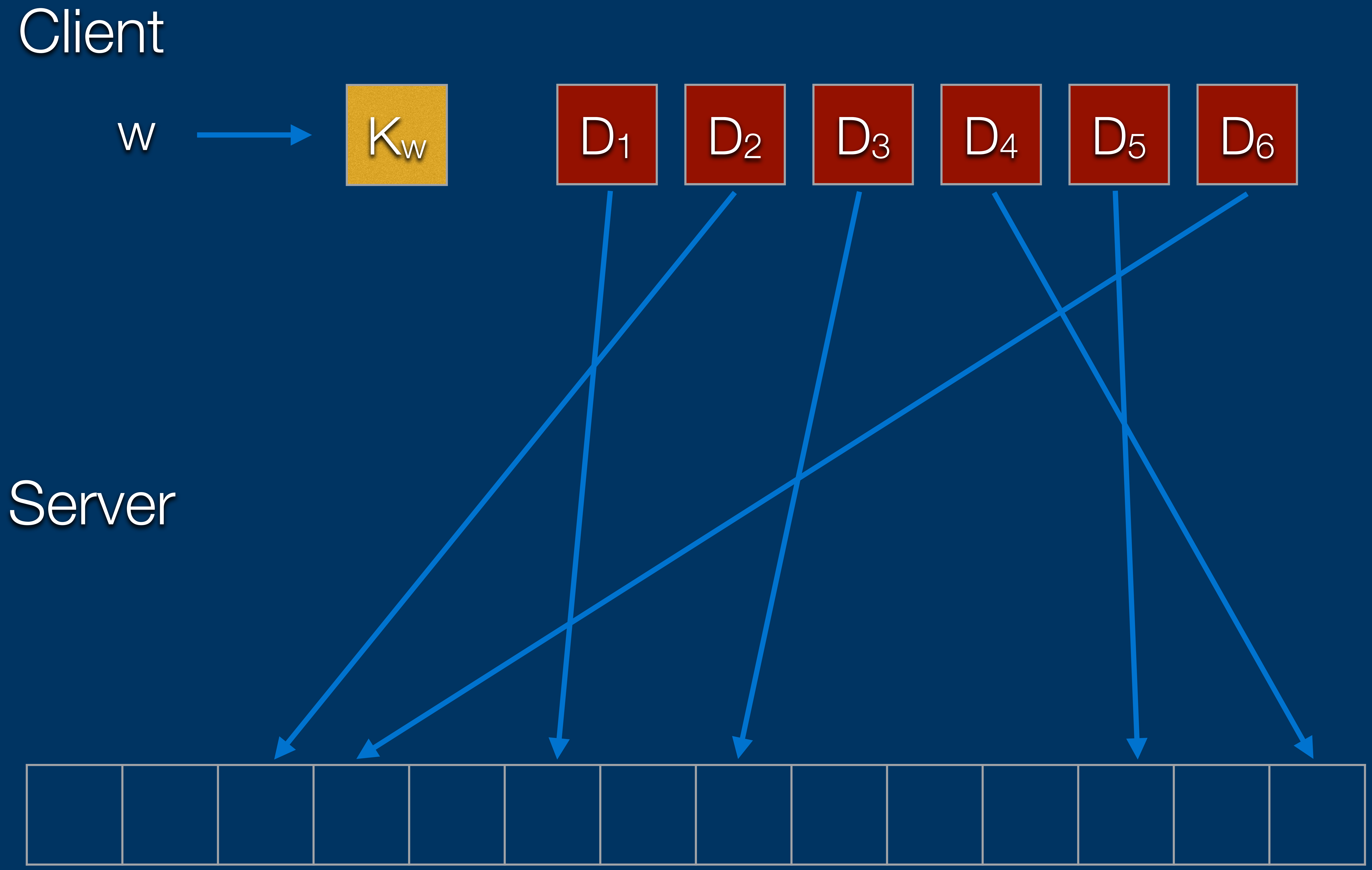
✓ Support arbitrary queries

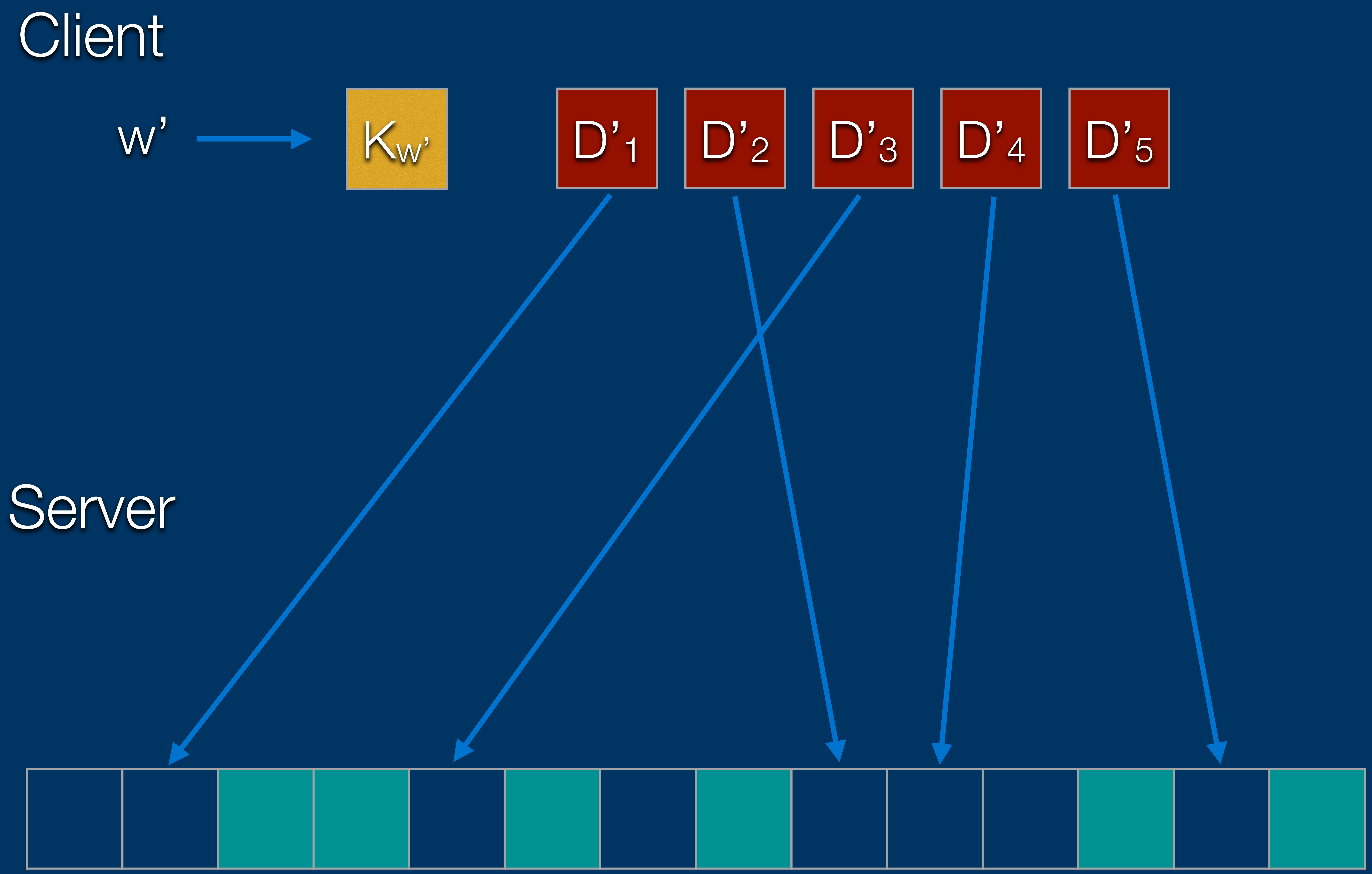
✓ Reveals the results count

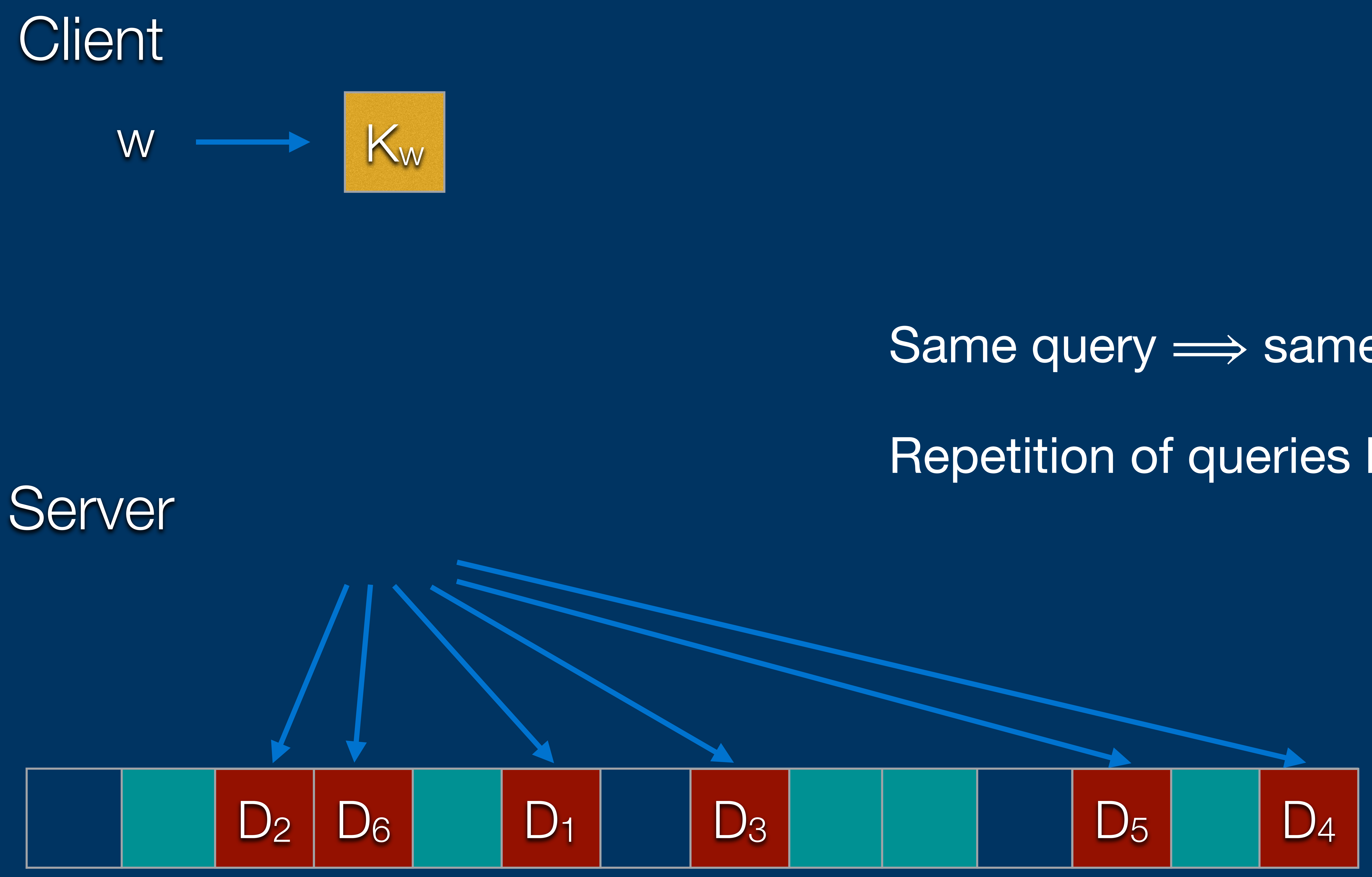
✗ Large communication overhead

Make compromise!


Can we get here?
Plaintext
Property Preserving Enc.
Efficient but insecure
Secure & efficient
The Grail
The Nope Zone
Secure but inefficient
ORAM
FHE
Efficiency
Security


