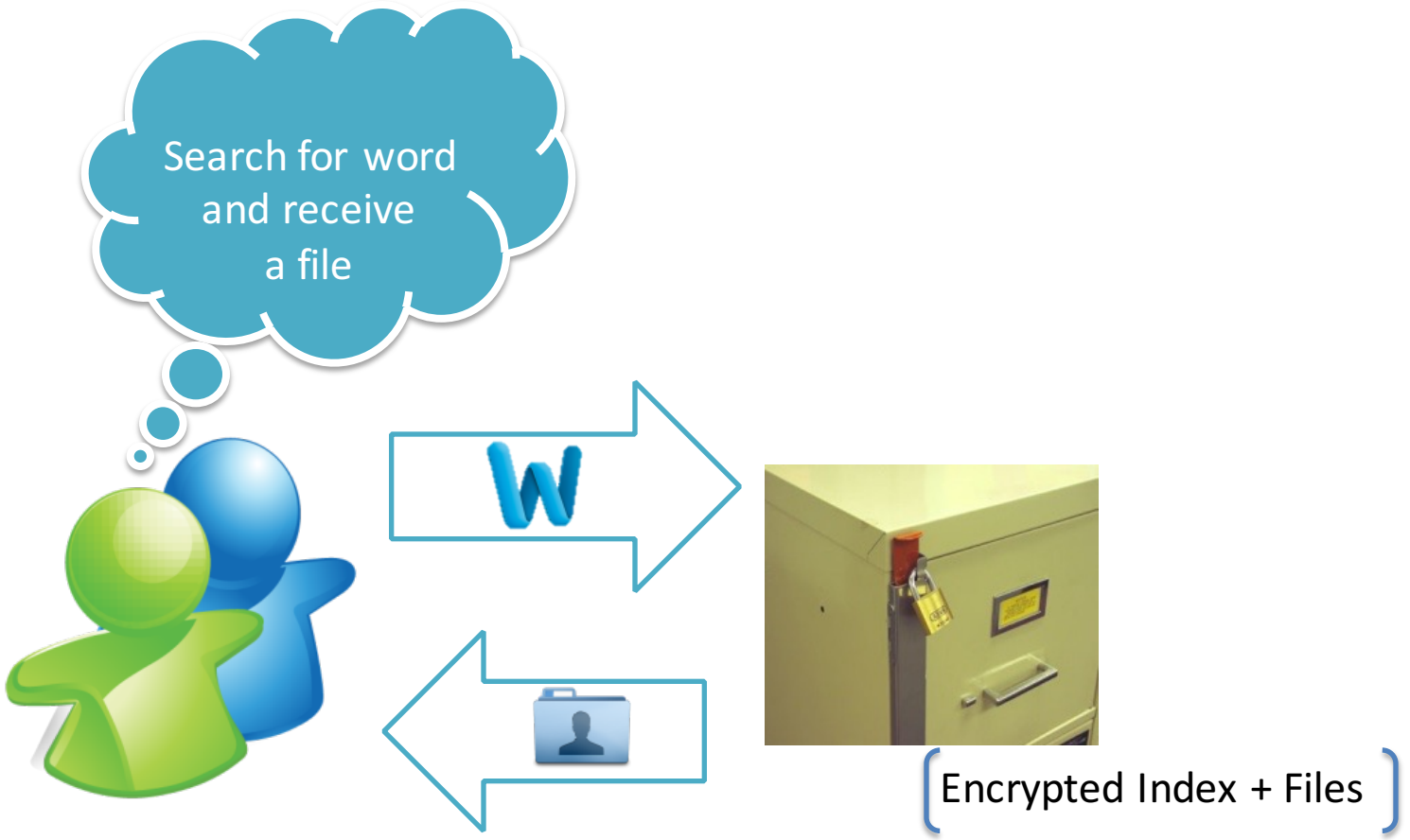
Constant-Time Dynamic Symmetric Searchable Encryption from Constrained Functional Encryption

Prof. Dr. Sebastian Gajek
NEC Research Labs and FUAS

is a game changer

Searchable Encryption (SENC)



Designing a SENC system means to juggle with requirements

Privacy

- Pattern (Information leaked)
  - **One-Time**[1] **(weakest):** Database learns after first query the search token
  - **Search:** Database does not learn that same search word is queried
  - **Access**[2] **(strongest):** Database does not learn that same data is queried

- Attack Model (Database is our foe)
  - **Honest-but-curious (weakest)**: Honest, but tries to infer information from protocol executions (passive)
  - **Covert**: When dishonest, some odds to detect curiosity (rational)
  - **Malicious (strongest)**: Dishonest, tries everything to derive information (active)

[1] Curtmola et al. Symmetric Searchable Encryption, CCS´06

[2] Goldreich, Ostrovsky: Software Protection and Simulation on Oblivious RAMs, STOC'95

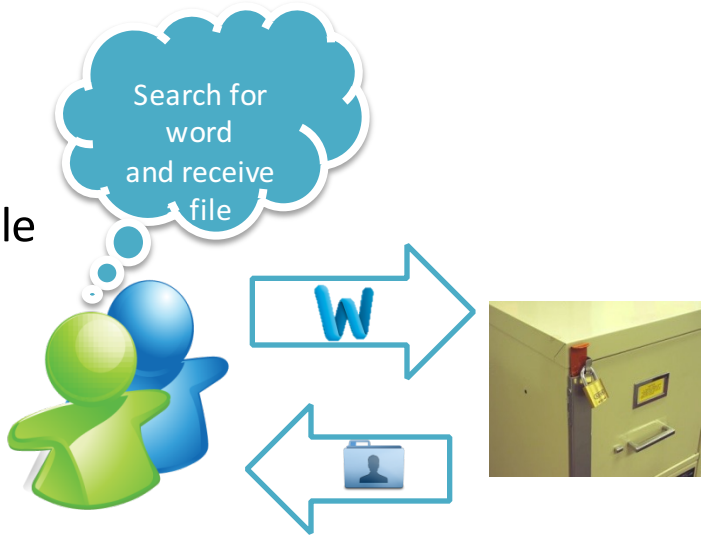
Functionality

- Database
  - **Static:** Database supports search and retrieval of (multiple) files
  - **Dynamic:** Database supports addition and deletion of (multiple) words and files

- Query language
  - **Single-word**: each token allows for searching for a single word
  - **Multi-word**: each token allows for searching multiple words (ideally, query for some CNF/DNF formula)
  - **Nearest-word**: each token allows for searching multiple words, each word w_i close to some word w_j, i.e. $|w_i - w_j| < \varepsilon$ (e.g., range queries)

Scalability

- Performance
  - **Non-parallelizable:** no gain by sharing search over multiple clouds
  - **Parallelizable:** performance gain by multiple cloud
- Generality
  - **Specific**: SENC system is a mash-up of cryptographic algorithms and data structures (e.g. CryptDB)
  - **Self-containted**: SENC system is a framework of cryptographic algorithms and data structures



Our Schemes

Privacy	Honest-Curious	Covert	Malicious
One-Time	✓	✓	✓
Search	✓	✓	✓
Access	✗	✗	✗

Functionality	Single-Word	Multiple-Word	Nearest-Word
Search/Retrieve	✓	✓	✓
Add/Delete words	✓	✓	✓
Add/Delete files	✓	✓	✓

Scalability	Parallelizable	Self-Contained	
Performance	✓		
Generality		✓	

Key Idea(s)

- A searchable encryption **framework**
  - Cryotographic layer
    - provides functionality and privacy
    - implemented by constrained functional encryption (for inner-product functions)
  - Data (structure) layer
    - provides functionality and scalability
    - implements search on trees, (unlinked) lists, matrices, graphs, …

(Constrained) Functional Encryption

C:=Encrypt(MSK, x)

C

SK_f:=KeyGen(MSK, f)

SK_f

f(x):=Decrypt(SK_f, x)

(Constrained) Functional Encryption

C:=Encrypt(MSK, x)

C

SK_f:=KeyGen(MSK, f)

SK_f

f(x):=Decrypt(SK_f, C)

SK_f **constrained** to decrypt a particular ciphertext

Our Result

Assume the subgroup membership problem holds, then there exists a secure[1] constrained functional encryption system for the class of inner product functions

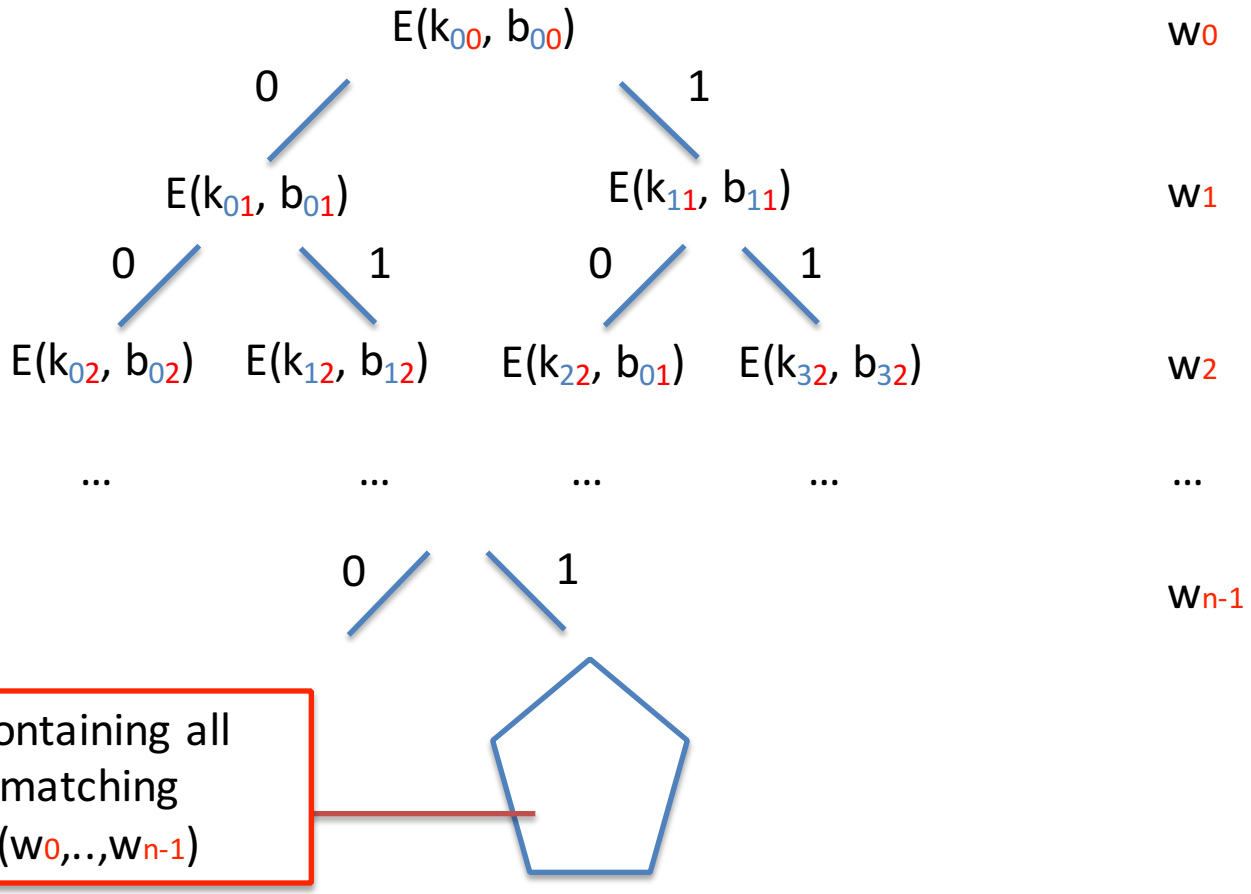
[1] Security game similar to predicate-private encryption [Shi-Waters, TCC'09]

