Dynamic Symmetric Searchable Encryption for Conjunctive Queries

Bibhas Chandra Das

JRF, IAI, TCG CREST

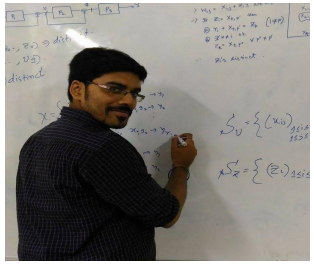
Who Am I ?

Bibhas Chandra Das

Working as JRF in IAI, TCG CREST Kolkata

Supervisor: Nilanjan Dutta, Assistant Professor, IAI Co-Supervisor: Avijit Dutta, Postdoc, IAI

With Support: Avishek Majumder, PhD, ISI

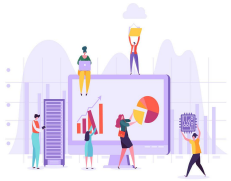


Issues with Big Data:

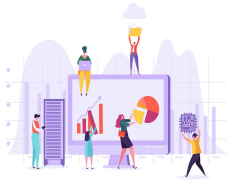
Issues with Big Data:

Issues with Big Data:

Issues with Big Data:



Issues with Big Data:



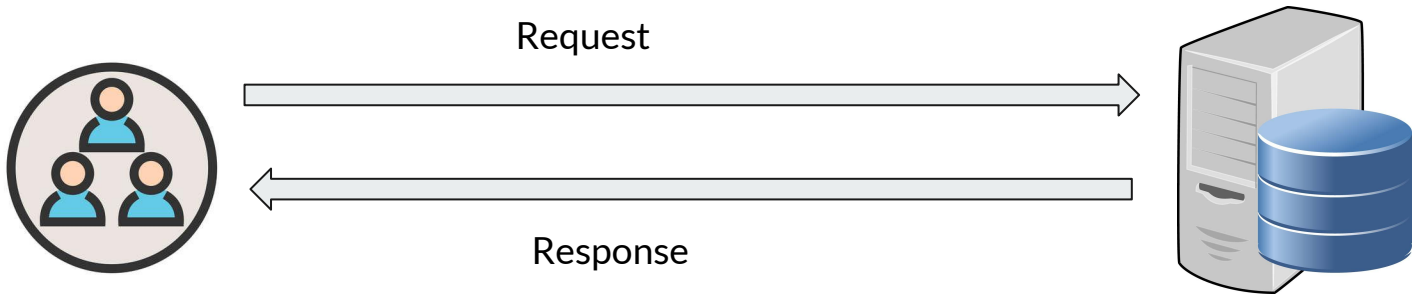
Problem of our Interest:

Where to Store the "BIG" Data

Problem of our Interest:

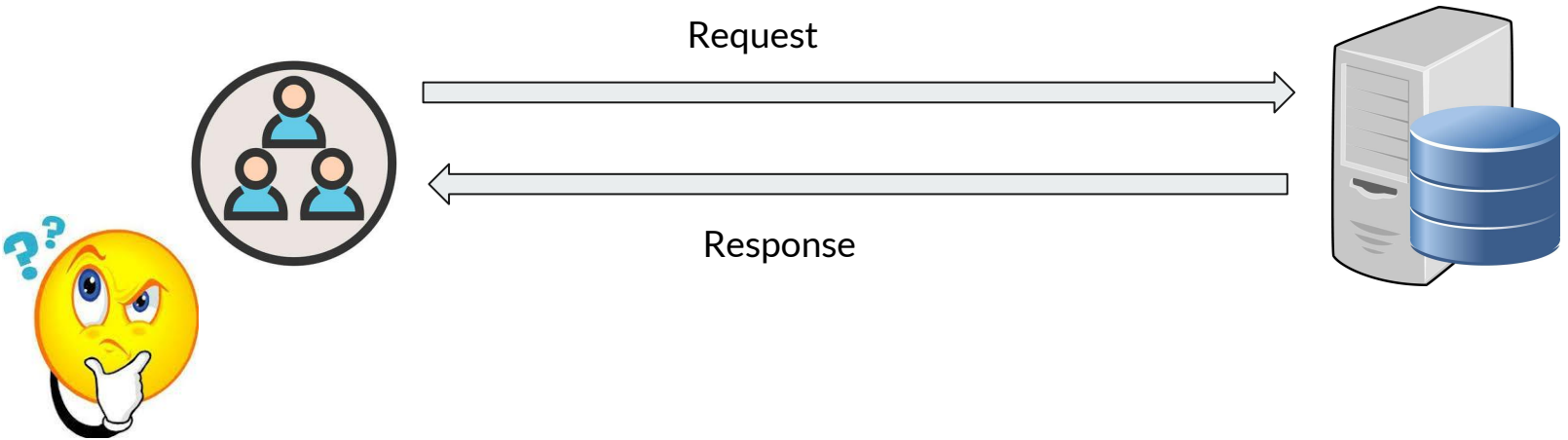
Where to Store the "BIG" Data

Data Outsourcing is a MUST.



Problem of our Interest:

But, there is a BIG BUT...



Problem of our Interest:

But, there is a BIG BUT...

Request

Response


THE WALL STREET JOURNAL.

What's News—

Facebook in Privacy Breach

Top-Ranked Applications Transmit Personal IDs, a Journal Investigation Finds

Tensions Lead to Dueling Protests in China and Japan


Business

BIGGER BITE: APPLE INDIA NE
PROFIT ZOOMS 253% IN FY20

BigBasket data of over 20 mn users breached

Hacker puts database on sale for ₹30L on darkweb

HOLE IN BASKET

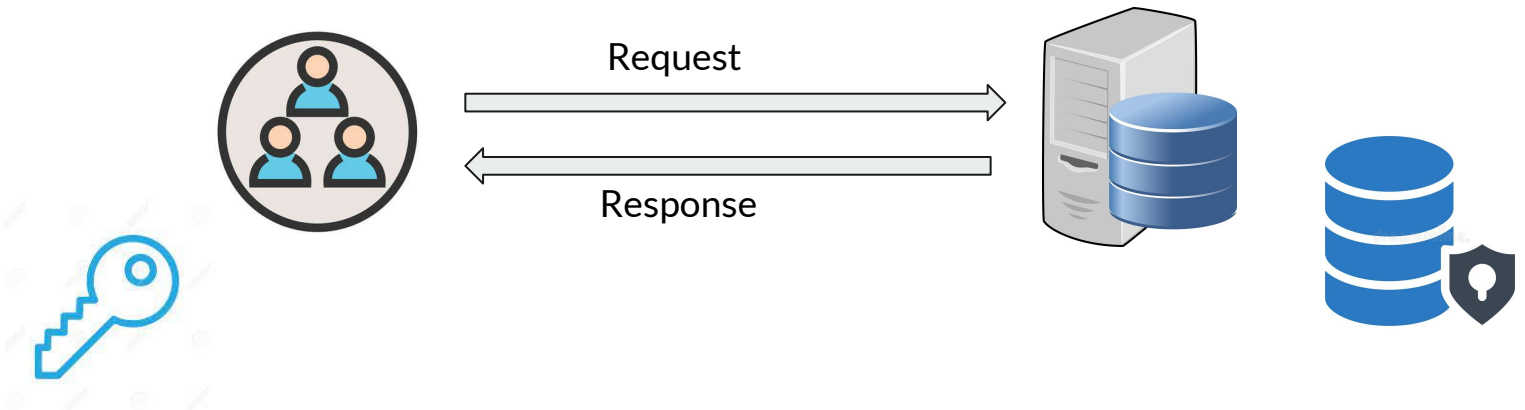
A Quick Solution..