Client
w
Kw
D1
D2
D3
D4
D5
D6
Server


Client
w'
$K_{w'}$
D'_1
D'_2
D'_3
D'_4
D'_5
Server



Same query $\Longrightarrow$ same accesses

Repetition of queries leaks

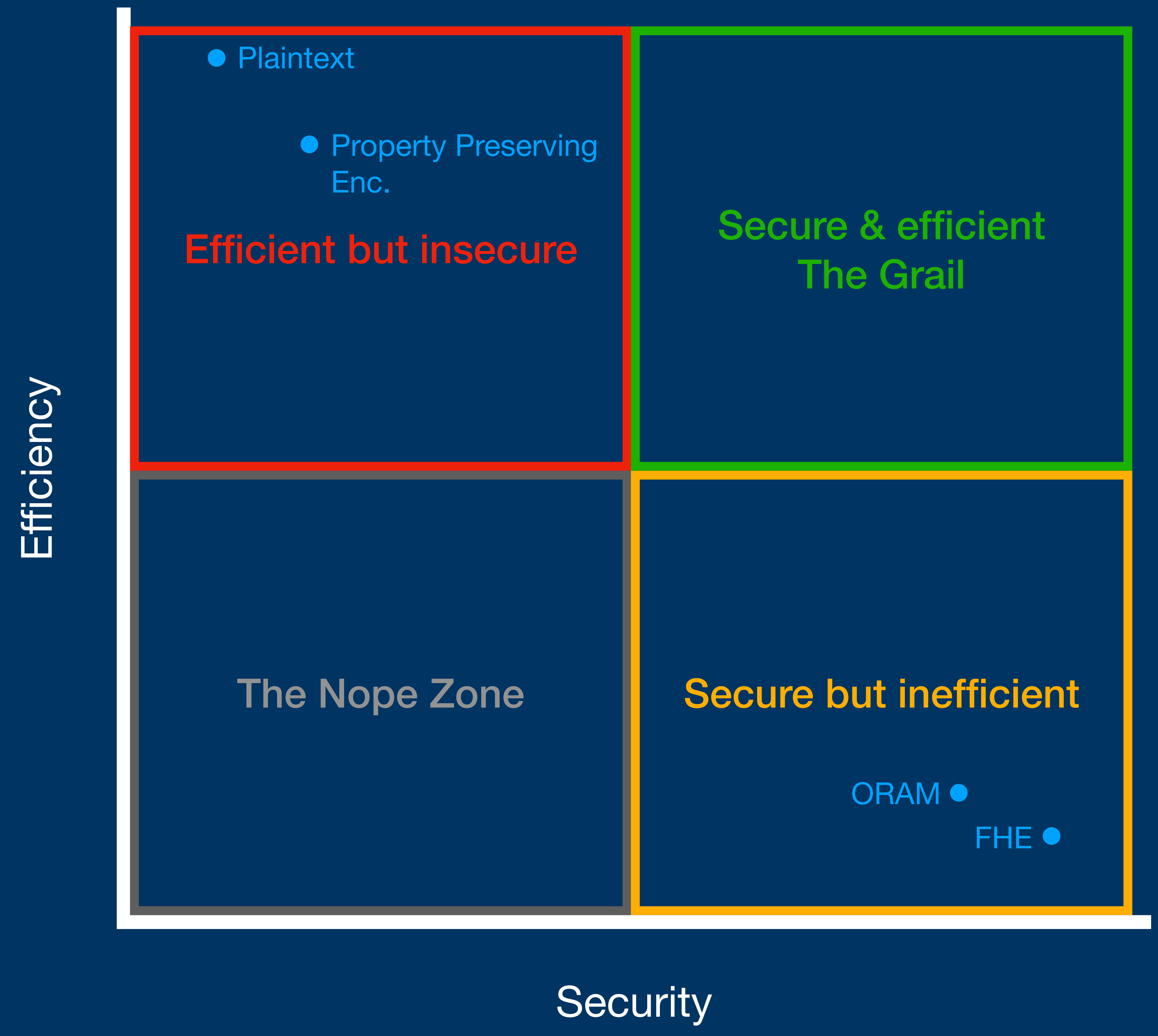
Lower bounds

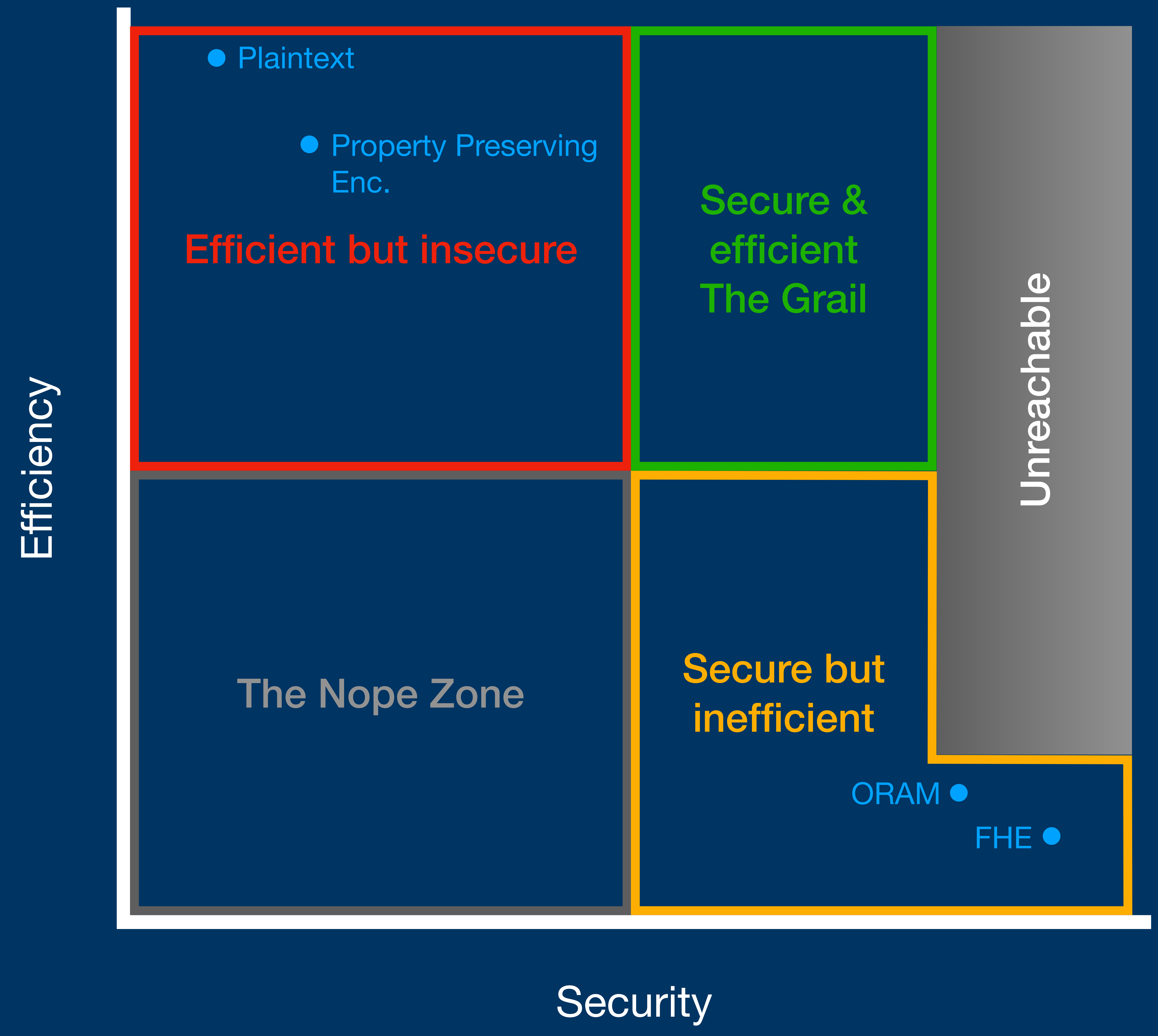
- Oblivious RAM lower bound: if one wants to hide the access pattern to a memory of size N, the computational overhead is