Scheme #1

Data Structure: Binary Tree

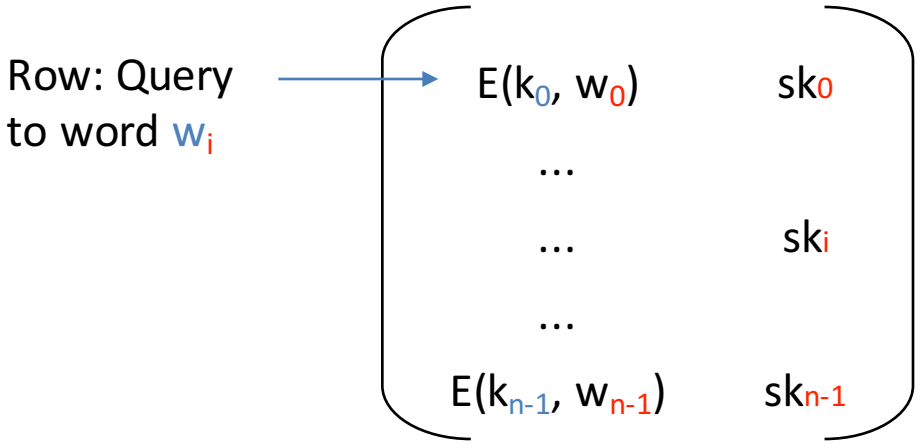
Technique (1)

- Binary tree of depth log|W|=n



Our Technique (2)

- Search query q for single $w=(w_0,..,w_{n-1})$



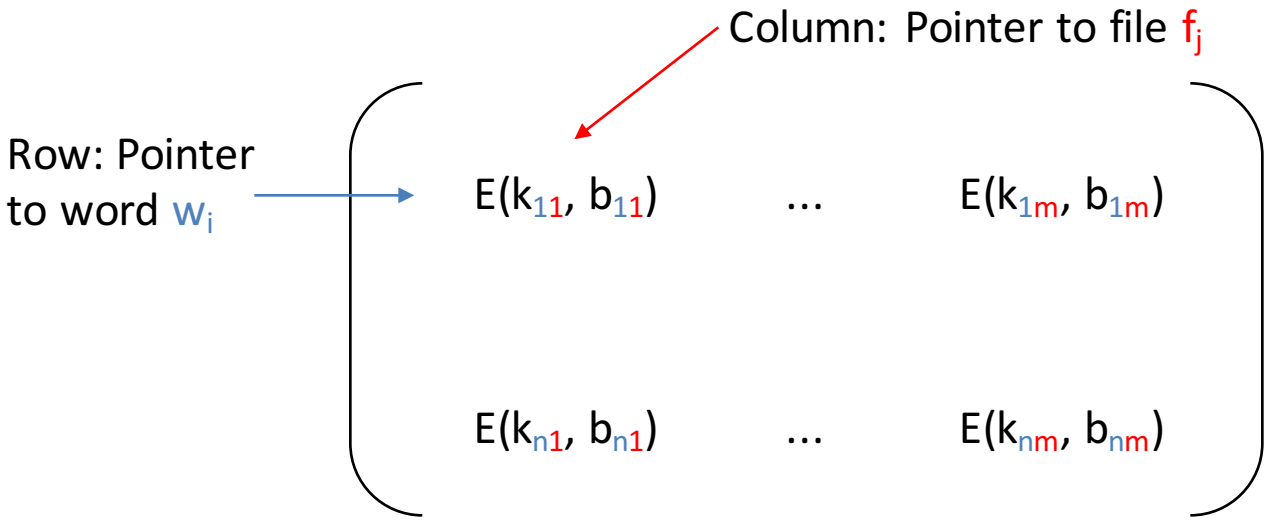
- Decryption key $sk_i=k_i*k_i$
  - Can only decrypt if correct evaluation of product performed

Scheme #2

Data Structure: (Unlinked) List

Our Technique (1)

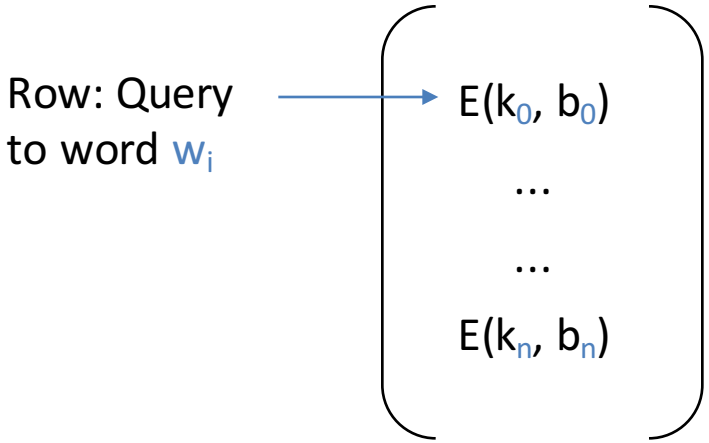
- Database look-up Matrix M^{nxm}



$i \leq n, j \leq m$: $b_{ij}=1$ if and only if f_j contains word w_i

Our Technique (2)

- Search query q for multiple words w_i



$i \leq n$: b_i=1 if and only if we query for word w_i

Our Technique (3)

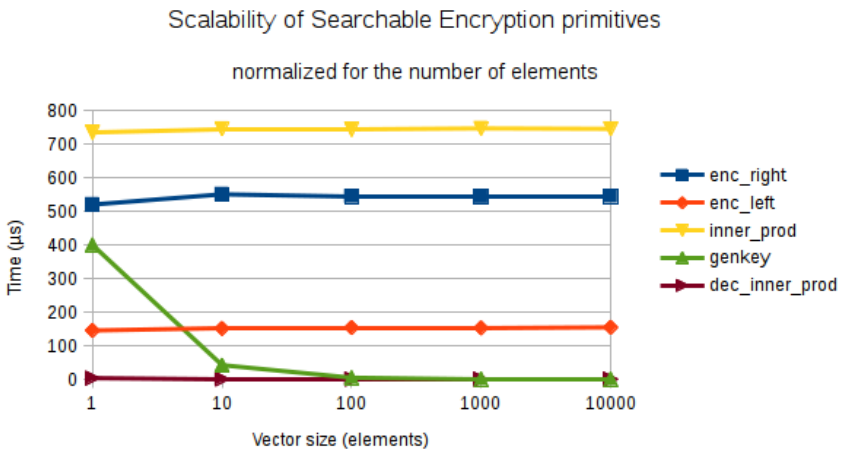
- Search on Encrypted Data
  - Matrix multiplication $p=q^T A$

$$\begin{pmatrix} E(k_0, b_0) \\ \dots \\ \dots \\ E(k_n, b_n) \end{pmatrix}^T \begin{pmatrix} E(k_{11}, b_{11}) & \dots & E(k_{1m}, b_{1m}) \\ & & \\ E(k_{n1}, b_{n1}) & \dots & E(k_{nm}, b_{nm}) \end{pmatrix}$$

  - Decryption key $sk_j=SUM(k_i * k_{ij})$
    - Can only decrypt if correct evaluation of inner product performed

Performance Evaluation

- Implemented scheme based on group G
  - of composite order N=pqr and symmetric pairing
  - of prime order N=p and asymmetric pairing



Group	Ciphertext Size	Search word	Security
N=pqr	***3076 bit***	***1.23 s***	***128 bit***
N=p	***2*254 bit***	***0.73 ms***	***128 bit***

Conclusion

- Searchable encryption requires to find a trade-off between privacy, functionality and scalability
- Our protocol framework is tailored towards privacy and functionality
- Yet many optimisations are not explored (e.g. clustering of matrices)

RSAConference2016
San Francisco | February 29 – March 4 | Moscone Center

Connect to Protect

SESSION ID: CRYP-T11

Private Large-Scale Databases with Distributed Searchable Symmetric Encryption

Authors:
Yuval Ishai
Eyal Kushilevitz
Steve Lu
Rafail Ostrovsky

Speaker:
Steve Lu
Senior Researcher
Stealth Software Technologies, Inc.

A Story

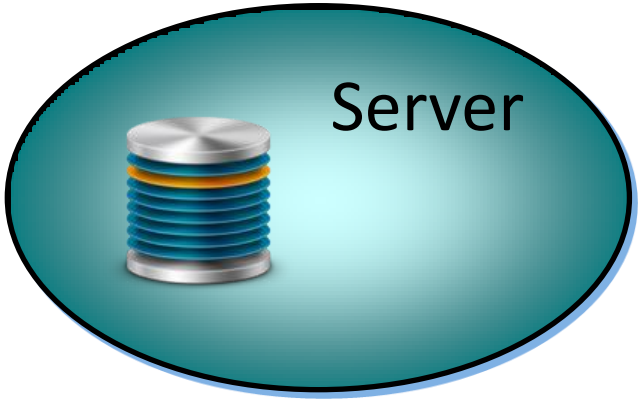
- In 2004...

A Story

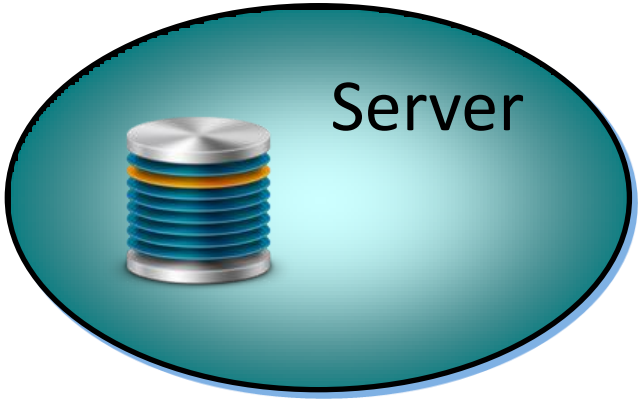
- In 2004...

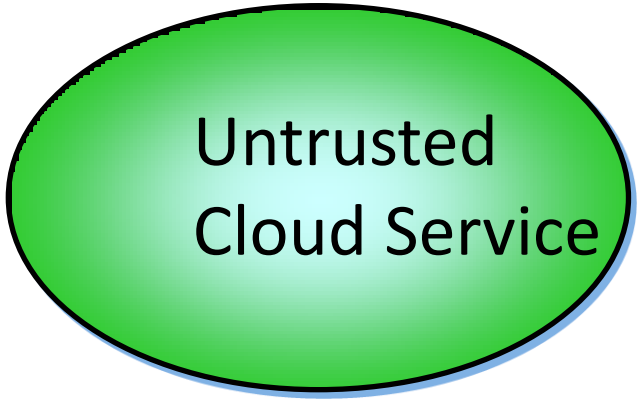
- Whatever happens in Vegas stays in Vegas?