Encrypt the Data and then Store.

A Quick Solution..

Encrypt the Data and then Store.

Consequently Decrypt the Response and then Use.



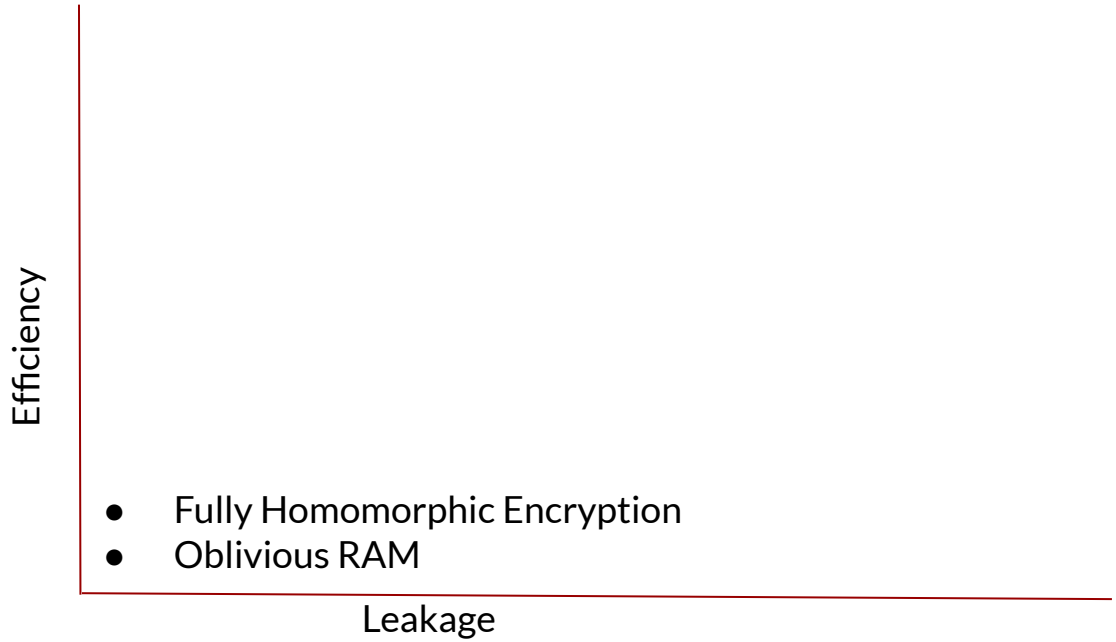
A Quick Solution..

Encrypt the Data and then Store.

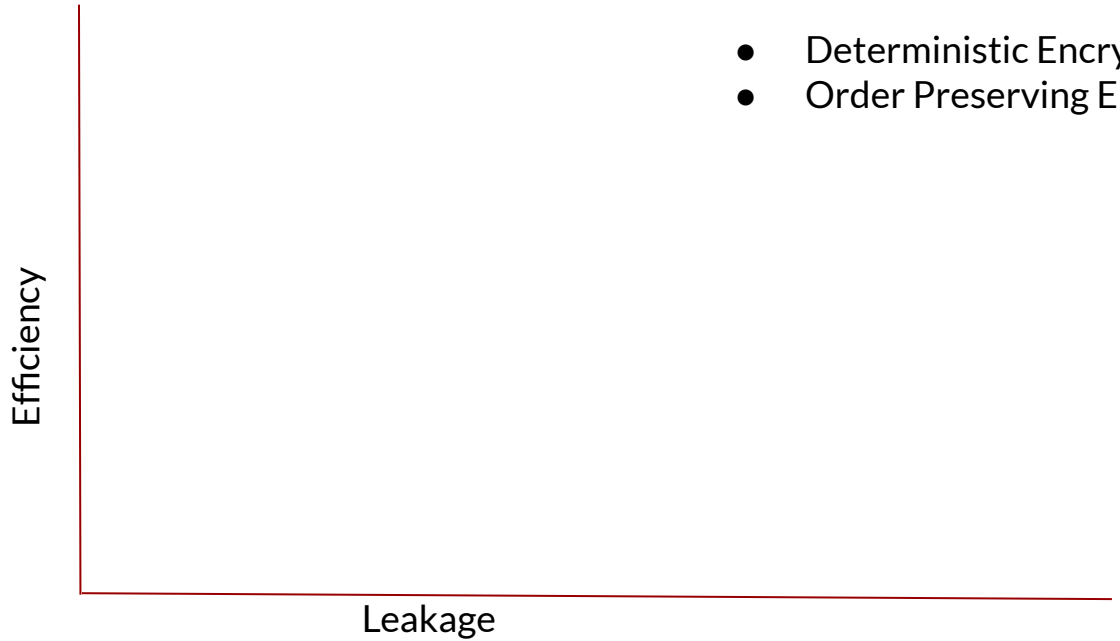
Consequently Decrypt the Response and then Use.

Need HUGE computation for each search

Need a Tradeoff..

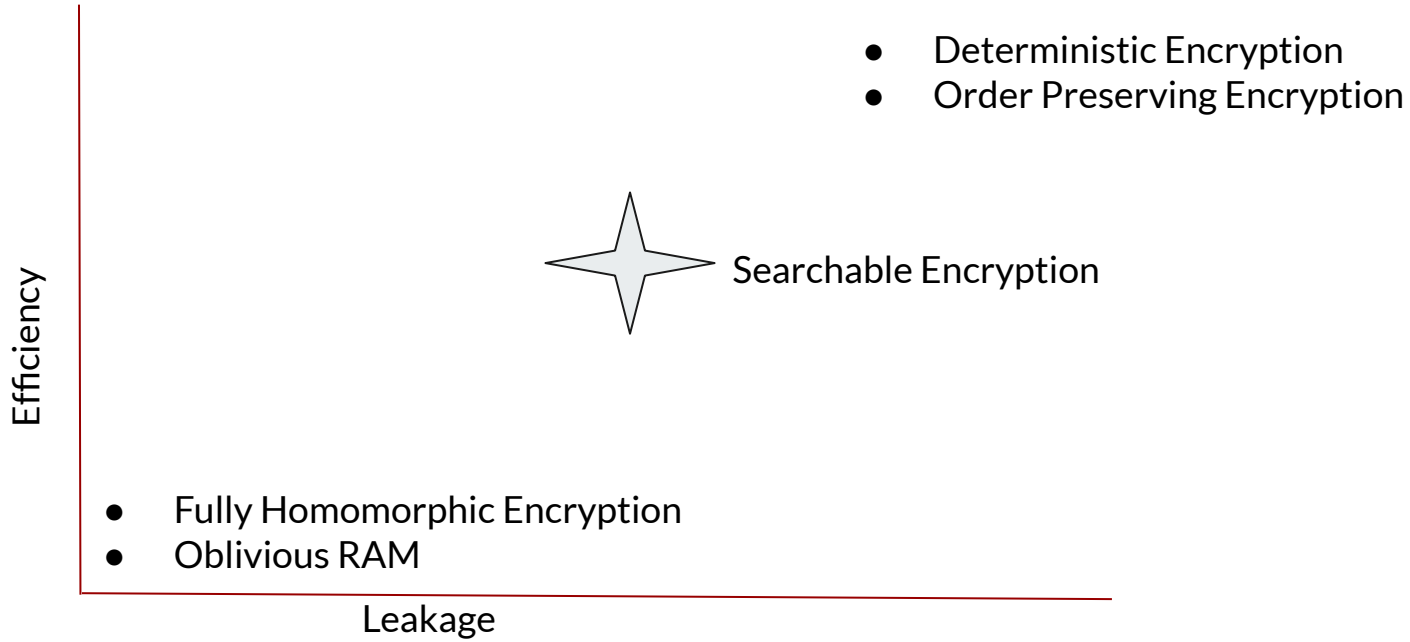


Need a Tradeoff..