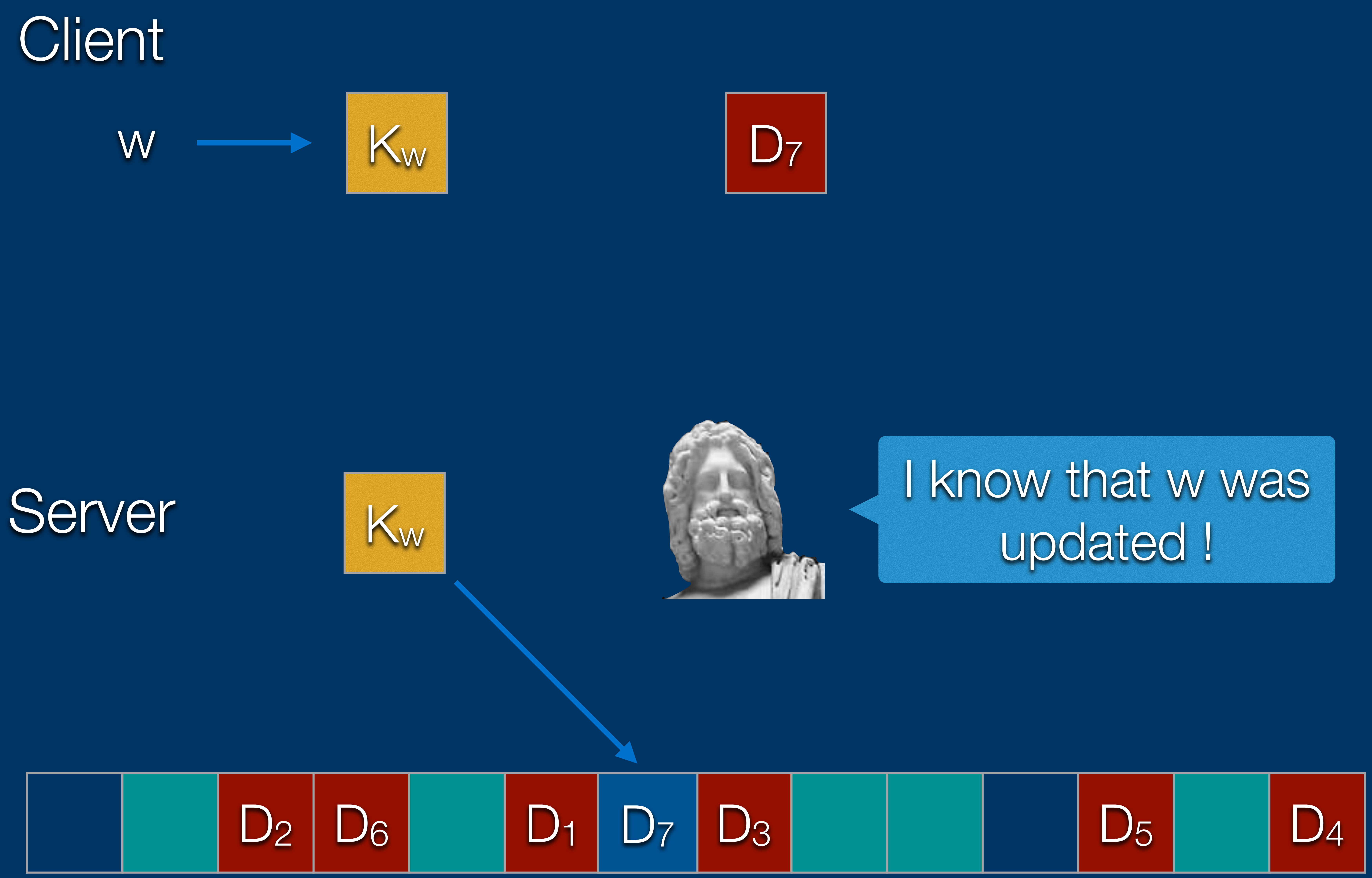
$$\Omega\left(\frac{\log N}{\log \sigma}\right)$$

- A similar lower bound exists for searchable encryption: a search pattern-hiding SE incurs a search overhead of

$$\Omega\left(\frac{\log \binom{|DB|}{n_w}}{\log \sigma}\right)$$

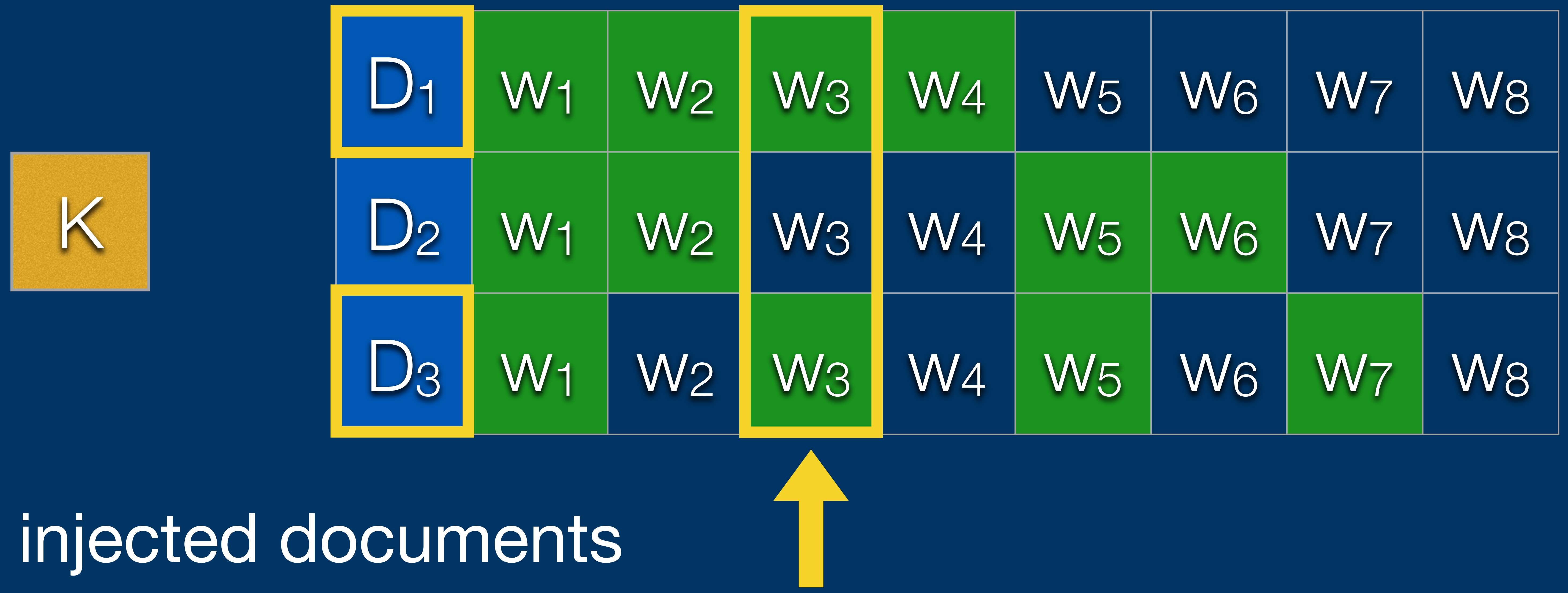

Plaintext
Property Preserving Enc.
Efficient but insecure
Secure & efficient
The Grail
The Nope Zone
Secure but inefficient
ORAM
FHE
Efficiency
Security