Motivating Problem

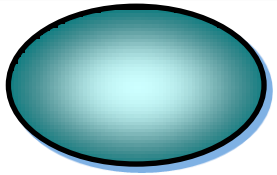
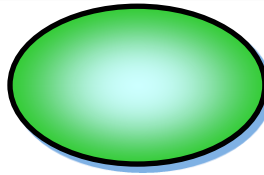


Motivating Problem



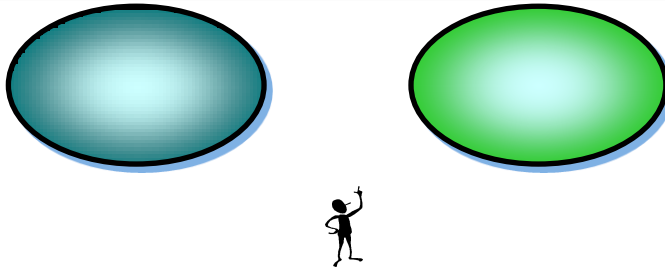


Privacy Goals

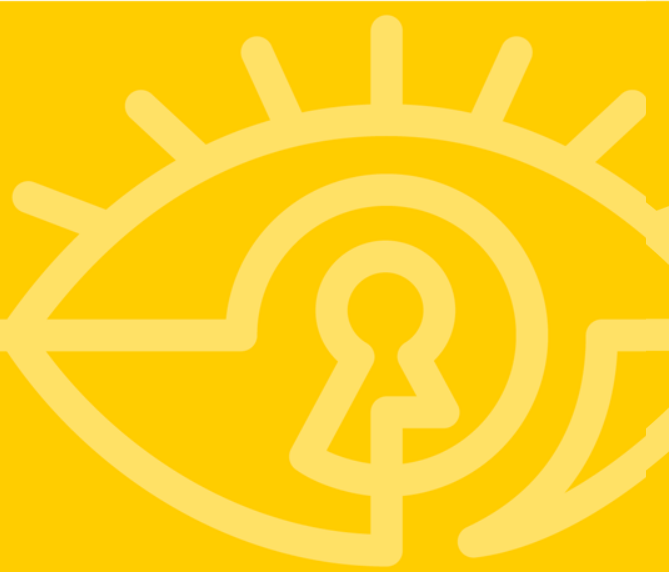




- Protect the privacy of the Server holding the Database (from Cloud and Client)
- Protect the privacy of Client queries (from Cloud and Server)
- Server can specify private policies that enforce queries
- Modeled as a 3-party computation, with security in the semi-honest (honest-but-curious) model against any single corrupted party

Additional Goals

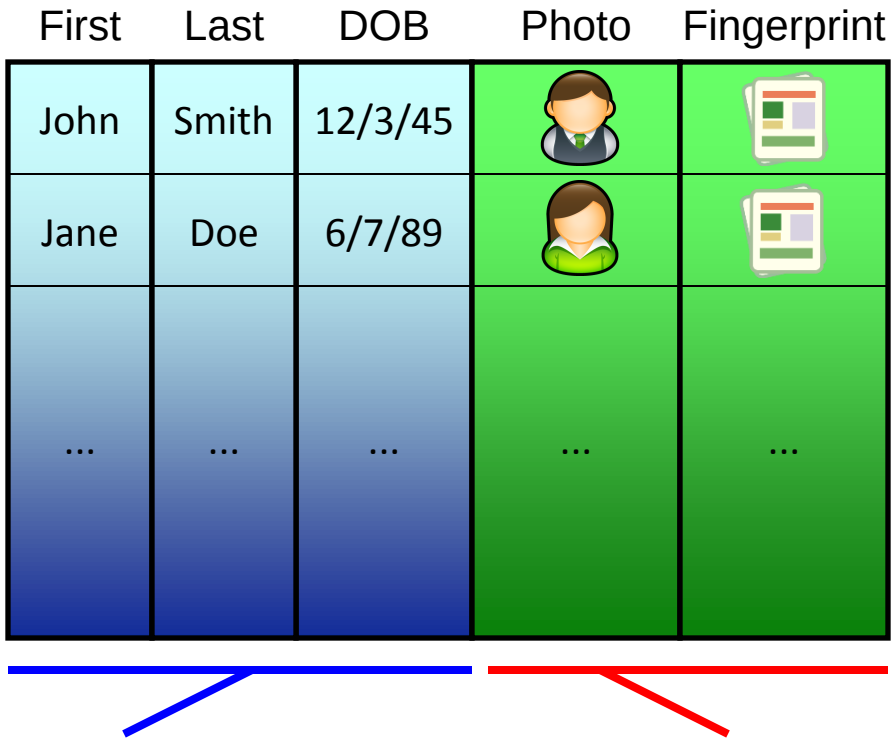


- Query Functionality
- Leverages the Cloud
- Scales! Goes Fast! ...



Background and Model

Data Format


First
Last
DOB
Photo
Fingerprint
John
Smith
12/3/45
Jane
Doe
6/7/89
...
...
...
...
...
Searchable Attributes
(First, Last, DOB, ...)
Rich Data (Photo, Fingerprint, ...)

Review of Crypto Primitives

Name	Properties	Complexity	Leakage
Private Information Retrieval(PIR) [Chor-Kushilevitz-Goldreich-Sudan95, Kushilevitz-Ostrovsky97,...]			
Oblivious RAM(ORAM) [Goldreich-Ostrovsky96,...]			
Searchable Symmetric Encryption(SSE) [Curtmola-Garay-Kamara-Ostrovsky06,...]			

Review of Crypto Primitives

Name	Properties	Complexity	Leakage
Private Information Retrieval(PIR) [Chor-Kushilevitz-Goldreich-Sudan95, Kushilevitz-Ostrovsky97,...]	Privately fetch index i from array	Low comm. Linear server work, mostly bitwise ops	Sizes only
Oblivious RAM(ORAM) [Goldreich-Ostrovsky96,...]			
Searchable Symmetric Encryption(SSE) [Curtmola-Garay-Kamara-Ostrovsky06,...]			

Review of Crypto Primitives

Name	Properties	Complexity	Leakage
Private Information Retrieval(PIR) [Chor-Kushilevitz-Goldreich-Sudan95, Kushilevitz-Ostrovsky97,...]	Privately fetch index i from array	Low comm. Linear server work, mostly bitwise ops	Sizes only
Oblivious RAM(ORAM) [Goldreich-Ostrovsky96,...]	Compiles program into one that hides data and access	Low comm. polylog work, symmetric-key ops	Sizes only
Searchable Symmetric Encryption(SSE) [Curtmola-Garay-Kamara-Ostrovsky06,...]			

Review of Crypto Primitives

Name	Properties	Complexity	Leakage
Private Information Retrieval(PIR) [Chor-Kushilevitz-Goldreich-Sudan95, Kushilevitz-Ostrovsky97,...]	Privately fetch index i from array	Low comm. Linear server work, mostly bitwise ops	Sizes only
Oblivious RAM(ORAM) [Goldreich-Ostrovsky96,...]	Compiles program into one that hides data and access	Low comm. polylog work, symmetric-key ops	Sizes only
Searchable Symmetric Encryption(SSE) [Curtmola-Garay-Kamara-Ostrovsky06,...]	Search on encrypted data	Very low comm/comp, symmetric-key ops	Sizes and some access "metadata"

Recent Advances

- (2-server) PIR
  - Subpolynomial [Dvir-Gopi15]
  - Distributed Point Functions [Gilboa-Ishai14,Boyle-Gilboa-Ishai15]
- ORAM
  - Asymptotic [Kushilevitz-Lu-Ostrovsky12]
  - Circuit complexity [Wang-Chan-Shi15]
  - Distributed [Lu-Ostrovsky13A]
  - Non-interactive (Garbled RAM) [Lu-Ostrovsky13B, Gentry-Halevi-Lu-Ostrovsky-Raykova-Wichs14, Garg-Lu-Ostrovsky-Scafuro15, Garg-Lu-Ostrovsky15]
- Searchable Symmetric Encryption
  - Large Scale [Cash-Jarecki-Jutla-Krawczyk-Rosu-Steiner13, Cash-Jaeger-Jarecki-Jutla-Krawczyk-Rosu-Steiner14, Pappas-Krell-Vo-Kolesnikov-Malkin-Choi-George-Keromytis-Bellovin14, Faber-Jarecki-Karwczyk-Nguyen-Rosu-Steiner15, Fisc-Vo-Krell-Kumarasubramanian-Klesnikov-Malkin-Bellovin15]
  - Dynamic [Kamara-Papamanthou-Roeder12, Kamara-Papamanthou13, Naveed-Prabhakaran-Gunter14]

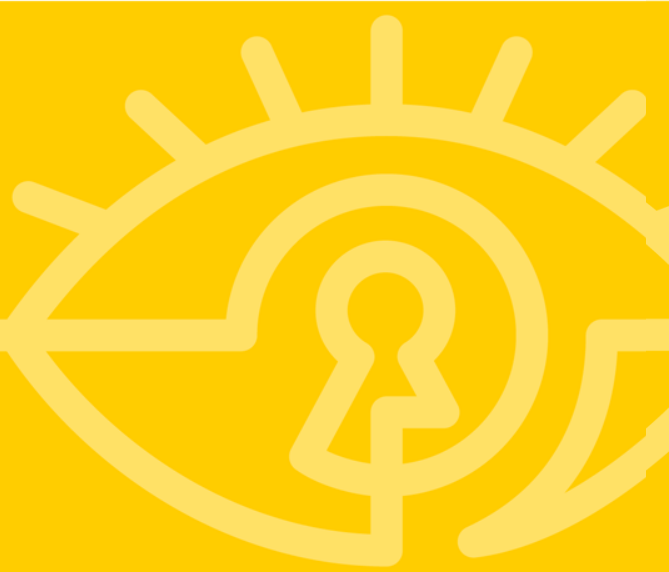
Recent Advances

- (2-server) PIR
  - Subpolynomial [Dvir-Gopi15]
  - Distributed Point Functio
- ORAM
  - Asymptotic [Kushilevitz-
  - Circuit complexity [Wan
  - Distributed [Lu-Ostrovsk
  - Non-interactive (Garble va-Wichs14, Garg-Lu-Ostrovsky-Scafuro15, Garg-Lu-Ostr
- Searchable Symmetric Encr
  - Large Scale [Cash-Jarecki-Jutla-Krawczyk-Rosu-Steiner13, Cash-Jaeger-Jarecki-Jutla-Krawczyk-Rosu-Steiner14, Pappas-Krell-Vo-Kolesnikov-Malkin-Choi-George-Keromytis-Bellovin14, Faber-Jarecki-Karwczyk-Nguyen-Rosu-Steiner15, Fisc-Vo-Krell-Kumarasubramanian-Klesnikov-Malkin-Bellovin15]
  - Dynamic [Kamara-Papamanthou-Roeder12, Kamara-Papamanthou13, Naveed-Prabhakaran-Gunter14]

Can we leverage these advances and get the best of all worlds?