- Deterministic Encryption
- Order Preserving Encryption

Need a Tradeoff..



Searchable Encyption

Documents

- Documents tagged by Key words
- Boolean Query Over Keywords
- Range Query over Keywords

Relational Database

A research paper in the area of Cryptogrpahy will have keywords:

- Cryptography
- Symmetric/Public Key
- Cryptanalysis
- Security

Searchable Encyption

Documents

- Documents tagged by Key words
- Boolean Query Over Keywords
- Range Query over Keywords

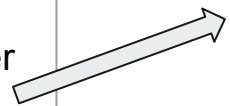
Relational Database

- Find all documents containing “Symmetric Key”
- Find all documents containing “Symmetric Key” and “Cryptanalysis”
- Find all documents containing either “Symmetric Key” or “Public Key” but not “Cryptanalysis”

Searchable Encyption

Documents

- Documents tagged by Key words
- Boolean Query Over Keywords
- Range Query over Keywords



Relational Database

- Find all documents published after “2020”
- Find all documents published before “2004”
- Find all documents published between “2006” and “2012”

Searchable Encyption

Documents

In a company database attributes like “Salary”, “Employee Id” will have their corresponding values “25000”, “4000” and “T022”, “T050”.

Relational Database

- Each record associated with attribute value pair
- Boolean Query over attribute-value pair
- Range Query over attribute value pair

Searchable Encyption

Documents

- Find all employees whose “Salary” is “25000”
- Find all employees whose “Salary” is “25000” and “Age” is “30”
- Find all employees whose “Salary” is “25000” or “Age” is “25” but “Gender” is not “Female”

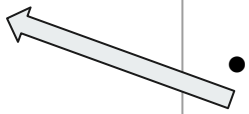
Relational Database

- Each record associated with attribute value pair
- Boolean Query over attribute-value pair
- Range Query over attribute value pair

Searchable Encyption

Documents

- Find all employees whose “Salary” is more than “25000”
- Find all the employees whose “Salary” is less than “50000”
- Find all the employees whose “Age” is between “25” and “45”



Relational Database

- Each record associated with attribute value pair
- Boolean Query over attribute-value pair
- Range Query over attribute value pair

Area of our Focus..

As of now our interest is “General Boolean Queries” for both kind of database.

To achieve this, literature shows the path:

Single Keyword Search

Efficient Schemes already Exists

Area of our Focus..

As of now our interest is "General Boolean Queries" for both kind of database.

To achieve this, literature shows the path:

Single Keyword Search ⟹ **Conjunctive Keyword Search**

Efficient Schemes already Exists

Schemes Exists but Not Efficient

So our First Target is to design "Efficient" scheme for "Conjunctive Keyword Search"

Area of our Focus..

As of now our interest is "General Boolean Queries" for both kind of database.

To achieve this, literature shows the path:

Single Keyword Search ⟹ **Conjunctive Keyword Search**

⇩ ⇩

Efficient Schemes Exists

Schemes Exists but Not Efficient

What does not exists at all:
Any scheme for General Boolean Query and
that is our Final Target

Schemes for Conjunctive Query:

Static Schemes:

- Consists of Two Algorithms: SETUP and SEARCH
- Once the database is STORED in the cloud NO UPDATE is allowed.
- LESS use for its STATIC nature.

Schemes for Conjunctive Query:

Static Schemes:

- Consists of Two Algorithms: SETUP and SEARCH
- Once the database is STORED in the cloud NO UPDATE is allowed.
- LESS use for its STATIC nature.
- Examples:
  1. Highly-Scalable Searchable Symmetric Encryption with Support for Boolean Queries (2013) by [3] David Cash et. al.

Introduced the concept of **TSET** and **XSET** to achieve solution for conjunctive query.

Keeps information about (keyword,file) pair

Interlinks files containing same keywords

Schemes for Conjunctive Query:

Static Schemes:

- Consists of Two Algorithms: SETUP and SEARCH
- Once the database is STORED in the cloud NO UPDATE is allowed.
- LESS use for its STATIC nature.
- Examples:
  1. Highly-Scalable Searchable Symmetric Encryption with Support for Boolean Queries (2013) [3] by David Cash et. al.
  2. Forward secure Conjunctive Keyword Searchable Encryption (2019) [4] by Chengyu Hu et. al.

Used **Bloom Filter** and **Inner Product Encryption** to achieve Conjunctive Search

False Positive results

Takes most of the time of implementation

Schemes for Conjunctive Query:

Dynamic Schemes:

- Consists of Three Algorithms: SETUP, UPDATE and SEARCH
- Can ADD or DELETE files while the database is in use.
- WIDE area of use.

Schemes for Conjunctive Query:

Dynamic Schemes:

- Consists of Three Algorithms: SETUP, UPDATE and SEARCH
- Can ADD or DELETE files while the database is in use.
- WIDE area of use.
- Examples:
    1. Forward and Backward Private Conjunctive Searchable Symmetric Encryption [1] (2020) by Sikhar Patranabis and Debdeep Mukhopadhyay

Based on the idea of **TSET** and **XSET** of Cash et. al.

Schemes for Conjunctive Query:

Dynamic Schemes:

- Consists of Three Algorithms: SETUP, UPDATE and SEARCH
- Can ADD or DELETE files while the database is in use.
- WIDE area of use.
- Examples:
  1. Forward and Backward Private Conjunctive Searchable Symmetric Encryption [1] (2020) by Sikhar Patranabis and Debdeep Mukhopadhyay
  2. Forward and Backward Private Dynamic Searchable Symmetric Encryption for Conjunctive Queries [2] (2021) by Shi-Feng Sun et. al.

Used **Bitmap Index** and **Symmetric Encryption with Homomorphic Addition**

Security Notions:

- Forward Privacy

  Server cannot learn Updated Document matches a Keyword previously Searched for.

- Backward Privacy

  Type I: Leaks the Documents currently Matching w, when they were Inserted, and the total number of Updates on w.

  Type II: Leaks the Documents currently Matching w, when they were Inserted, and When all the Updates on w happened(but Not their Content).

  Type III: Leaks the Documents currently Matching w, when they were Inserted, When all the Updates on w happened, and which Deletion Update Canceled which Insertion Update.