Plaintext
Property Preserving Enc.
Efficient but insecure
Secure & efficient
The Grail
Unreachable
The Nope Zone
Secure but inefficient
ORAM
FHE
Efficiency
Security


Client
w
K_w
D_7
Server
K_w
I know that w was updated !
D_2
D_6
D_1
D_7
D_3
D_5
D_4

File injection attacks [ZKP'16]

- Insert purposely crafted documents in the DB (*e.g.* spam for encrypted emails)



log |W| injected documents

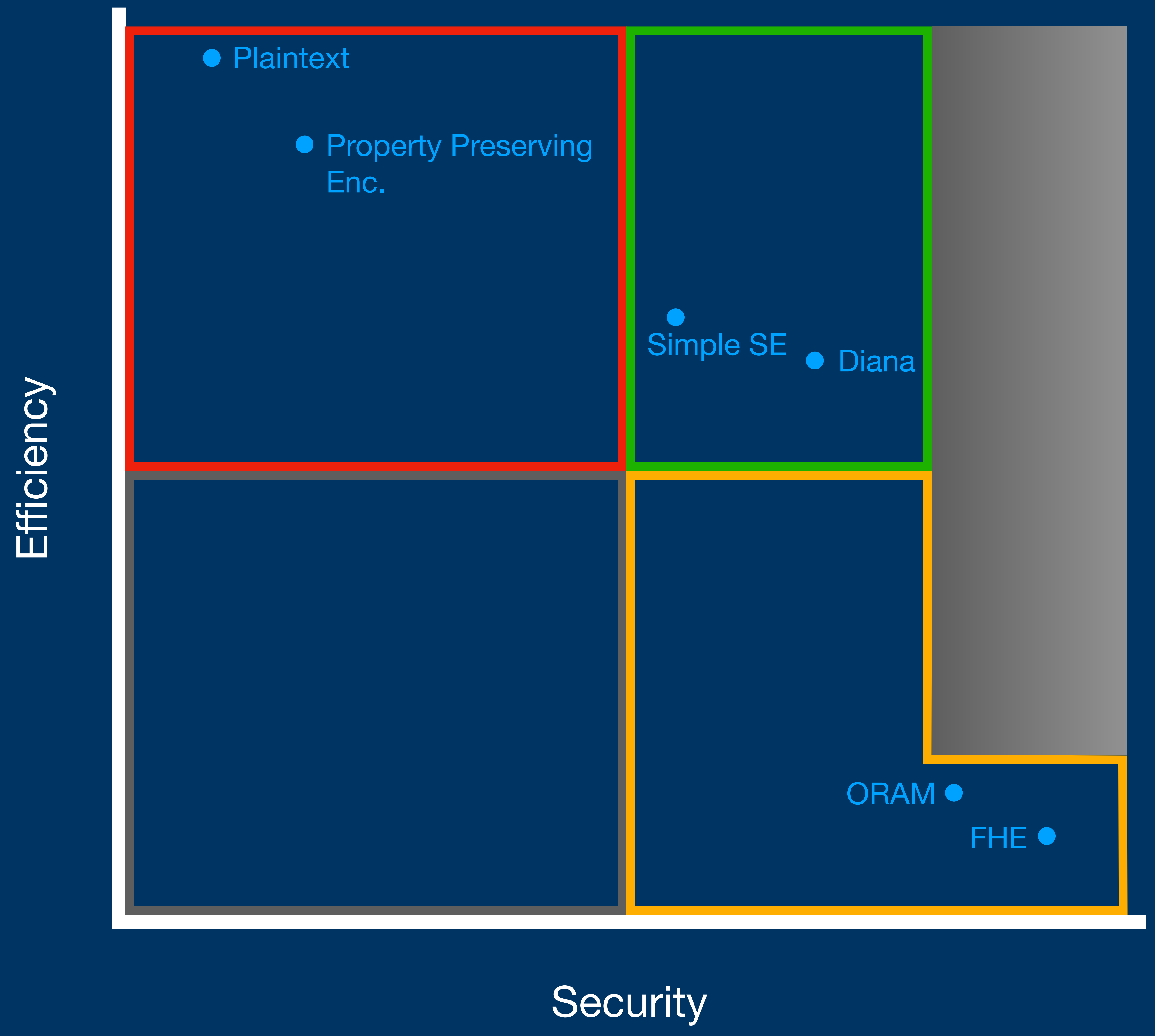
Active adaptive attacks

- These adaptive attacks use the update leakage
- We need SE schemes with oblivious updates