Our Results

- We create a private SSE scheme that only leaks sizes (and query types), assuming semi-honest (honest-but-curious) model with no collusion.
- Privacy is modeled via the ideal/real paradigm from secure computation literature
- Supports range, substring, Boolean,... queries
- Supports simple deny policies
- Supports updates

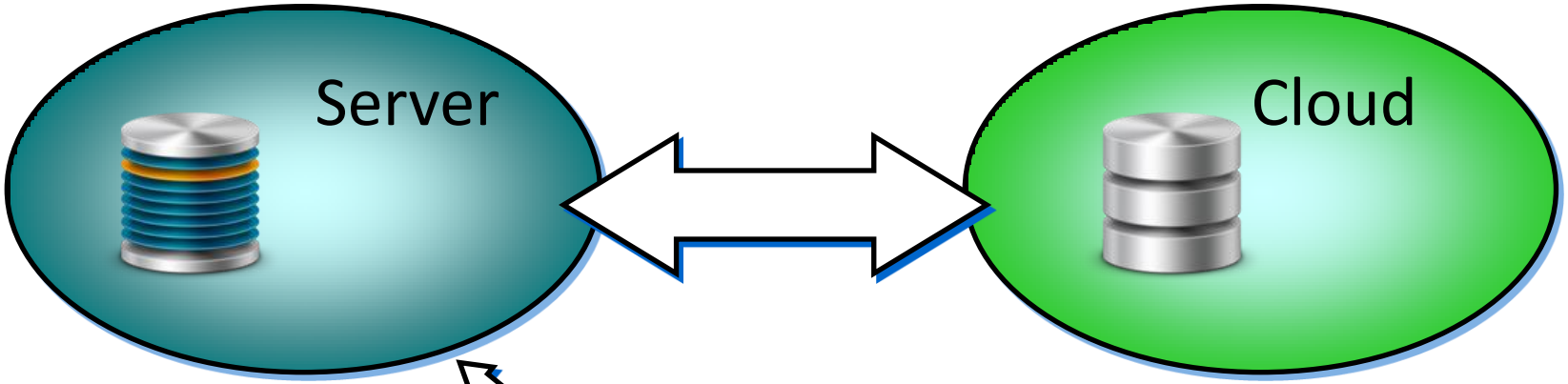


Our Construction

Overview

- Setup
- Query
- Updates
- Policy

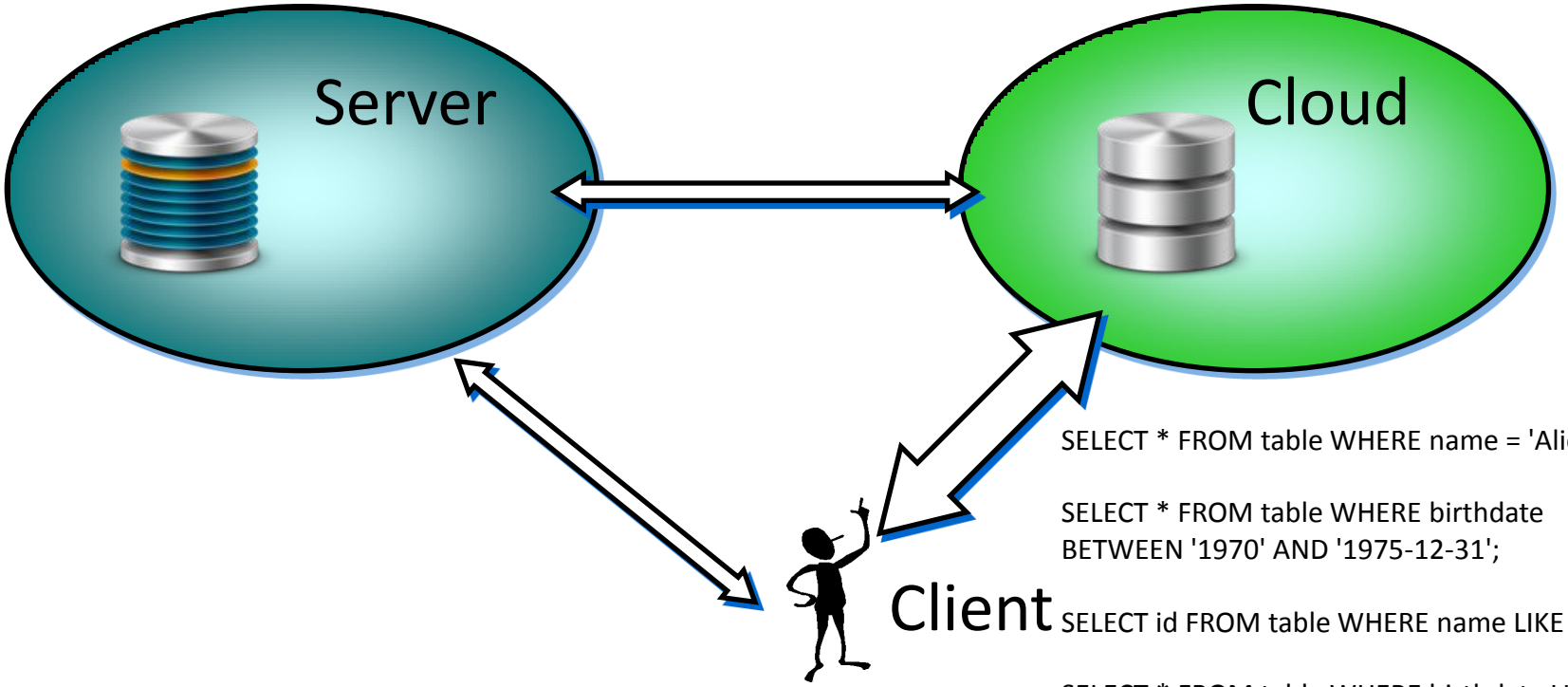
Setup




```
CREATE TABLE foo (
    id INT(8) RANGE,
    name CHAR(20) RANGE WILDCARD SUBSTRING(5),
    income INT(8) RANGE,
    birthdate CHAR(10) RANGE SUBSTRING(4),
    description CHAR(1000)
  );
```

Client

Query



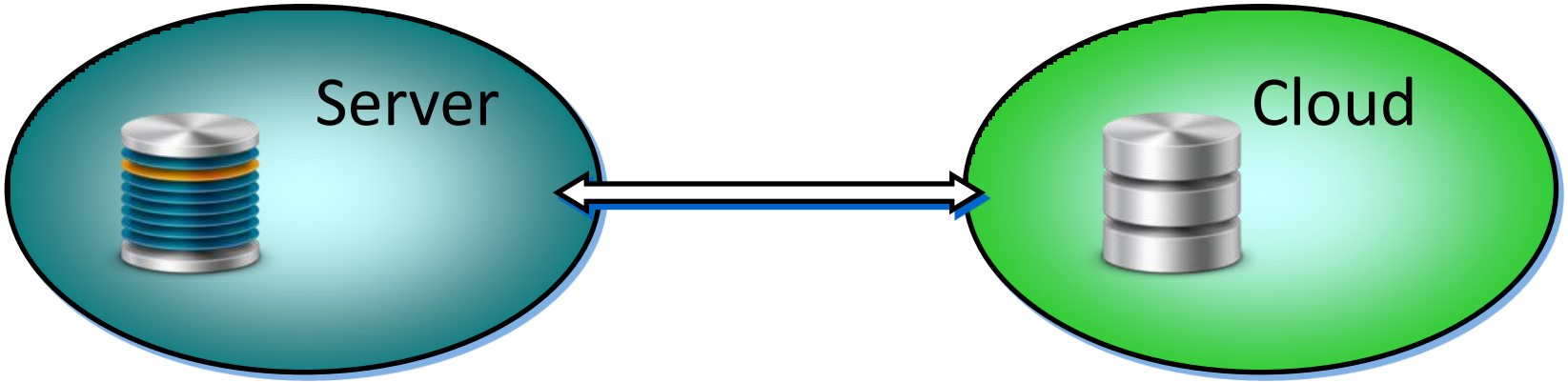

```
SELECT * FROM table WHERE name = 'Alice';

SELECT * FROM table WHERE birthdate
BETWEEN '1970' AND '1975-12-31';

SELECT id FROM table WHERE name LIKE 'Car_l';

SELECT * FROM table WHERE birthdate LIKE '%-05-%';
```

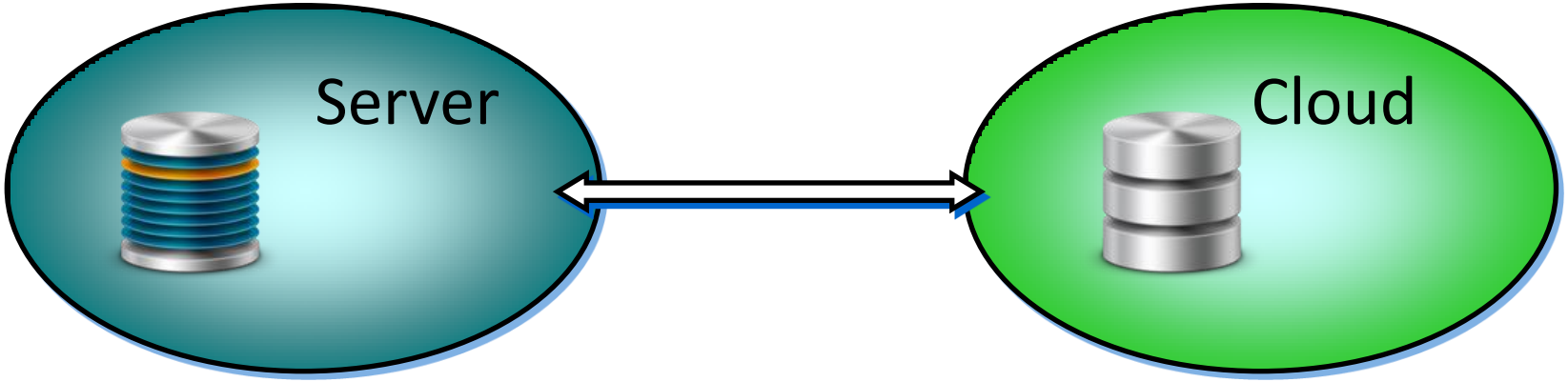
Update



INSERT INTO table VALUES
(4, 'Carol', 123456, '1970-05-17', NULL);

DELETE FROM table WHERE id = 4;

Policies



Deny all range queries

Deny wildcard queries on 'birthdate'

Deny all queries on 'gender'

High Level Idea

- Setup: Server stores encrypted database and encrypted indices (B-tree) in Cloud
- Query: Use a combination of PIR, ORAM, and secure computation techniques to traverse tree privately
- Updates: Create a mini-database and intermittently merge
- Policies: Interweave policy enforcement with query mechanism