Let's Compare:

Scheme [1]

Computation on Client Side		Communication	Forward Privacy	Backward Privacy
Update	Search			
• 5 PRF evaluation • 1 Group Exponentiation • 1 Group Multiplication • 1 Group Inverse				

Let's Compare:

Scheme by [1]

Computation on Client Side		Communication	Forward Privacy	Backward Privacy
Update	Search			
• 5 PRF evaluation • 1 Group Exponentiation • 1 Group Multiplication • 1 Group Inverse	• |w1|(n+1) PRF evaluations • |w1|(n-1) Group Exponentiations			

Let's Compare:

Scheme by [1]

Computation on Client Side		Communication	Forward Privacy	Backward Privacy
Update	Search			
• 5 PRF evaluation • 1 Group Exponentiation • 1 Group Multiplication • 1 Group Inverse	• |w1|(n+1) PRF evaluations • |w1|(n-1) Group Exponentiations	• Client Sends: |w1| many sets of (n-1) group elements • Server Sends: |matchw1| many tuples containing encrypted identifiers		

Let's Compare:

Scheme by [1]

Computation on Client Side		Communication	Forward Privacy	Backward Privacy
Update	Search			
• 5 PRF evaluation • 1 Group Exponentiation • 1 Group Multiplication • 1 Group Inverse	• |w1|(n+1) PRF evaluations • |w1|(n-1) Group Exponentiations	• Client Sends: |w1| many sets of (n-1) group elements • Server Sends: |matchw1| many tuples containing encrypted identifiers	Achieved for TSET, but Not Achieved for XSET	Provavbly Achieved Type II Privacy

Let's Compare:

Scheme by [2]

Computation on Client Side		Communication	Forward Privacy	Backward Privacy
Update	Search			
• 3 Hash Computations • 1 Symmetric Encryption				

Let's Compare:

Scheme by [2]

Computation on Client Side		Communication	Forward Privacy	Backward Privacy
Update	Search			
• 3 Hash Computations • 1 Symmetric Encryption	• n(|w1| + |w2| + … + |wn|) many Hash Computations • n(|w1| + |w2| + … + |wn|) many Homomorphic Additions • 1 Symmetric Decryption			

Let's Compare:

Scheme by [2]

Computation on Client Side		Communication	Forward Privacy	Backward Privacy
Update	Search			
• 3 Hash Computations • 1 Symmetric Encryption	• n(|w1| + |w2| + … + |wn|) many Hash Computations • n(|w1| + |w2| + … + |wn|) many Homomorphic Additions • 1 Symmetric Decryption	• Client Sends: n many Search Tokens • Server Sends: 1 Encrypted Bit String of length |DB|log |W|		

Let's Compare:

Scheme by [2]

Computation on Client Side		Communication	Forward Privacy	Backward Privacy
Update	Search			
• 3 Hash Computations • 1 Symmetric Encryption	• n(|w1| + |w2| + … + |wn|) many Hash Computations • n(|w1| + |w2| + … + |wn|) many Homomorphic Additions • 1 Symmetric Decryption	• Client Sends: n many Search Tokens • Server Sends: 1 Encrypted Bit String of length |DB|log |W|	Achieved	Provavbly Achieved Type III Privacy

Our Observations:

- Huge Communication in the scheme [2] ***BUT*** Not in the scheme [1]
  -

Our Observations:

- Huge Communication in the scheme [2] ***BUT*** Not in the scheme [1]

- Huge Computation in the scheme [1] ***BUT*** Not in the scheme [2]

Our Observations:

- Huge Communication in the scheme by [2] ***BUT*** Not in the scheme [1]
- Huge Computation in the scheme [1] ***BUT*** Not in the scheme [2]
- Partial Forward Privacy in the scheme [1] ***BUT*** Complete in the scheme [2]

Our Observations:

- Huge Communication in the scheme [2] ***BUT*** Not in the scheme [1]
- Huge Computation in the scheme [1] ***BUT*** Not in the scheme [2]
- Partial Forward Privacy in the scheme [1] ***BUT*** Complete in the scheme [2]
- Type III Backward Privacy in the scheme [2] BUT Type II in the scheme [1]

Our Observations:

- Huge Communication in the scheme [2] ***BUT*** Not in the scheme [1]
- Huge Computation in the scheme [1] ***BUT*** Not in the scheme [2]
- Partial Forward Privacy in the scheme [1] ***BUT*** Complete in the scheme [2]
- Type III Backward Privacy in the scheme [2] ***BUT*** Type II in the scheme [1]
- And Most Shockingly Partial Correctness in the scheme [1]

Scheme by Patranabis et. al. [1]

Update

Client

1. Parse $\mathsf{sk} = (K_T, K_X, K_Y, K_Z)$ and $\mathsf{st} = \mathsf{UpdateCnt}$
2. **If** $\mathsf{UpdateCnt}[w]$ is NULL **then** set $\mathsf{UpdateCnt}[w] = 0$
3. Set $\mathsf{UpdateCnt}[w] = \mathsf{UpdateCnt}[w] + 1$
4. Set $\mathsf{addr} = F(K_T, w||\mathsf{UpdateCnt}[w]||0)$
5. Set $\mathsf{val} = (\mathsf{id}||\mathsf{op}) \oplus F(K_T, w||\mathsf{UpdateCnt}[w]||1)$
6. Set $\alpha = F_p(K_Y, \mathsf{id}||\mathsf{op}) \cdot (F_p(K_Z, w||\mathsf{UpdateCnt}[w]))^{-1}$
7. Set $\mathsf{xtag} = g^{F_p(K_X, w) \cdot F_p(K_Y, \mathsf{id}||\mathsf{op})}$
8. Send $(\mathsf{addr}, \mathsf{val}, \alpha, \mathsf{xtag})$ to the server

Server

1. Parse $\mathbf{EDB} = (\mathsf{TSet}, \mathsf{XSet})$
2. Set $\mathsf{TSet}[\mathsf{addr}] = (\mathsf{val}, \alpha)$
3. Set $\mathsf{XSet}[\mathsf{xtag}] = 1$

Search

Scheme by Patranabis et. al. [1]

Client

1. Parse $\mathsf{sk} = (K_T, K_X)$ and $\mathsf{st} = \mathsf{UpdateCnt}$
2. Use $\mathsf{UpdateCnt}$ to identify keyword with least updates (assumed to be w_1 w.l.o.g)
3. Initialize $\mathsf{stokenList}$ to an empty list
4. Initialize $\mathsf{xtokenList}_1, \ldots, \mathsf{xtokenList}_{\mathsf{UpdateCnt}[w_1]}$ to empty lists
5. **For** $j = 1$ **to** $\mathsf{UpdateCnt}[w_1]$:
   (a) Set $\mathsf{saddr}_j = F(K_T, w_1||j||0)$
   (b) Set $\mathsf{stokenList} = \mathsf{stokenList} \cup \{\mathsf{saddr}_j\}$
   (c) **For** $i = 2$ **to** n:
      i. Set $\mathsf{xtoken}_{i,j} = g^{F_p(K_X, w_i) \cdot F_p(K_Z, w_1||j)}$
      ii. Set $\mathsf{xtokenList}_j = \mathsf{xtokenList}_j \cup \{\mathsf{xtoken}_{i,j}\}$
   (d) **End For**
   (e) Randomly permute the tuple-entries of $\mathsf{xTagList}_j$
6. **End For**
7. Send $(\mathsf{stokenList}, \mathsf{xtokenList}_1, \ldots, \mathsf{xtokenList}_{\mathsf{UpdateCnt}[w_1]})$ to the server