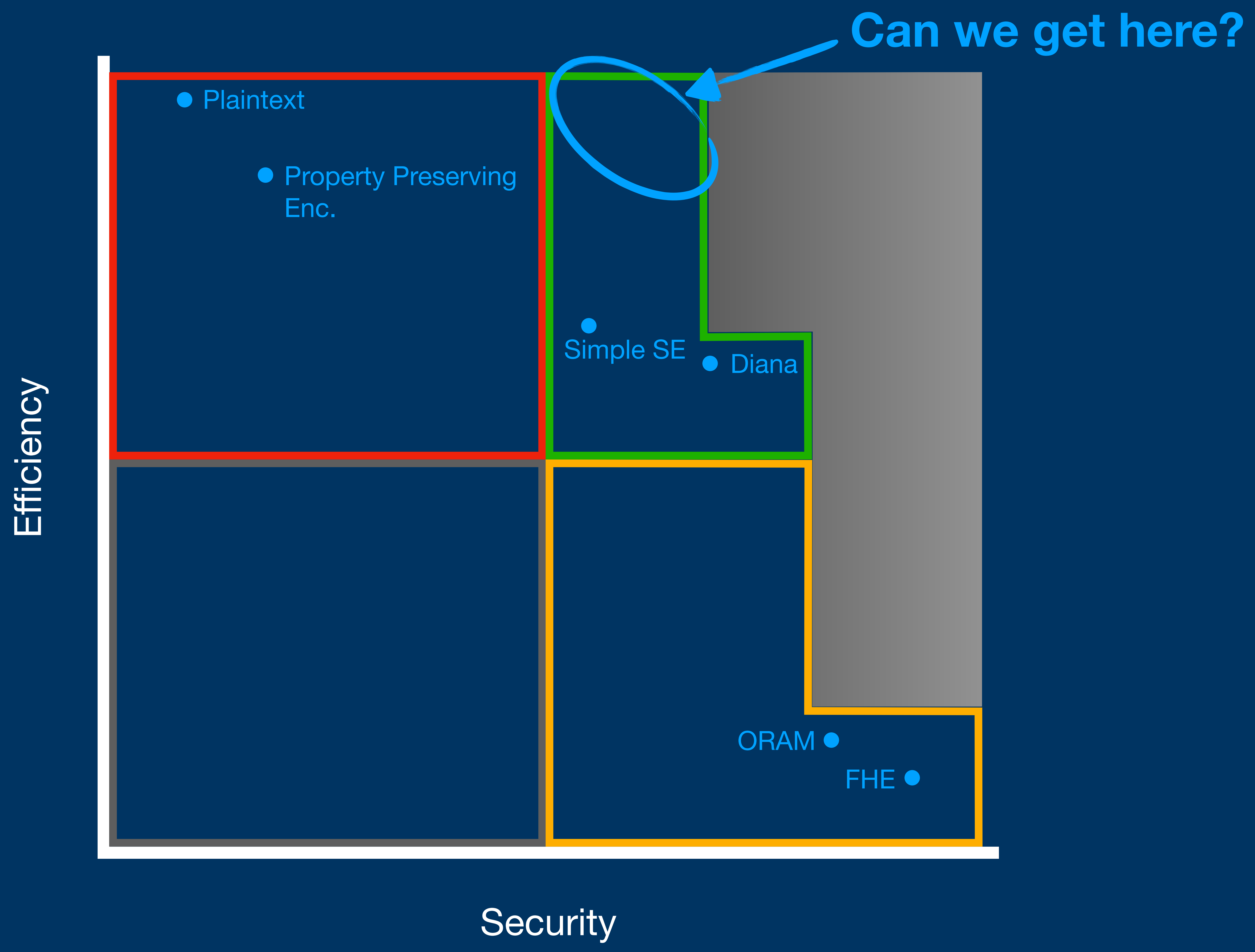
Forward Privacy

- Good news: we know how to do it at a small cost (see Σοφος or Diana)

  ⚠️ but there is also a lower bound on the efficiency of such schemes



Efficiency
Plaintext
Property Preserving Enc.
Simple SE
Diana
ORAM
FHE
Security


Can we get here?
Plaintext
Property Preserving Enc.
Simple SE
Diana
ORAM
FHE
Efficiency
Security

Practical Efficiency

- We mostly focused on the asymptotical complexity (comp. & comm.), but this is not enough.
- On hard drives, locality of accesses is important.