High Level Idea

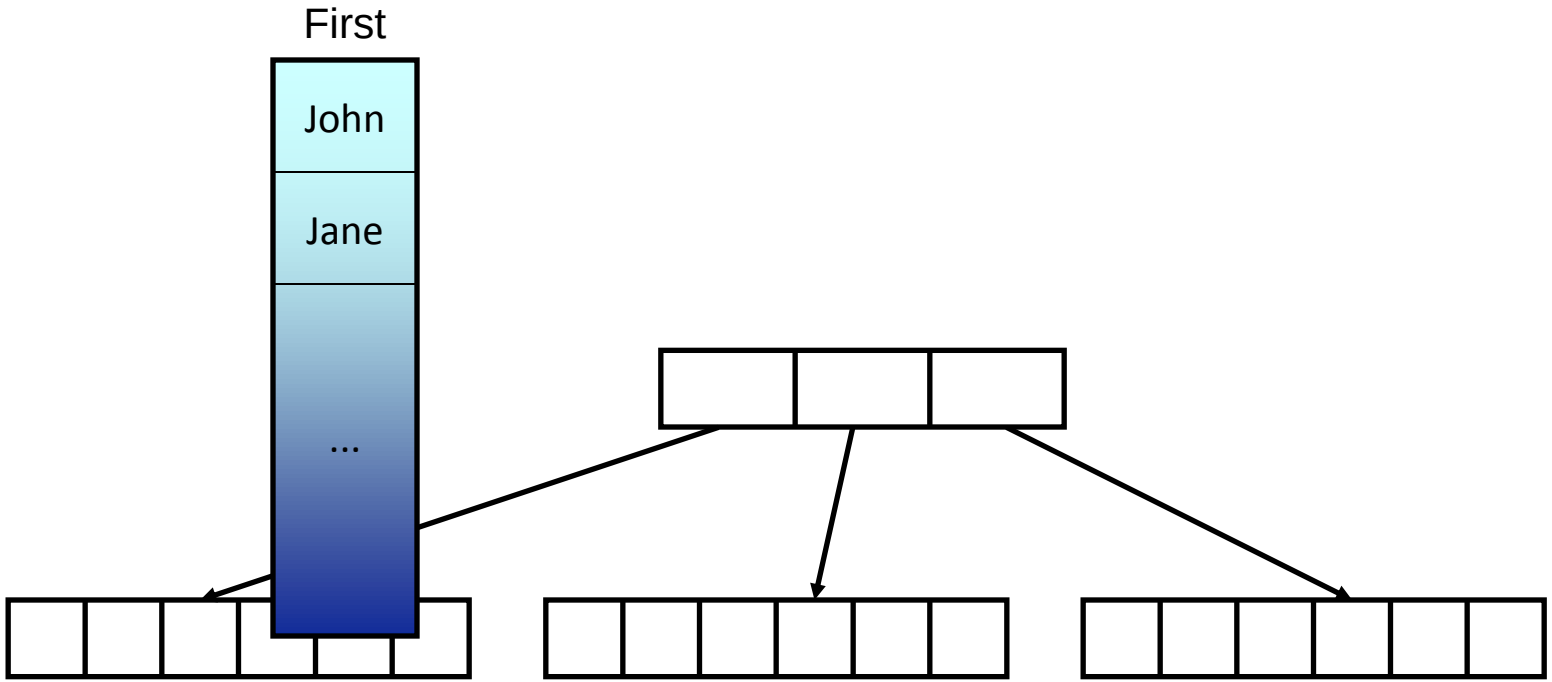
- Setup: Server stores encrypted database and encrypted indices (B-tree) in Cloud
- Query: Use a combi and secure computation tech ivately
- Updates: Create a nittently merge
- Policies: Interwea h query mechanism

Unlike outsourced storage of Client's own data: Client (and Cloud) must never see decrypted index values nor unqueried data

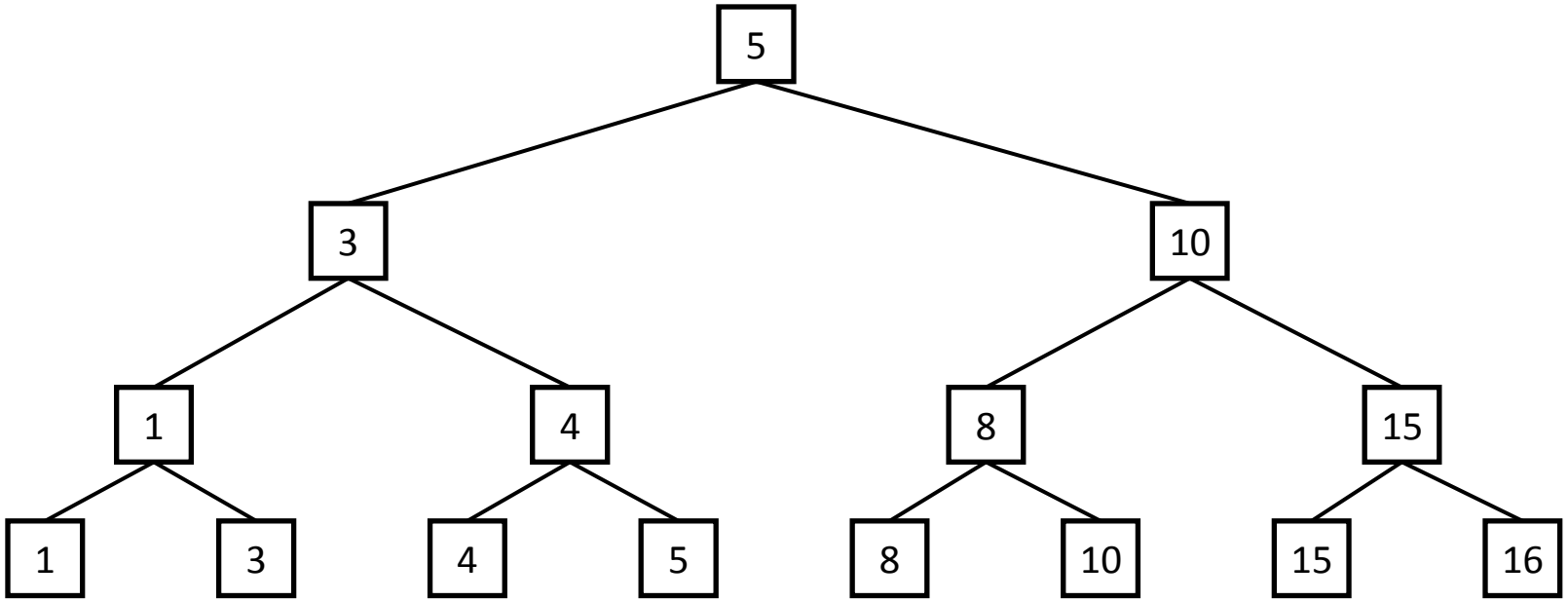
High Level Idea

- Setup: Server stores encrypted database and encrypted indices (B-tree) in Cloud
- Query: Use a combination of PIR, ORAM, and secure computation techniques to traverse tree privately
- Updates: Create a mini-database and intermittently merge
- Policies: Interweave policy enforcement with query mechanism