Server

1. Parse **EDB** $= (\mathsf{TSet}, \mathsf{XSet})$
2. Initialize $\mathsf{sEOpList}$ to an empty list
3. **For** $j = 1$ **to** $\mathsf{stokenList}.size$:
   (a) Set $\mathsf{cnt}_j = 1$
   (b) Set $(\mathsf{sval}_j, \alpha_j) = \mathsf{TSet}[\mathsf{stokenList}[j]]$
   (c) **For** $i = 2$ **to** n:
      i. Set $\mathsf{xtoken}_{i,j} = \mathsf{xtokenList}_j[i]$
      ii. Compute $\mathsf{xtag}_{i,j} = (\mathsf{xtoken}_{i,j})^{\alpha_j}$
      iii. **If** $\mathsf{XSet}[\mathsf{xtag}_{i,j}] = 1$, **then** set $\mathsf{cnt}_j = \mathsf{cnt}_j + 1$
   (d) **End For**
   (e) Set $\mathsf{sEOpList} = \mathsf{sEOpList} \cup \{(j, \mathsf{sval}_j, \mathsf{cnt}_j)\}$
4. **End For**
5. Send $\mathsf{sEOpList}$ to the client

Client: Final Output Computation

1. Initialize IdList to an empty list
2. **For** $\ell = 1$ **to** $\mathsf{sEOpList}.size$:
   (a) Let $(j, \mathsf{sval}_j, \mathsf{cnt}_j) = \mathsf{sEOpList}[\ell]$
   (b) Recover $(\mathsf{id}_j||\mathsf{op}_j) = \mathsf{sval}_j \oplus F(K_T, w_1||j||1)$
   (c) **If** op_j is add **and** $\mathsf{cnt}_j = n$ **then** set $\mathsf{sIdList} = \mathsf{sIdList} \cup \{\mathsf{id}_j\}$
   (d) **Else if** op_j is del **and** $\mathsf{cnt}_j > 0$ **then** set $\mathsf{sIdList} = \mathsf{sIdList} \setminus \{\mathsf{id}_j\}$
3. **End For**

Problem in Correctness:

Suppose:

File 1 has keyword w1, w2 and w3

File 2 has keyword w1, w2 and w4

File 3 has keyword w2, w3 and w4

Computation:

$$\text{Set UpdateCnt}[w] = \text{UpdateCnt}[w] + 1$$

$$\text{Set addr} = F(K_T, w||\text{UpdateCnt}[w]||0)$$

$$\text{Set val} = (\text{id}||\text{op}) \oplus F(K_T, w||\text{UpdateCnt}[w]||1)$$

$$\text{Set } \alpha = F_p(K_Y, \text{id}||\text{op}) \cdot (F_p(K_Z, w||\text{UpdateCnt}[w]))^{-1}$$

TSET

Location	Value
$addr_1[w1]$	$Val_1[w1], \alpha_1[w1]$
$addr_1[w3]$	$Val_1[w3], \alpha_1[w3]$
$addr_1[w2]$	$Val_1[w2], \alpha_1[w2]$
$addr_2[w1]$	$Val_2[w1], \alpha_2[w1]$
$addr_3[w3]$	$Val_3[w3], \alpha_3[w3]$
$addr_2[w3]$	$Val_2[w3], \alpha_2[w3]$
$addr_3[w2]$	$Val_3[w2], \alpha_3[w2]$
$addr_2[w2]$	$Val_2[w2], \alpha_2[w2]$
$addr_3[w1]$	$Val_3[w1], \alpha_3[w1]$

Computation:

$$\text{Set xtag} = g^{F_p(K_X, w) \cdot F_p(K_Y, \text{id}||\text{op})}$$

XSET

Location	Value
$xtag_1[w1]$	1
$xtag_1[w3]$	1
$xtag_1[w2]$	1
$xtag_2[w1]$	1
$xtag_3[w3]$	1
$xtag_2[w3]$	1
$xtag_3[w2]$	1
$xtag_2[w2]$	1
$xtag_3[w1]$	1

Problem in Correctness:

Suppose:

File 1 has keyword w1, w2 and w3

File 2 has keyword w1, w2 and w4

File 3 has keyword w2, w3 and w4

Client Search: All Files containing $w1 \wedge w2$

For $j = 1$ **to** $\mathsf{UpdateCnt}[w_1]$:

(a) Set $\mathsf{saddr}_j = F(K_T, w_1||j||0)$

(b) Set $\mathsf{stokenList} = \mathsf{stokenList} \cup \{\mathsf{saddr}_j\}$

(c) **For** $i = 2$ **to** n:

i. Set $\mathsf{xtoken}_{i,j} = g^{F_p(K_X, w_i) \cdot F_p(K_Z, w_1||j)}$

ii. Set $\mathsf{xtokenList}_j = \mathsf{xtokenList}_j \cup \{\mathsf{xtoken}_{i,j}\}$

(d) **End For**

(e) Randomly permute the tuple-entries of $\mathsf{xTagList}_j$

End For

Problem in Correctness:

Suppose:

File 1 has keyword w1, w2 and w3

File 2 has keyword w1, w2 and w4

File 3 has keyword w2, w3 and w4

Client Search: All Files containing w1 ∧ w2

$$\text{Set}\ (\text{sval}_j, \alpha_j) = \text{TSet}[\text{stokenList}[j]]$$

TSET

Location	Value
$addr_1[w1]$	$Val_1[w1], \alpha_1[w1]$
$addr_1[w3]$	$Val_1[w3], \alpha_1[w3]$
$addr_1[w2]$	$Val_1[w2], \alpha_1[w2]$
$addr_2[w1]$	$Val_2[w1], \alpha_2[w1]$
$addr_3[w3]$	$Val_3[w3], \alpha_3[w3]$
$addr_2[w3]$	$Val_2[w3], \alpha_2[w3]$
$addr_3[w2]$	$Val_3[w2], \alpha_3[w2]$
$addr_2[w2]$	$Val_2[w2], \alpha_2[w2]$
$addr_3[w1]$	$Val_3[w1], \alpha_3[w1]$

Problem in Correctness:

Suppose:

File 1 has keyword w1, w2 and w3

File 2 has keyword w1, w2 and w4

File 3 has keyword w2, w3 and w4

Client Search: All Files containing $w1 \wedge w2$

i. Set $\text{xtoken}_{i,j} = \text{xtokenList}_j[i]$

ii. Compute $\text{xtag}_{i,j} = (\text{xtoken}_{i,j})^{\alpha_j}$

iii. If $\text{XSet}[\text{xtag}_{i,j}] = 1$, then set $\text{cnt}_j = \text{cnt}_j + 1$

XSET

Location	Value
$\text{xtag}_1[w1]$	1
$\text{xtag}_1[w3]$	1
$\text{xtag}_1[w2]$	1
$\text{xtag}_2[w1]$	1
$\text{xtag}_3[w3]$	1
$\text{xtag}_2[w3]$	1
$\text{xtag}_3[w2]$	1
$\text{xtag}_2[w2]$	1
$\text{xtag}_3[w1]$	1

Problem in Correctness:

Suppose:

File 1 has keyword w1, w2 and w3

File 2 has keyword w1, w2 and w4

File 3 has keyword w2, w3 and w4

Client Search: All Files containing $w1 \wedge w2$

$$\text{Set}\ (\text{sval}_j, \alpha_j) = \text{TSet}[\text{stokenList}[j]]$$

TSET

Location	Value
$addr_1[w1]$	$Val_1[w1], \alpha_1[w1]$
$addr_1[w3]$	$Val_1[w3], \alpha_1[w3]$
$addr_1[w2]$	$Val_1[w2], \alpha_1[w2]$
$addr_2[w1]$	$Val_2[w1], \alpha_2[w1]$
$addr_3[w3]$	$Val_3[w3], \alpha_3[w3]$
$addr_2[w3]$	$Val_2[w3], \alpha_2[w3]$
$addr_3[w2]$	$Val_3[w2], \alpha_3[w2]$
$addr_2[w2]$	$Val_2[w2], \alpha_2[w2]$
$addr_3[w1]$	$Val_3[w1], \alpha_3[w1]$