Cleartext DB

One (random) access

Simple SSE

n_w random accesses

Practical Efficiency

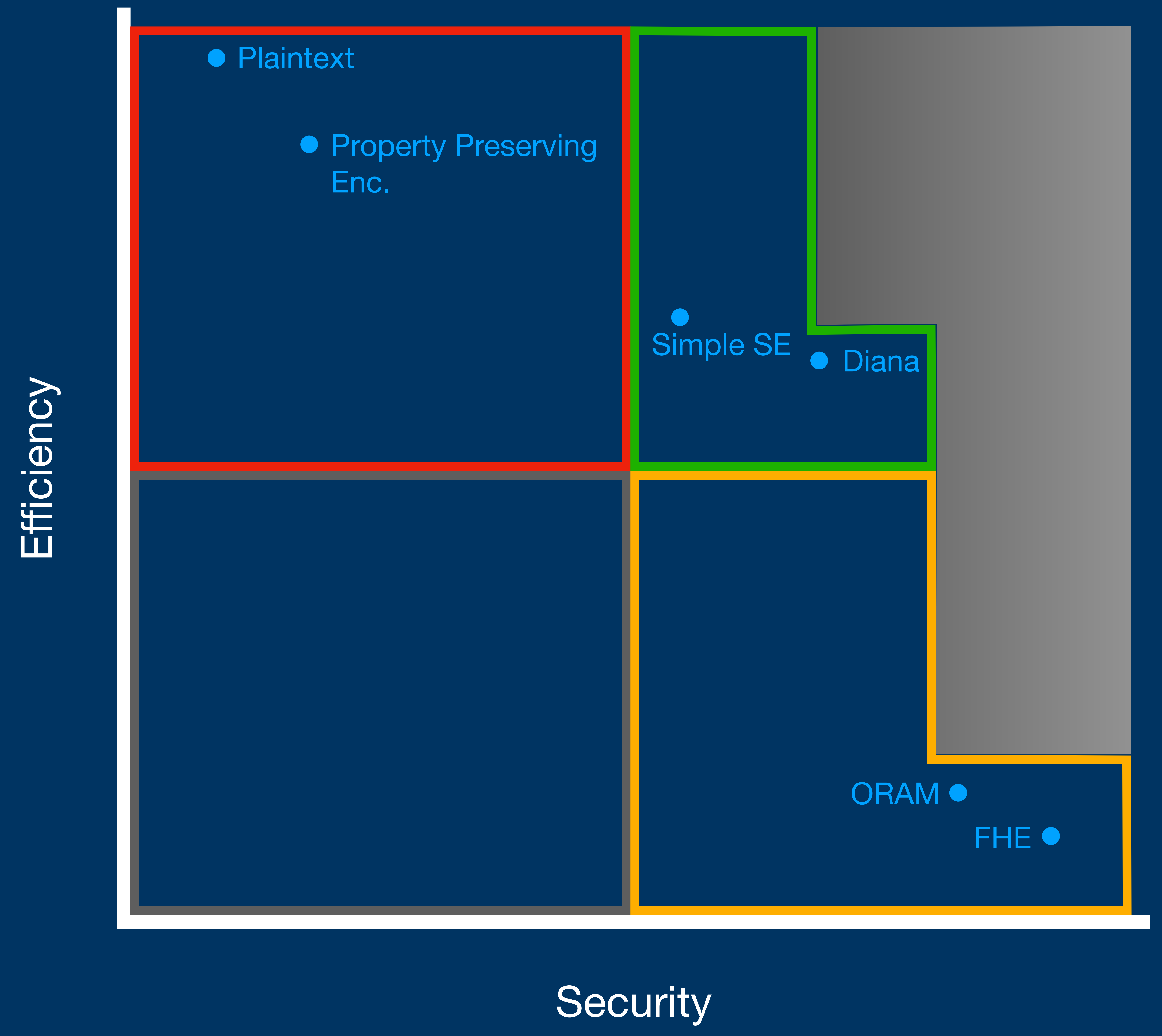
Locality

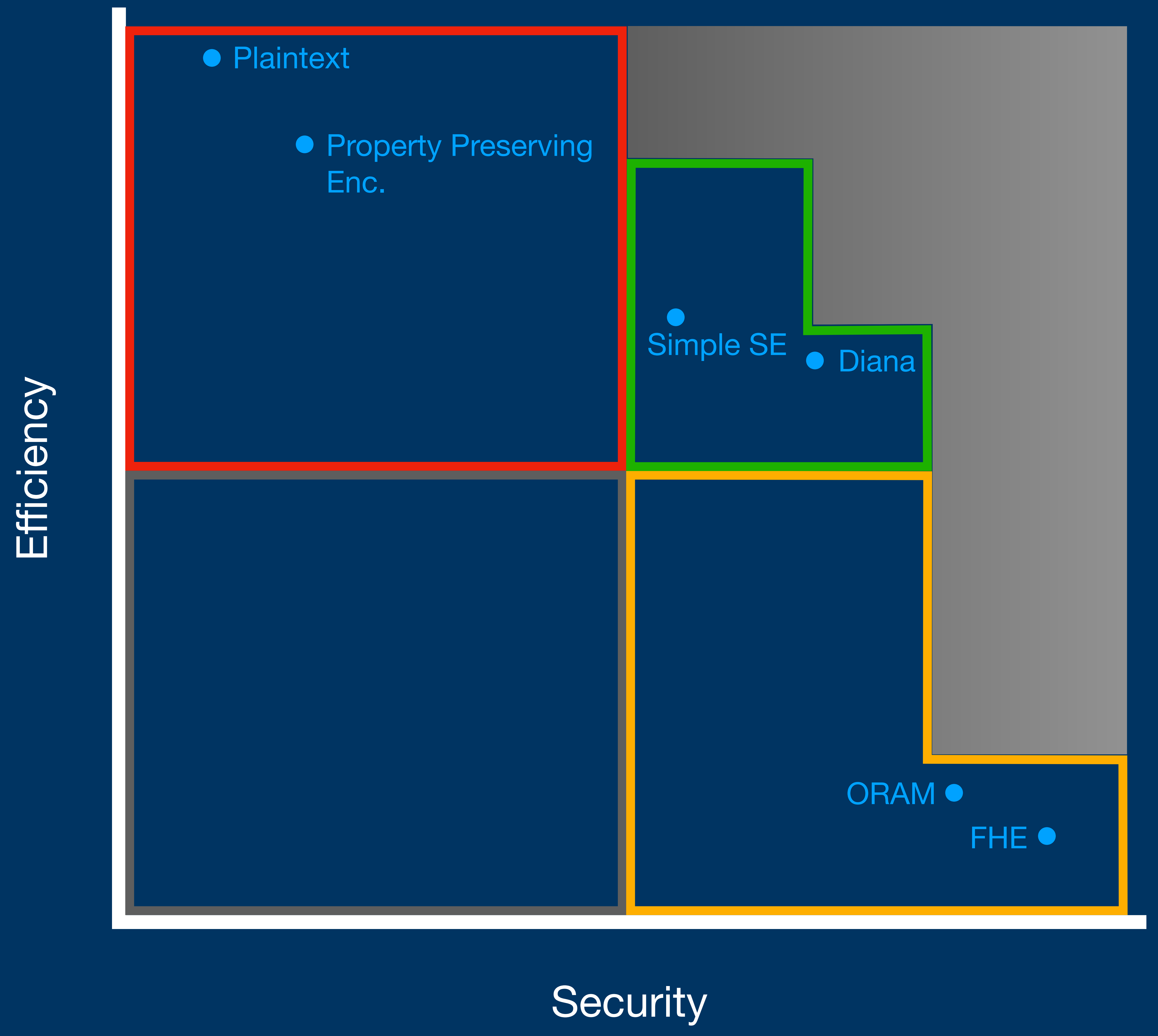
- Making many accesses is **very** costly

Action	Latency
4kB read (HDD)	6 ms
RSA SK Operation	1 ms
RSA PK Operation	0.05 ms
ECC exponentiation	0.2 ms
PRF Evaluation	300 ns

- It is worth reading more than necessary to avoid some accesses: reading once $O(\log N)$ bytes is better than reading $O(\log \log N)$ times $O(1)$ bytes.

- No free lunch ☹️ :
  [CT'14] Constant locality & constant read efficiency implies $\omega(N)$ storage.