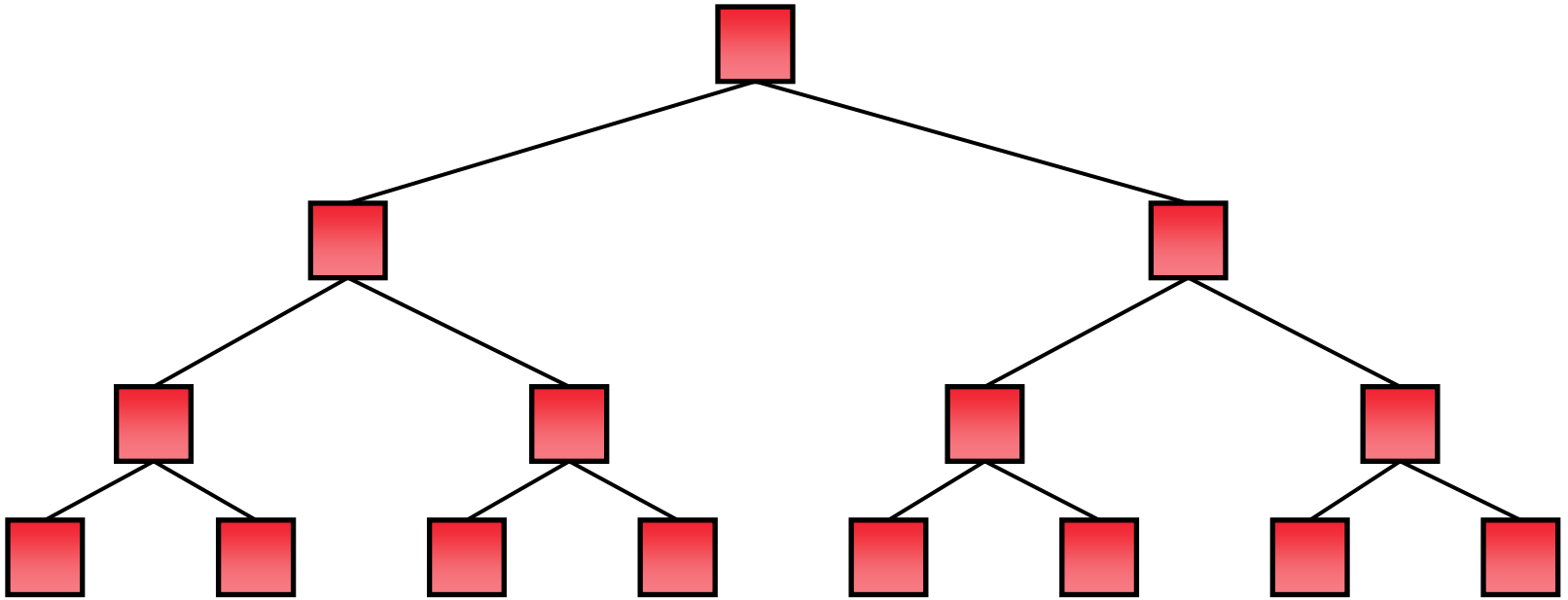
Structure of Setup



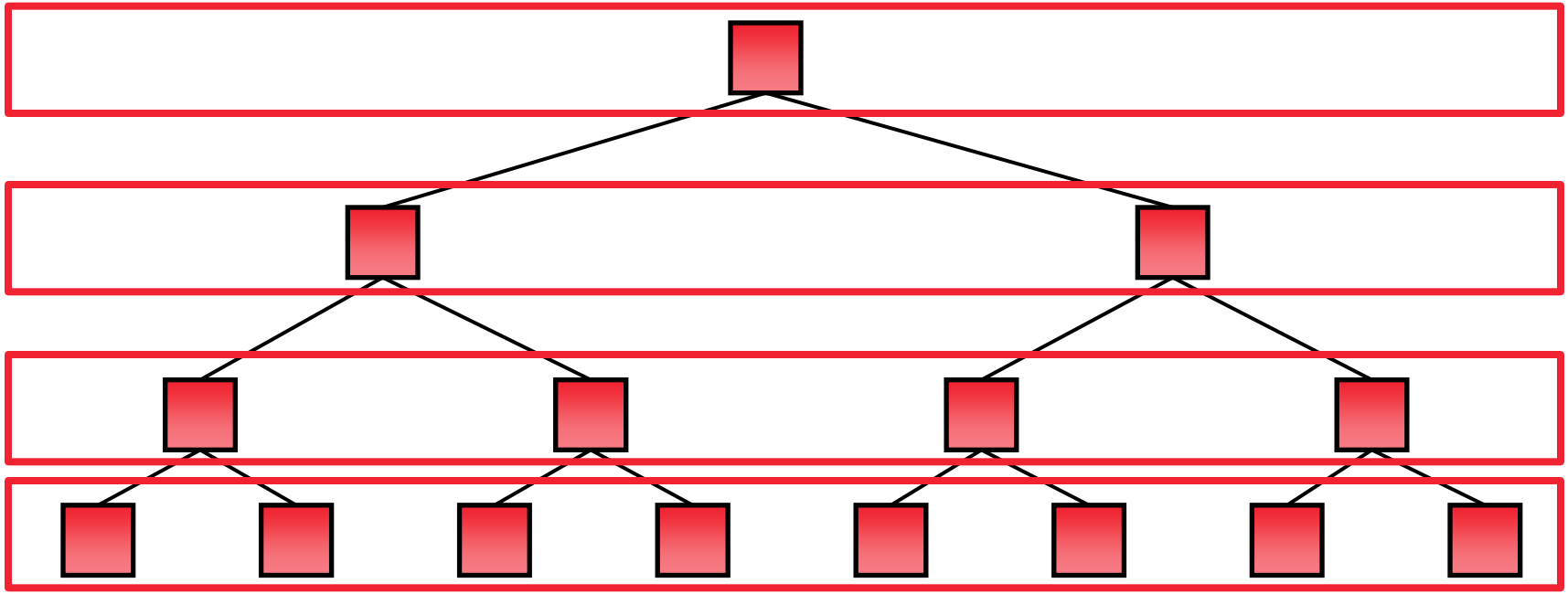
Binary Tree Example For Range Query



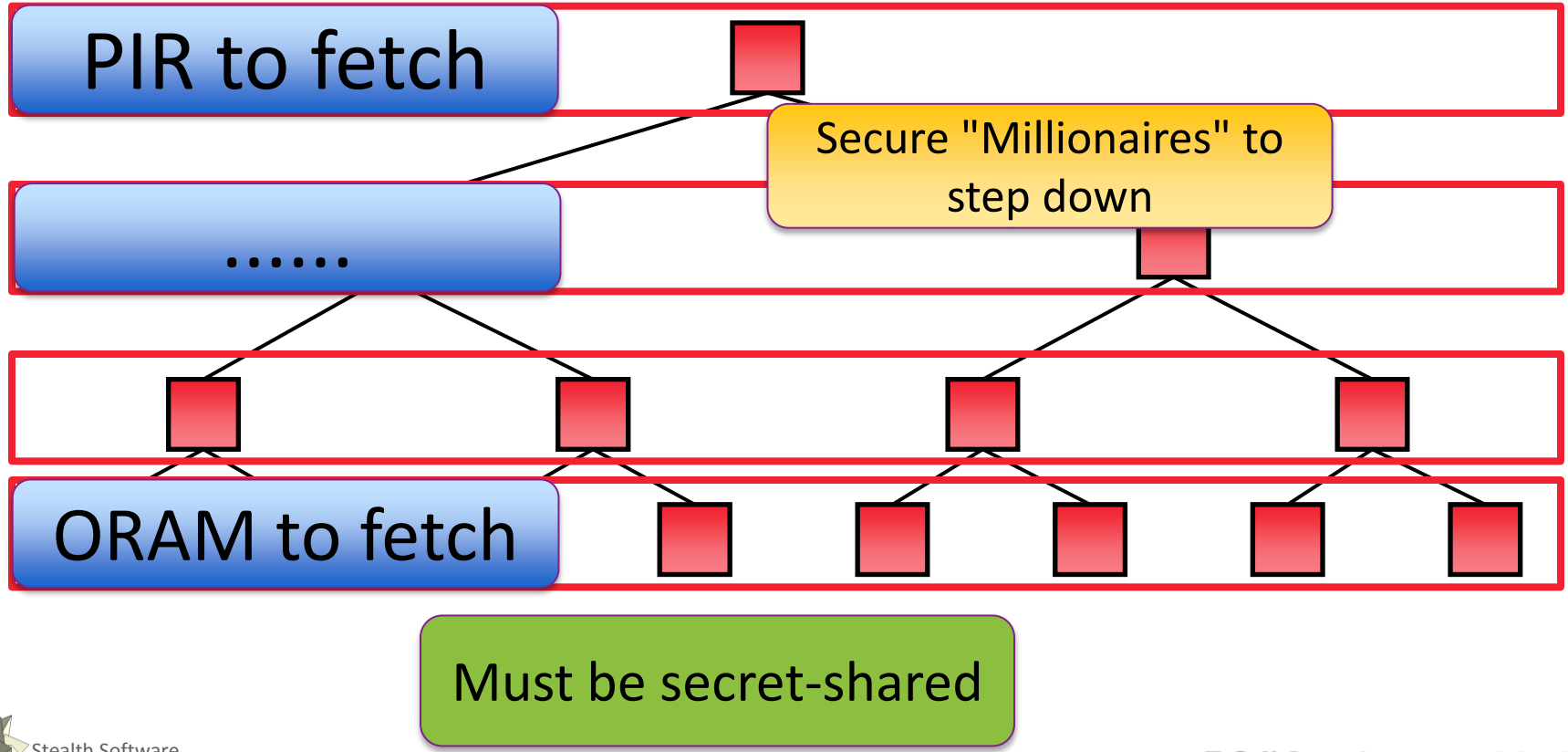
Binary Tree Example



Binary Tree Example



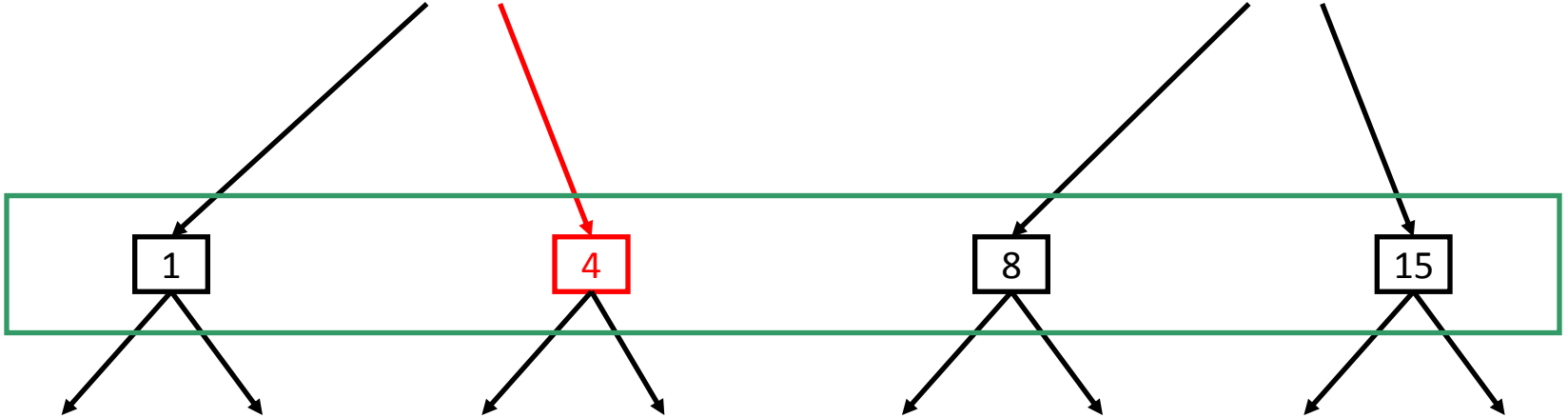
Main Traversal Technique



Reminder: Secret Sharing/One-Time Pad

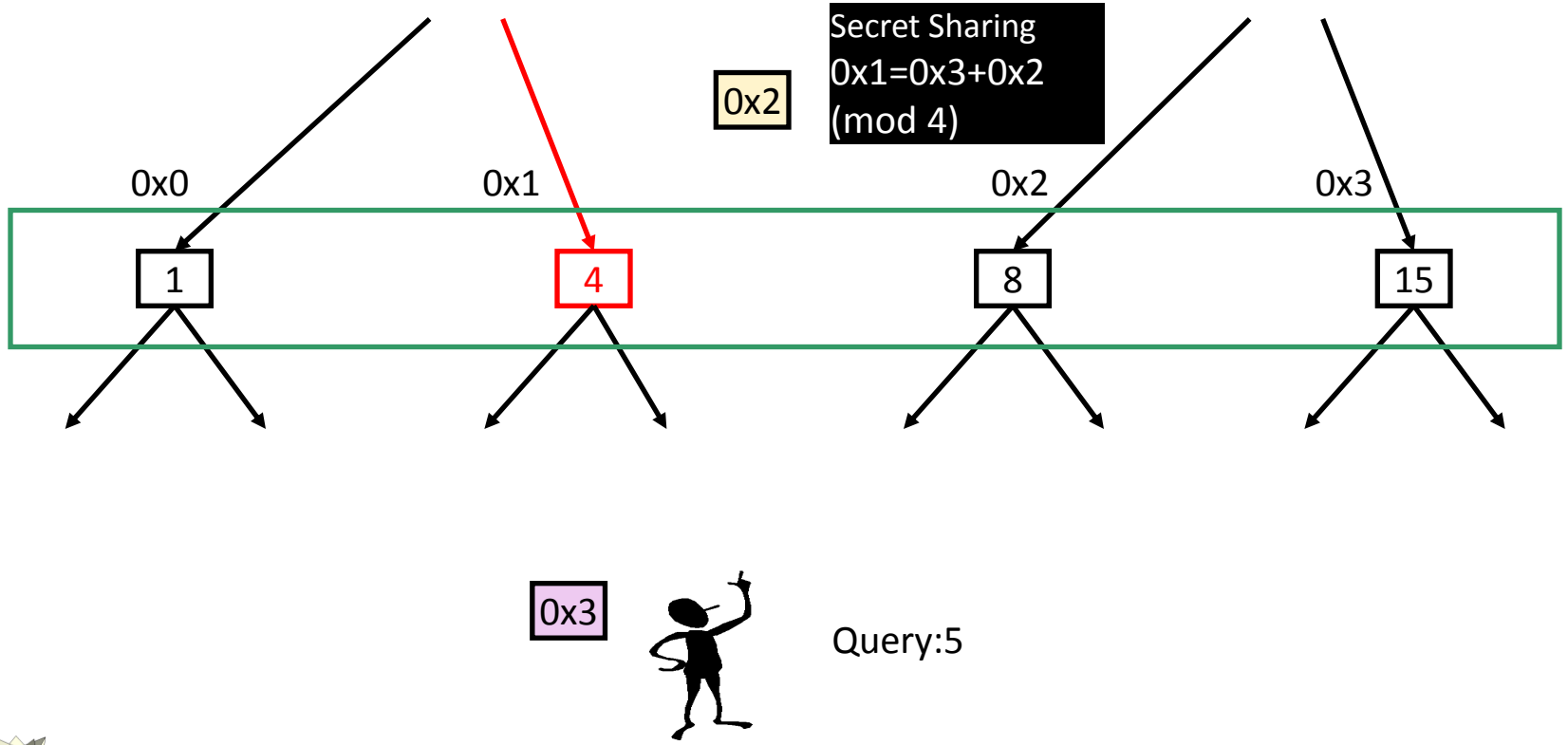
- $M \oplus R = C$
  - One-time pad, R is random key, used only once, C perfectly hides M
- $M = C \oplus R$
  - C and R form a secret sharing of M, each perfectly hiding M

Main Traversal Technique (Details)