Problem in Correctness:

Suppose:

File 1 has keyword w1, w2 and w3

File 2 has keyword w1, w2 and w4

File 3 has keyword w2, w3 and w4

Client Search: All Files containing $w1 \wedge w2$

i. Set $\text{xtoken}_{i,j} = \text{xtokenList}_j[i]$

ii. Compute $\text{xtag}_{i,j} = (\text{xtoken}_{i,j})^{\alpha_j}$

iii. If $\text{XSet}[\text{xtag}_{i,j}] = 1$, then set $\text{cnt}_j = \text{cnt}_j + 1$

XSET

Location	Value
$\text{xtag}_1[w1]$	1
$\text{xtag}_1[w3]$	1
$\text{xtag}_1[w2]$	1
$\text{xtag}_2[w1]$	1
$\text{xtag}_3[w3]$	1
$\text{xtag}_2[w3]$	1
$\text{xtag}_3[w2]$	1
$\text{xtag}_2[w2]$	1
$\text{xtag}_3[w1]$	1

Problem in Correctness:

Suppose:

File 1 has keyword w1, w2 and w3

File 2 has keyword w1, w2 and w4

File 3 has keyword w2, w3 and w4

Client Search: All Files containing $w1 \wedge w2$

Final Output: Identifier for File 1 and File 2

For $\ell = 1$ **to** sEOpList.$size$:

(a) Let $(j, \mathsf{sval}_j, \mathsf{cnt}_j) = \mathsf{sEOpList}[\ell]$

(b) Recover $(\mathsf{id}_j || \mathsf{op}_j) = \mathsf{sval}_j \oplus F(K_T, w_1 || j || 1)$

(c) **If** op_j is add **and** $\mathsf{cnt}_j = n$ **then** set $\mathsf{sIdList} = \mathsf{sIdList} \cup \{\mathsf{id}_j\}$

(d) **Else if** op_j is del **and** $\mathsf{cnt}_j > 0$ **then** set $\mathsf{sIdList} = \mathsf{sIdList} \setminus \{\mathsf{id}_j\}$

End For

Problem in Correctness:

Suppose:

File 1 has keyword w1, w2 and w3

File 2 has keyword w1, ~~w2~~ and w4

File 3 has keyword w2, w3 and w4

Client Search: All Files containing $w1 \wedge w2$

Final Output: Identifier for File 1 and File 2

Now Suppose:

Client Deletes: w2 from File 2 and again Performs Search: $w1 \wedge w2$

For $j = 1$ **to** UpdateCnt$[w_1]$:

(a) Set $\mathsf{saddr}_j = F(K_T, w_1||j||0)$

(b) Set $\mathsf{stokenList} = \mathsf{stokenList} \cup \{\mathsf{saddr}_j\}$

(c) **For** $i = 2$ **to** n:

i. Set $\mathsf{xtoken}_{i,j} = g^{F_p(K_X, w_i) \cdot F_p(K_Z, w_1||j)}$

ii. Set $\mathsf{xtokenList}_j = \mathsf{xtokenList}_j \cup \{\mathsf{xtoken}_{i,j}\}$

(d) **End For**

(e) Randomly permute the tuple-entries of $\mathsf{xTagList}_j$

End For

Problem in Correctness:

Suppose:

File 1 has keyword w1, w2 and w3

File 2 has keyword w1, ~~w2~~ and w4

File 3 has keyword w2, w3 and w4

Client Search: All Files containing $w1 \wedge w2$

Final Output: Identifier for File 1 and File 2

Now Suppose:

Client Deletes: w2 from File 2 and again Performs Search: $w1 \wedge w2$

Final Output: Identifier for File 1 and File 2

For $j = 1$ **to** stokenList.size:

(a) Set $\mathsf{cnt}_j = 1$

(b) Set $(\mathsf{sval}_j, \alpha_j) = \mathsf{TSet}[\mathsf{stokenList}[j]]$

(c) **For** $i = 2$ **to** n:

i. Set $\mathsf{xtoken}_{i,j} = \mathsf{xtokenList}_j[i]$

ii. Compute $\mathsf{xtag}_{i,j} = (\mathsf{xtoken}_{i,j})^{\alpha_j}$

iii. **If** $\mathsf{XSet}[\mathsf{xtag}_{i,j}] = 1$, **then** set $\mathsf{cnt}_j = \mathsf{cnt}_j + 1$

(d) **End For**

(e) Set $\mathsf{sEOpList} = \mathsf{sEOpList} \cup \{(j, \mathsf{sval}_j, \mathsf{cnt}_j)\}$

End For

Problem in Correctness:

Suppose:

File 1 has keyword w1, w2 and w3

File 2 has keyword w1, ~~w2~~ and w4

File 3 has keyword w2, w3 and w4

Client Search: All Files containing $w1 \wedge w2$

Server Replies: Identifier for File 1 and File 2

Now Suppose:

Client Deletes: w2 from File 2 and again Performs Search: $w1 \wedge w2$

Correct Reply: Identifier for File 1 ~~and File 2~~

Ideally Our Target:

- Huge Communication in the scheme [2] ***BUT*** Not in the scheme [1]
- Huge Computation in the scheme [1] ***BUT*** Not in the scheme [2]
- Partial Forward Privacy in the scheme [1] ***BUT*** Complete in the scheme [2]
- Type III Backward Privacy in the scheme [2] ***BUT*** Type II in the scheme [1]
- And Most Shockingly Partial Correctness in the scheme [1]

Our Approach:

First Observation:

- At the time of Update Minimal Computation.

Client

1. Parse $\mathsf{sk} = (K_T, K_X, K_Y, K_Z)$ and $\mathsf{st} = \mathsf{UpdateCnt}$
2. **If** $\mathsf{UpdateCnt}[w]$ is NULL **then** set $\mathsf{UpdateCnt}[w] = 0$
3. Set $\mathsf{UpdateCnt}[w] = \mathsf{UpdateCnt}[w] + 1$
4. Set $\mathsf{addr} = F(K_T, w||\mathsf{UpdateCnt}[w]||0)$
5. Set $\mathsf{val} = (\mathsf{id}||\mathsf{op}) \oplus F(K_T, w||\mathsf{UpdateCnt}[w]||1)$
6. Set $\alpha = F_p(K_Y, \mathsf{id}||\mathsf{op}) \cdot (F_p(K_Z, w||\mathsf{UpdateCnt}[w]))^{-1}$
7. Set $\mathsf{xtag} = g^{F_p(K_X, w) \cdot F_p(K_Y, \mathsf{id}||\mathsf{op})}$
8. Send $(\mathsf{addr}, \mathsf{val}, \alpha, \mathsf{xtag})$ to the server

Server

1. Parse $\mathbf{EDB} = (\mathsf{TSet}, \mathsf{XSet})$
2. Set $\mathsf{TSet}[\mathsf{addr}] = (\mathsf{val}, \alpha)$
3. Set $\mathsf{XSet}[\mathsf{xtag}] = 1$

Our Approach:

First Observation:

- At the time of Update Minimal Computation.
- But at time of Search Huge Computation.