Plaintext
Property Preserving Enc.
Simple SE
Diana
ORAM
FHE
Efficiency
Security


Plaintext
Property Preserving Enc.
Simple SE
Diana
ORAM
FHE
Efficiency
Security

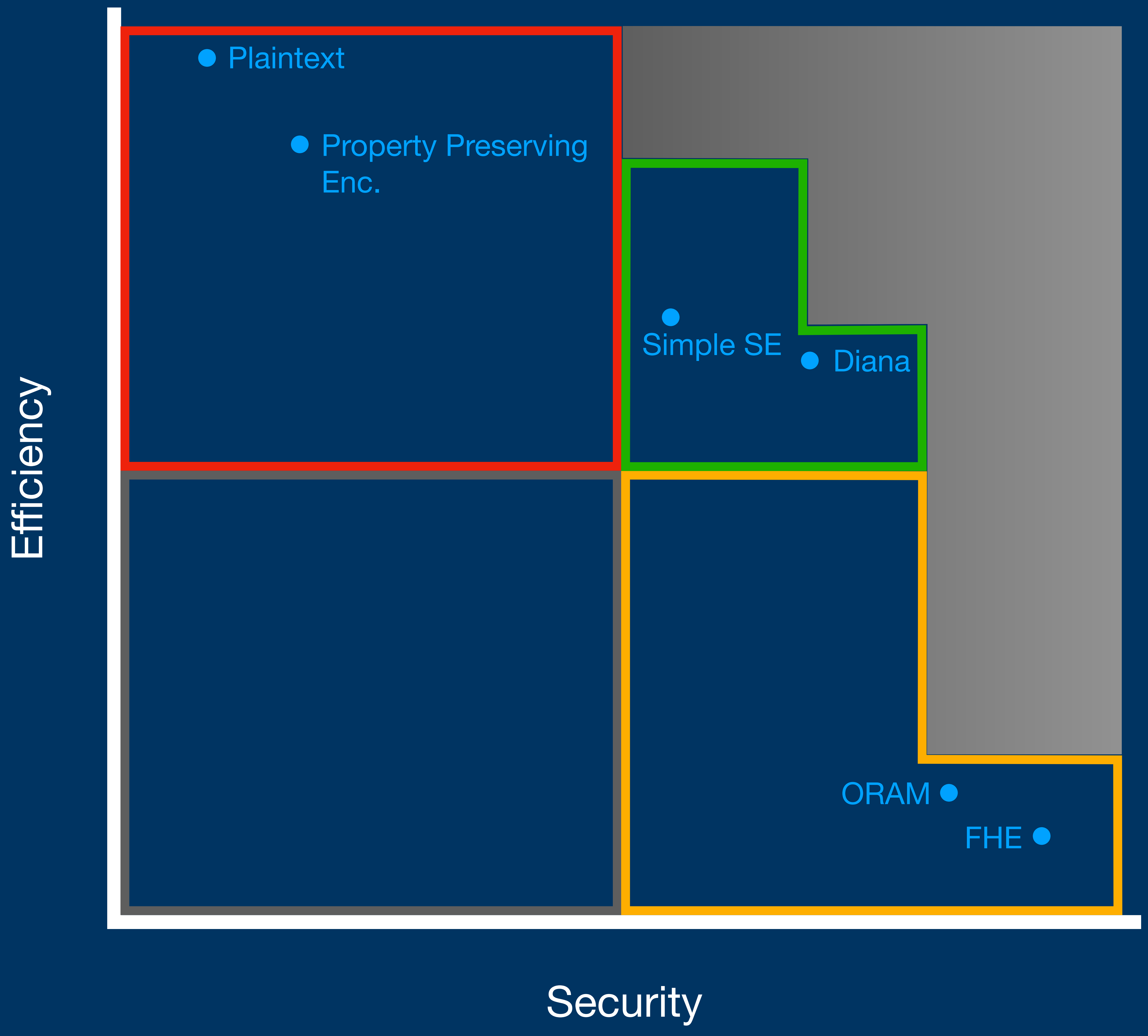
Practical Efficiency

SSDs

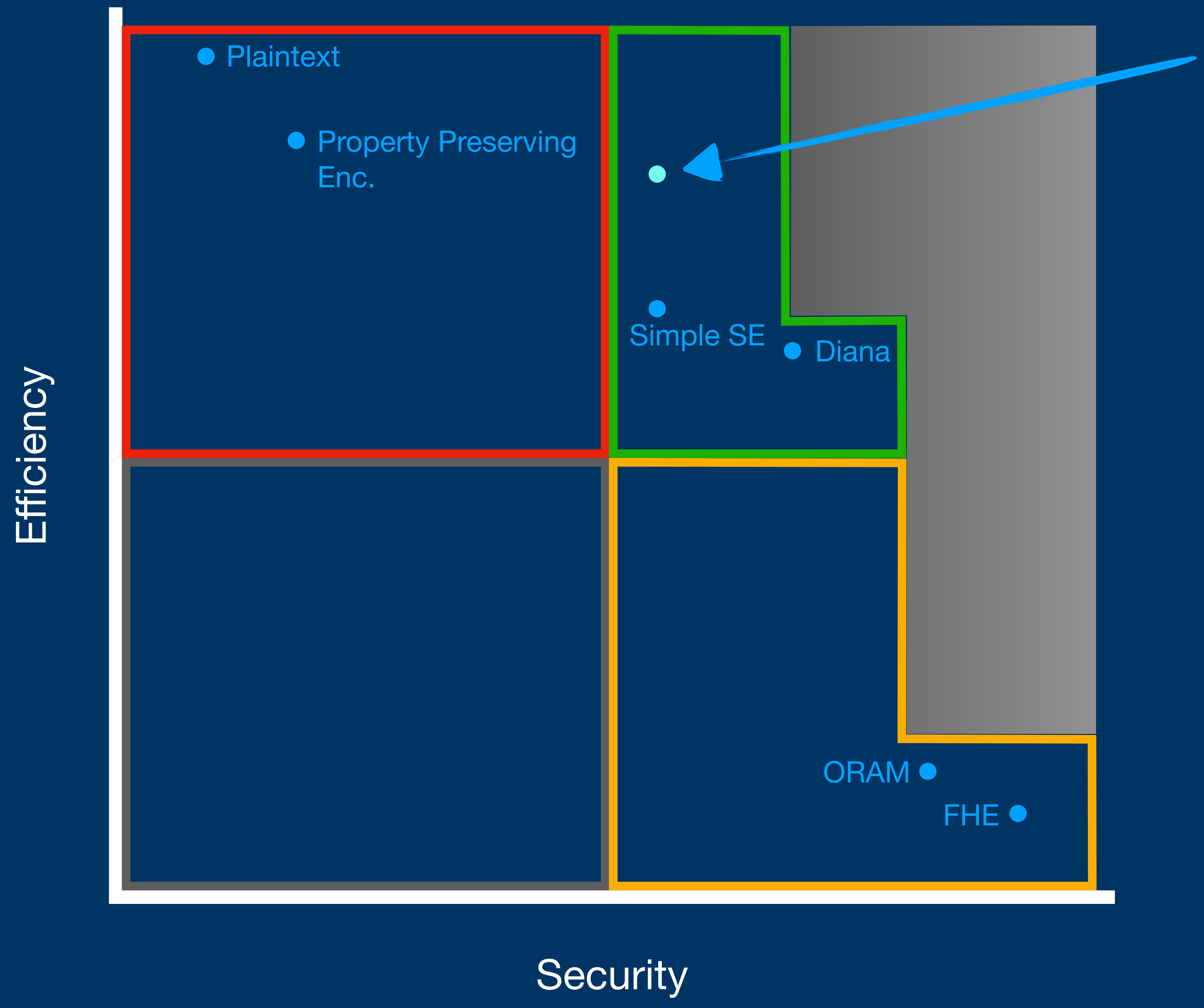
- Cool guys use flash memory now!

Action	Latency
4kB read (SSD)	0.1 ms
RSA SK Operation	1 ms
RSA PK Operation	0.05 ms
ECC exponentiation	0.2 ms
PRF Evaluation	300 ns

- SSDs are not local at all! There is built-in parallelism.
- Locality is no longer the right metric. Focus on the # of read pages.
- The previous lower bound no longer applies 😀



Plaintext
Property Preserving Enc.
Simple SE
Diana
ORAM
FHE
Efficiency
Security



Under submission

Throughput half a raw read of the results (on a SSD)

Recipe:

- mix a systems-oriented approach, …
- a pinch of cryptography, …
- a lot of algorithmic, …
- a spoon of statistics, …
- shake everything, …
- and implement the result in your favorite language (C/C++/Rust)

Conclusion

- It is hard (sometimes impossible) to combine efficiency, features and security
- A lot of improvements have been made in the knowledge of SE:
  - Better security models and constructions
  - Better understanding of attacks
  - Practical implementations
- What about a large scale adoption?

Conclusion

What about a large scale adoption?

- Probably still too inefficient for large scale databases (think TB)
- Not suited for complex queries yet (think SQL)

- Maybe we are asking for too much security? 🤔
- Basic database encryption would higher the cost of database theft (memory dumps are hard) and prevent 90% of today's leaks

0% of leaked databases were encrypted

Questions?

Slides: https://raphael.bost.fyi/publications/
Code: https://github.com/opensse/