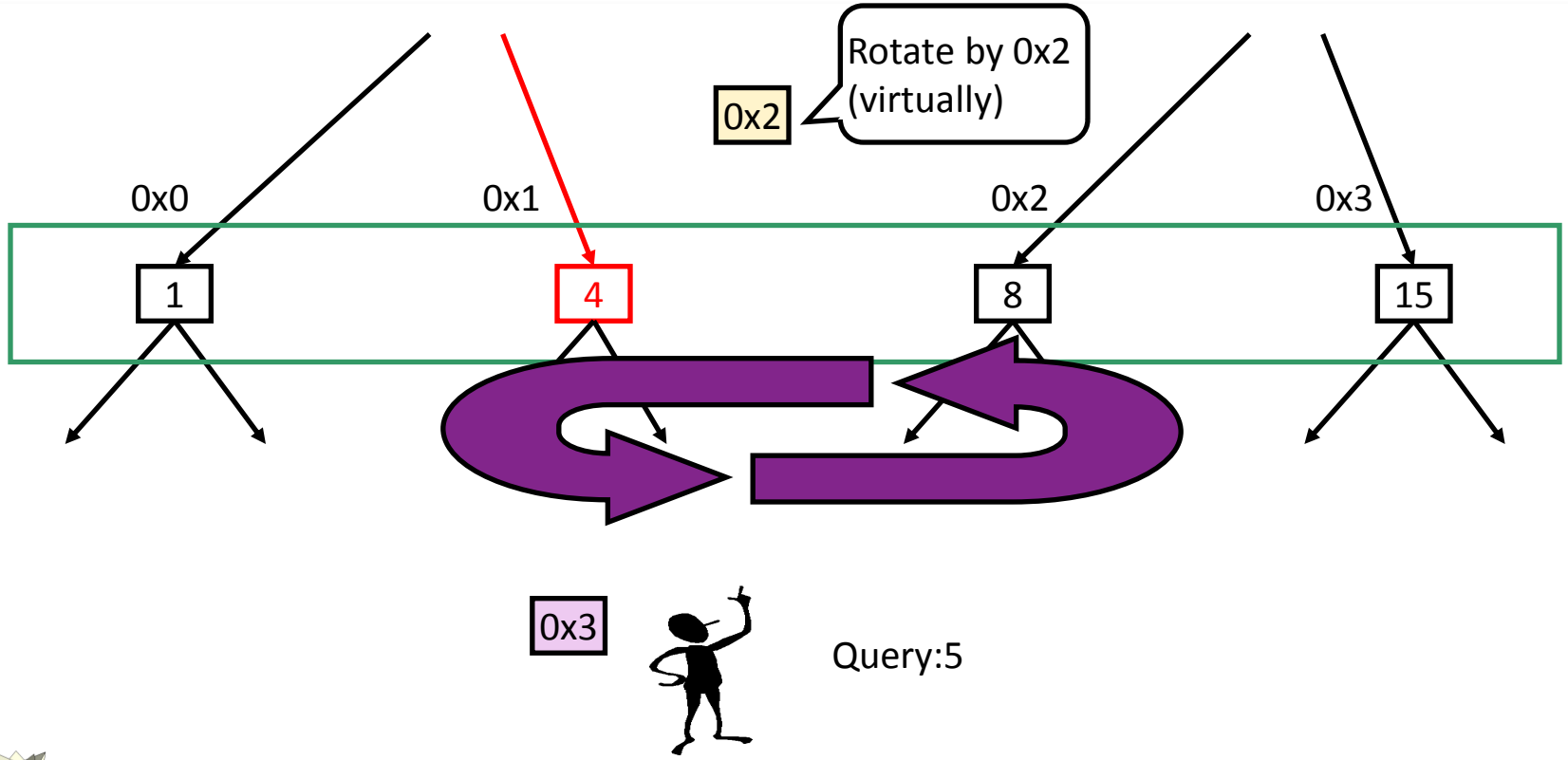


Query:5

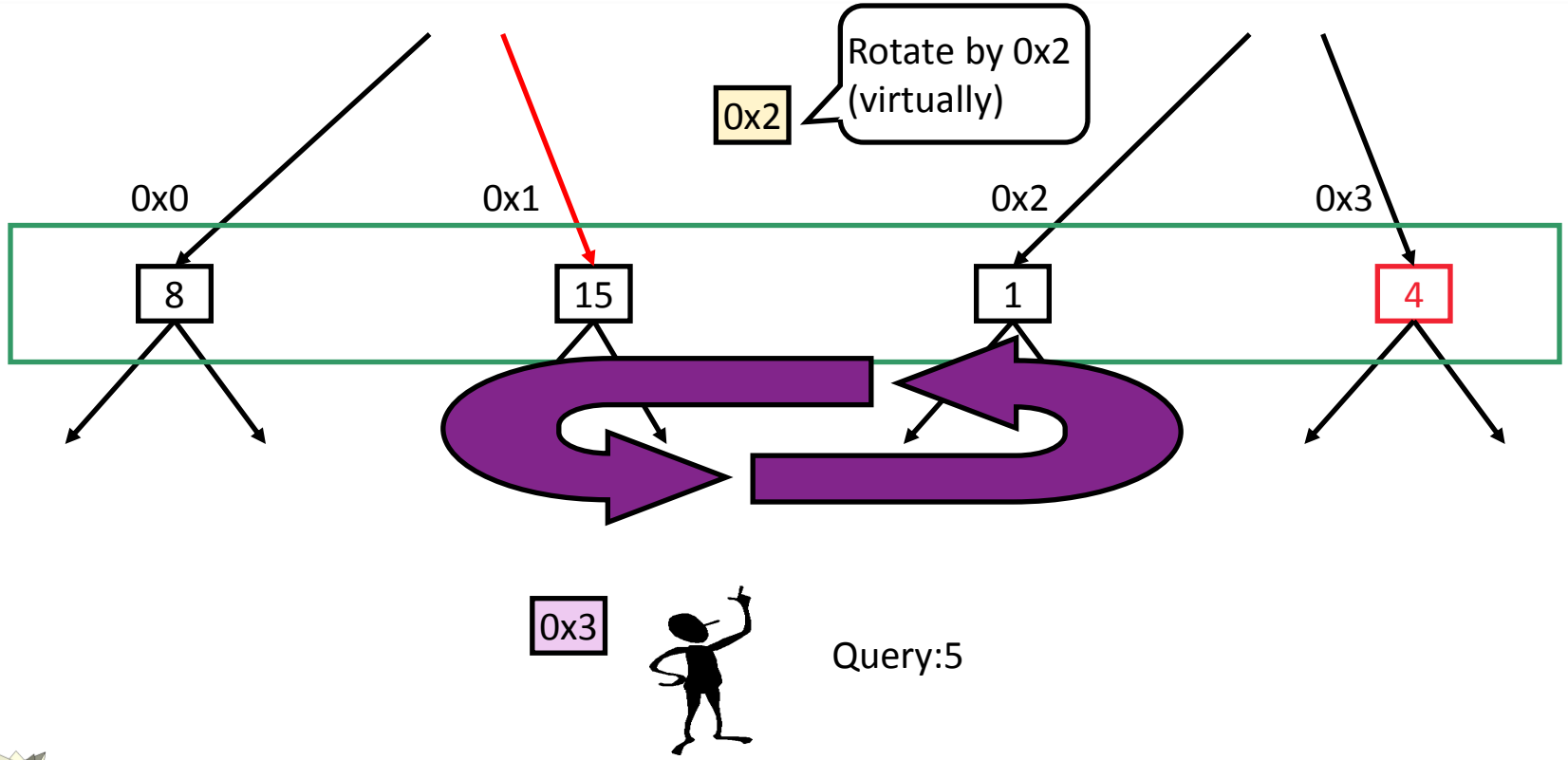
Recursive Assumption



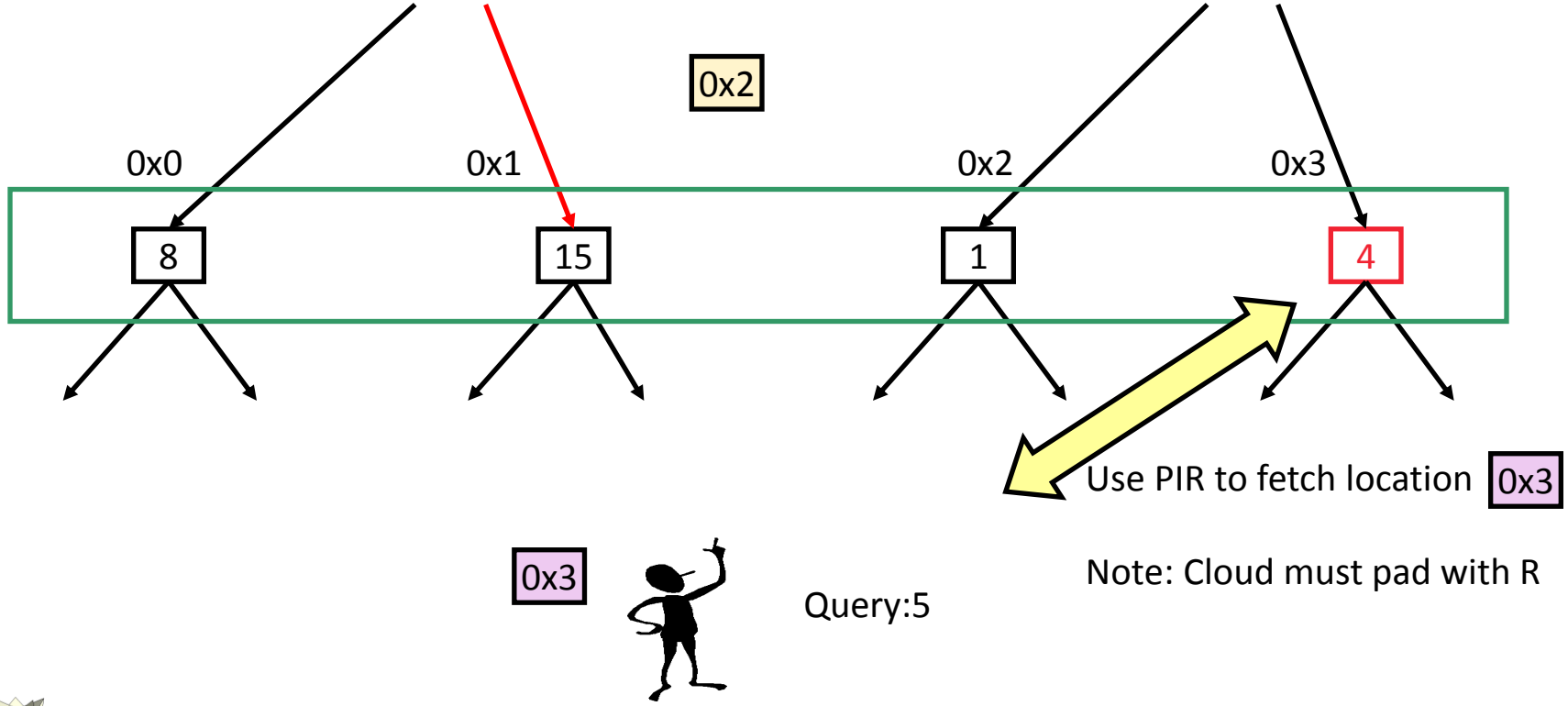
Virtual Rotation



Virtual Rotation



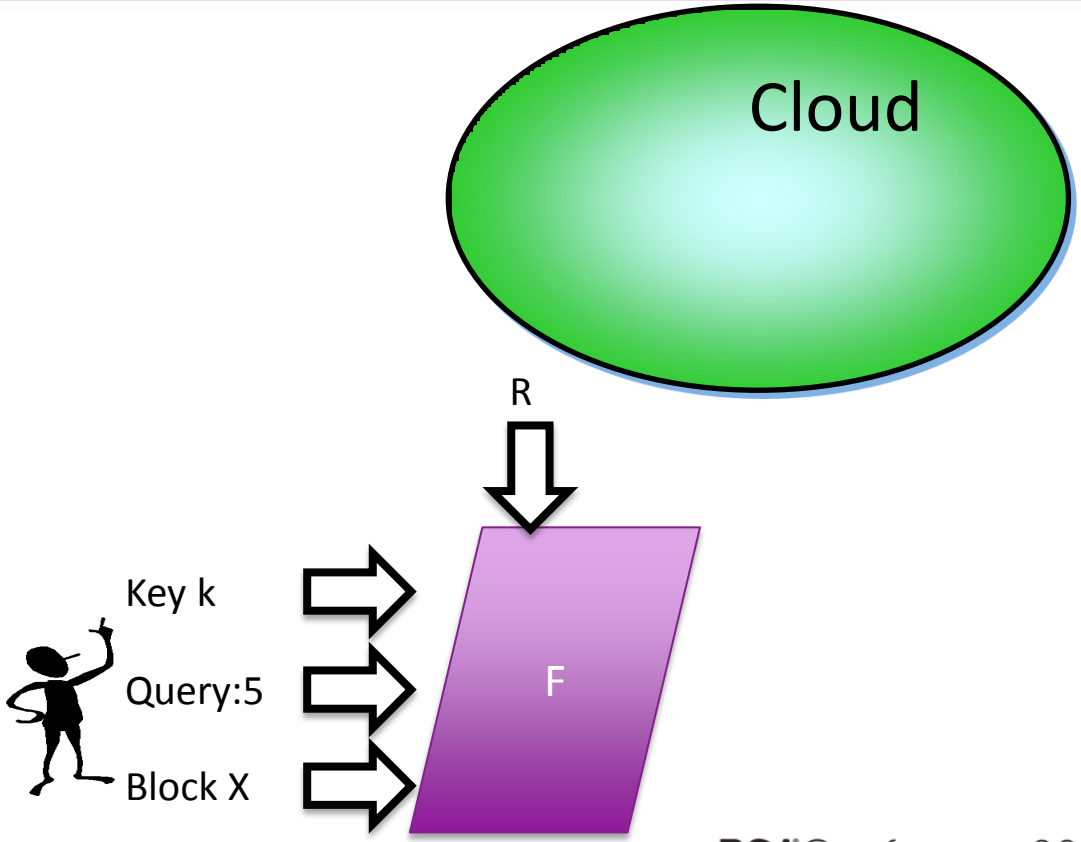
Use PIR to fetch



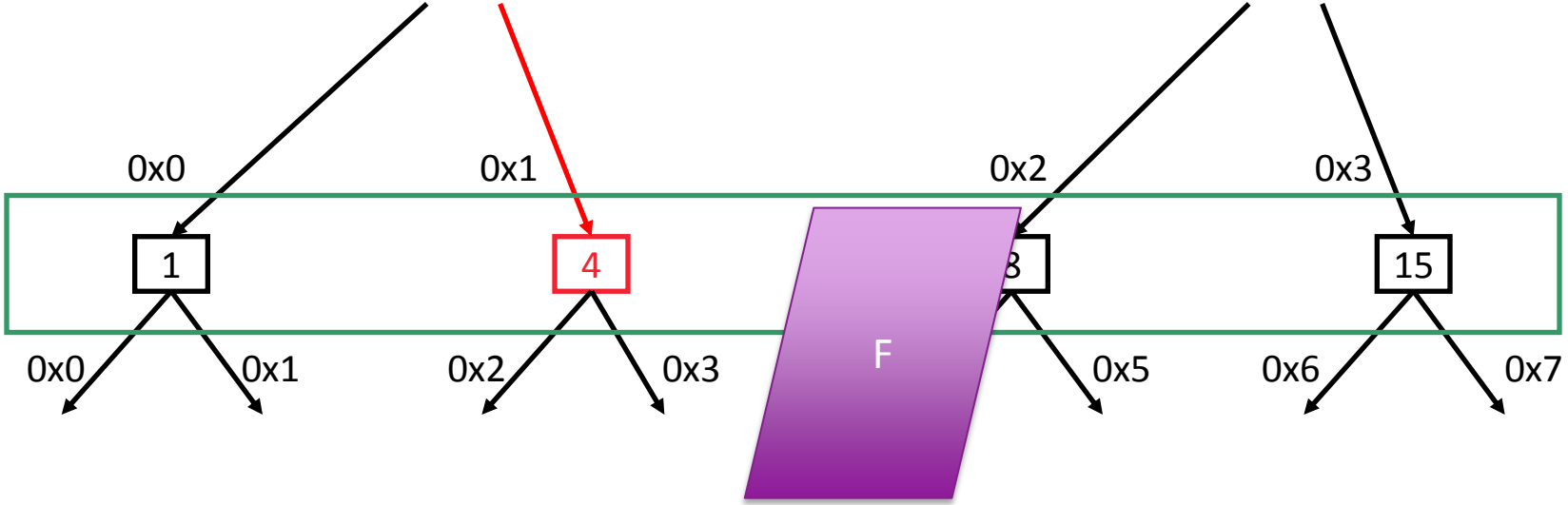
Special Purpose Secure Computation

- PIR returns a block B containing (value=4,Lptr,Rptr)
- Client holds: Secret key k, Query=5, Result of PIR $X=E_k(B) \oplus R$
- Cloud holds: R
- Need to securely compute F(k,Query,X;R):
  - Set $B=D_k(X \oplus R)$ (Custom protocol for this)
  - Return (q>B.value) ? B.Rptr : B.Lptr (Millionaires problem!)
- Final caveat: must return secret shared output

Completing the Recursion

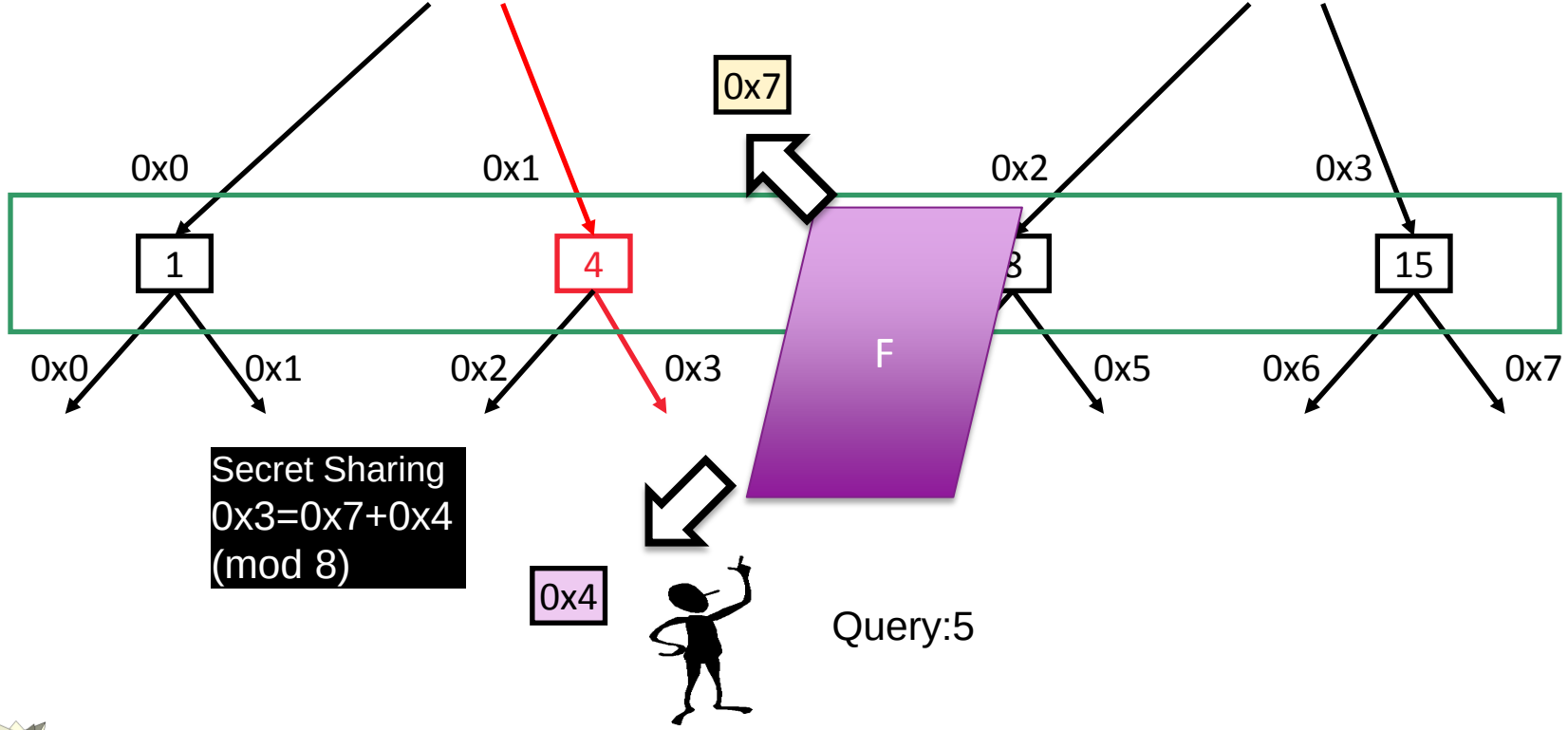


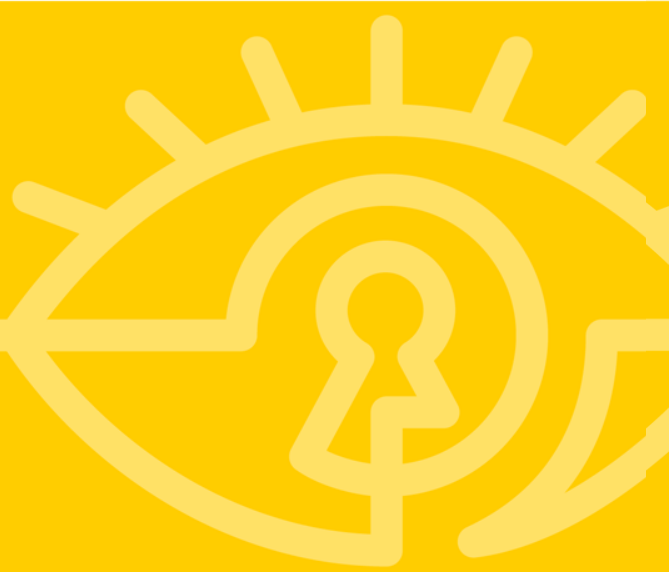
Completing the Recursion



Query:5

Completing the Recursion





Summary and Conclusion

Summary

- We constructed a new SSE scheme that supports a wide variety of queries and can enforce policies and support updates
  - Combination of several techniques including PIR, ORAM, and secure computation
- We implemented the solution, and for large queries we are only 5x slower than MySQL (smaller queries have overhead of up to 100x, but actual time is under 1 second)

Apply What You Have Learned Today

- Within 1 month you should:
  - Identify scenario where this setting occurs
  - Further research our paper and others
- Within 3 months you should:
  - Understand and identify acceptable and unacceptable leakage amongst secure database solutions
- Within 6 months you should:
  - Have a broader understanding of different solutions
  - Discuss applying this technology to suit your needs

THANK YOU!

Full version available on ePrint (2015/1190):
eprint.iacr.org/2015/1190