For $j = 1$ **to** $\mathsf{UpdateCnt}[w_1]$:

(a) Set $\mathsf{saddr}_j = F(K_T, w_1||j||0)$

(b) Set $\mathsf{stokenList} = \mathsf{stokenList} \cup \{\mathsf{saddr}_j\}$

(c) **For** $i = 2$ **to** n:

i. Set $\mathsf{xtoken}_{i,j} = g^{F_p(K_X, w_i) \cdot F_p(K_Z, w_1||j)}$

ii. Set $\mathsf{xtokenList}_j = \mathsf{xtokenList}_j \cup \{\mathsf{xtoken}_{i,j}\}$

(d) **End For**

(e) Randomly permute the tuple-entries of $\mathsf{xTagList}_j$

End For

Our Approach:

So, we designed the Update Operation as:

Client

Parse $\mathsf{sk} = (K_t, K_X, K_Y, K_Z)$ and $\mathsf{st} = (\mathsf{CT}, \mathsf{Cnt})$

if $\mathsf{Cnt[w]} = \perp$ **then**

$\mathsf{Cnt[w]} \leftarrow 0$

$\mathsf{ST}_c \xleftarrow{\$} \{0,1\}^\lambda$, and $\mathsf{CT[w]} \leftarrow \mathsf{ST}_c$

end if

$\mathsf{Cnt[w]} \leftarrow \mathsf{Cnt[w]} + +$

$\mathsf{ST}_c \xleftarrow{\$} \{0,1\}^\lambda$

$\mathsf{CT[w]} \leftarrow \mathsf{ST}_{c+1}$

$UT_{c+1} \leftarrow H_1(K_T, \mathsf{ST}_{c+1})$

$C_{\mathsf{ST}_c} \leftarrow H_2(K_T, \mathsf{ST}_{c+1}) \oplus \mathsf{ST}_c$

$D_{\mathsf{ST}_c} \leftarrow H_3(K_T, \mathsf{ST}_{c+1}) \oplus \mathcal{E}(\mathsf{id}, \mathsf{op})$

$\alpha_1 \leftarrow F_p(K_Y, \mathcal{E}(\mathsf{id}, \mathsf{op})) \cdot F_p(K_z, \mathsf{w} \| \mathsf{Cnt[w]})^{-1}$

$\alpha_2 \leftarrow F_p(K_Z, \mathsf{w} \| \mathsf{Cnt[w]})^{-1} \cdot F_p(K_Z, \mathsf{w} \| \mathsf{Cnt[w]} - 1)$

$\mathsf{xtag} \leftarrow g^{F_p(K_X, \mathsf{w}) \cdot F_p(K_Y, \mathcal{E}(\mathsf{id}, \mathsf{op}))}$

Send $(UT_{c+1}, C_{\mathsf{ST}_c}, D_{\mathsf{ST}_c}, \alpha_1, \alpha_2, \mathsf{xtag})$ to server

Our Approach:

And, we immediately get this:

Client

1: Parse $\mathsf{sk} = (K_t, K_X, K_Y, K_Z)$ and $\mathsf{st} = (\mathsf{CT}, \mathsf{Cnt})$

2: **for** $i = 2$ to n **do**

3: $\quad \mathsf{xtoken}_i \leftarrow g^{F_p(K_X, \mathsf{w}_i) \cdot F_p(K_Z, \mathsf{w}_1 \| \mathsf{Cnt}[\mathsf{w}_1])}$

4: **end for**

5: **Send** $(\mathsf{CT}[\mathsf{w}_1], \mathsf{Cnt}[\mathsf{w}_1], K_T, \mathsf{xtoken}_2, \mathsf{xtoken}_3 \cdots, \mathsf{xtoken}_n)$ to server

Our Approach:

Next Observation:

The problem with Correctness still Exists.

- For Each Cross Term, Must Look for both Addition and Deletion operation in every Identifier matching Short Term.

Our Approach:

Next Observation:

The problem with Correctness still Exists.

- For Each Cross Term, Must Look for both Addition and Deletion operation in every Identifier matching Short Term.

$$\alpha_{1T} \leftarrow F_p(K_Y, \mathcal{E}(\mathsf{id},\ \mathsf{op})) \cdot F_p(K_z, \mathsf{w} \| \mathsf{Cnt}[\mathsf{w}])^{-1}$$
$$\alpha_{1F} \leftarrow F_p(K_Y, \mathcal{E}(\mathsf{id}, \mathsf{op}^c)) \cdot F_p(K_z, \mathsf{w} \| \mathsf{Cnt}[\mathsf{w}])^{-1}$$

Our Approach:

Next Observation:

The problem with Correctness still Exists.

- For Each Cross Term, Must Look for both Addition and Deletion operation in every Identifier matching Short Term.
- But then, when Server Recomputes the xtags it will Map Addition Operation with Deletion Operation for the Cross Terms in such Identifiers.

Our Solution:

Update

function Client

$(K_T, K_X, K_Y, K_Z) \leftarrow \mathsf{sk}$ and $\mathsf{Cnt} \leftarrow \mathsf{st}$

if $\mathsf{Cnt}[\mathsf{w}] = \perp$ **then**

$\mathsf{Cnt}[\mathsf{w}] \leftarrow 0, \quad \mathsf{ST}_c \xleftarrow{\$} \{0,1\}^\lambda$

end if

$\mathsf{Cnt}[\mathsf{w}] \leftarrow \mathsf{Cnt}[\mathsf{w}] + +$

$\mathsf{ST}_{c+1} \xleftarrow{\$} \{0,1\}^\lambda$

$UT_{c+1} \leftarrow H_1(K_T, \mathsf{ST}_{c+1})$

$C_{\mathsf{ST}_c} \leftarrow H_2(K_T, \mathsf{ST}_{c+1}) \oplus \mathsf{ST}_c$

$D_{\mathsf{ST}_c} \leftarrow H_3(K_T, \mathsf{ST}_{c+1}) \oplus \mathcal{E}(\mathsf{id}, \mathsf{op})$

$\alpha_{1T} \leftarrow F_p(K_Y, \mathcal{E}(\mathsf{id}, \mathsf{op})) \cdot F_p(K_z, \mathsf{w} \| \mathsf{Cnt}[\mathsf{w}])^{-1}$

$\alpha_{1F} \leftarrow F_p(K_Y, \mathcal{E}(\mathsf{id}, \mathsf{op}^c)) \cdot F_p(K_z, \mathsf{w} \| \mathsf{Cnt}[\mathsf{w}])^{-1}$

$\alpha_2 \leftarrow F_p(K_Z, \mathsf{w} \| \mathsf{Cnt}[\mathsf{w}])^{-1} \cdot F_p(K_Z, \mathsf{w} \| \mathsf{Cnt}[\mathsf{w}] - 1)$

$\mathsf{xtag} \leftarrow g^{F_p(K_X, \mathsf{w} \| \mathsf{op}) \cdot F_p(K_Y, \mathcal{E}(\mathsf{id}, \mathsf{op}))}$

Send $(UT_{c+1}, C_{\mathsf{ST}_c}, D_{\mathsf{ST}_c}, \alpha_{1T}, \alpha_{1F}, \alpha_2, \mathsf{xtag})$ to server

end function

Our Solution:

Search-Client

1: **function Client**
2: $(K_T, K_X, K_Y, K_Z) \leftarrow \mathsf{sk}$ and $\mathsf{Cnt} \leftarrow \mathsf{st}$
3: $UT_{\mathsf{Cnt}[\mathsf{w}_1]} \leftarrow H_1(K_T, \mathsf{ST}_{\mathsf{Cnt}[\mathsf{w}_1]})$
4: **for** $i = 2$ to n **do**
5: $\mathsf{xtoken}_{i+} \leftarrow g^{F_p(K_X, \mathsf{w}_i \| +) \cdot F_p(K_Z, \mathsf{w}_1 \| \mathsf{Cnt}[\mathsf{w}_1])}$
6: $\mathsf{xtoken}_{i-} \leftarrow g^{F_p(K_X, \mathsf{w}_i \| -) \cdot F_p(K_Z, \mathsf{w}_1 \| \mathsf{Cnt}[\mathsf{w}_1])}$
7: **end for**
8: **Send** $(UT_{\mathsf{Cnt}[\mathsf{w}_1]}, \mathsf{Cnt}[\mathsf{w}_1], K_T)$ to server
9: **Send**Permuted version of $\mathsf{xtoken}_{2+}, \mathsf{xtoken}_{3+} \cdots, \mathsf{xtoken}_{n+} \rightarrow$ server
10: **Send**(Permuted version of $\mathsf{xtoken}_{2-}, \mathsf{xtoken}_{3-} \cdots, \mathsf{xtoken}_{n-}) \rightarrow$ server
11: **end function**

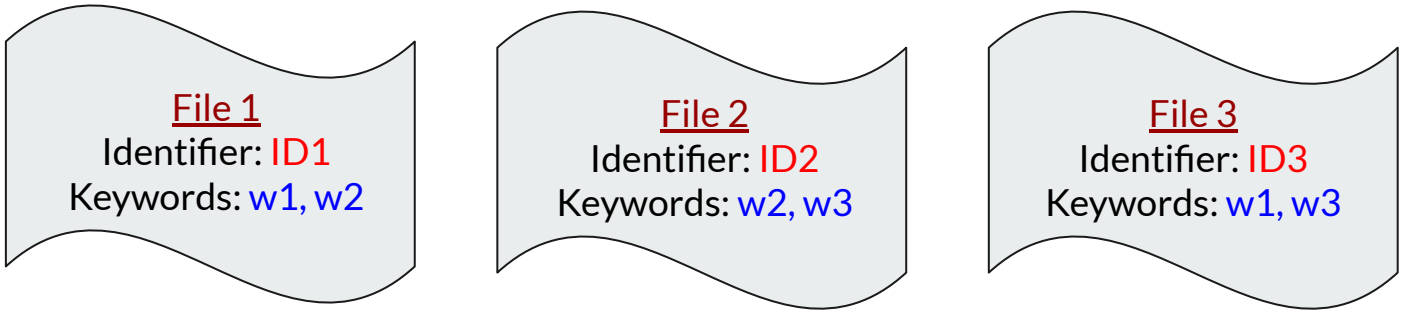
Search-Server

Our Solution:

1: **function** Server
2: sEOpList $\leftarrow \Phi$
3: **for** $j = \mathsf{Cnt}[w_1]$ to 1 **do**
4: **for** $k = 1$ to 4 **do**
5: $\mathsf{Cnt}_k[j] \leftarrow 1$
6: **end for**
7: $UT_j \leftarrow H_1(K_T, \mathsf{ST}_j)$
8: $Parse\mathsf{TSet}[UT_j] = (C_{\mathsf{ST}_{j-1}}, D_{\mathsf{ST}_{j-1}}, \alpha_{1T}, \alpha_{1F}, c$
9: $\mathsf{ST}_{j-1} \leftarrow C_{\mathsf{ST}_{j-1}} \oplus H_2(K_T, \mathsf{ST}_j)$
10: $\mathcal{E}(\mathsf{id}_j, \mathsf{op}_j) \leftarrow D_{\mathsf{ST}_{j-1}} \oplus H_3(K_T, \mathsf{ST}_j)$
11: **for** $i = 2$ to n **do**
12: $\mathsf{xtag}^{ij}_{+\mathsf{T}} = (\mathsf{xtoken}_{i+})^{\alpha_{1T}}$
13: **if** $\mathsf{XSet}[\mathsf{xtag}^{ij}_{+\mathsf{T}}] = 1$ **then**
14: $\mathsf{Cnt}_1[j] \leftarrow \mathsf{Cnt}_1[j] + 1$
15: **end if**
16: $\mathsf{xtag}^{ij}_{+\mathsf{F}} = (\mathsf{xtoken}_{i+})^{\alpha_{1F}}$
17: **if** $\mathsf{XSet}[\mathsf{xtag}^{ij}_{+\mathsf{F}}] = 1$ **then**
18: $\mathsf{Cnt}_2[j] \leftarrow \mathsf{Cnt}_2[j] + 1$
19: **end if**
20: **end for**
21: **for** $i = 2$ to n **do**
22: $\mathsf{xtag}^{ij}_{-\mathsf{T}} = (\mathsf{xtoken}_{i-})^{\alpha_{1T}}$
23: **if** $\mathsf{XSet}[\mathsf{xtag}^{ij}_{-\mathsf{T}}] = 1$ **then**
24: $\mathsf{Cnt}_3[j] \leftarrow \mathsf{Cnt}_3[j] + 1$
25: **end if**
26: $\mathsf{xtag}^{ij}_{-\mathsf{F}} = (\mathsf{xtoken}_{i-})^{\alpha_{1F}}$
27: **if** $\mathsf{XSet}[\mathsf{xtag}^{ij}_{-\mathsf{F}}] = 1$ **then**
28: $\mathsf{Cnt}_4[j] \leftarrow \mathsf{Cnt}_4[j] + 1$
29: **end if**
30: **end for**
31: **for** $i = 2$ to n **do**
32: $\mathsf{xtoken}_{i+} = (\mathsf{xtoken}_{i+})^{\alpha_2}$
33: **end for**
34: **for** $i = 2$ to n **do**
35: $\mathsf{xtoken}_{i-} = (\mathsf{xtoken}_{i-})^{\alpha_2}$
36: **end for**
37: $\mathsf{sEOpList} = \mathsf{sEOpList} \cup \{(j, \mathcal{E}(\mathsf{id}, \mathsf{op}), \mathsf{Cnt}_1[j], \mathsf{Cnt}_2[j], \mathsf{Cnt}_3[j], \mathsf{Cnt}_4[j])\}$
38: **end for**
39: **Send** sEOpList to client

Our Solution:

Let us try to understand Our Proposed Solution through an Example:



Document Representation

Our Solution:

Operation: Add(w1, ID1)

CLIENT

$Cnt[w1] = 1$

Randomly Sample ST_2

$UT_2 = H_1(K_T, ST_2)$

$CT_1 = H_2(K_T, ST_2) \oplus ST_1$

$DT_1 = H_3(K_T, ST_2) \oplus \varepsilon(ID1,add)$

$\alpha_{1T} = F_p(K_Y, \varepsilon(ID1,add)) * F_p(K_Z, w1||Cnt[w1])^{-1}$

$\alpha_{1F} = F_p(K_Y, \varepsilon(ID1,del)) * F_p(K_Z, w1||Cnt[w1])^{-1}$

$\alpha_2 = F_p(K_Z, w1||Cnt[w1]-1) * F_p(K_Z, w1||Cnt[w1])^{-1}$

$xtag = \exp[g, F_p(K_Y, \varepsilon(ID1,add)) * F_p(K_Y, w1||add)]$

Our **Solution:**

Operation: Add(w1, ID1)

TSET

Location	Value
$UT_2[w1]$	$CT_1, DT_1, \alpha_{1T}, \alpha_{1F}, \alpha_2$

XSET

Location	Value
$xtag_2[w1]$	1

Our **Solution:**

Operation: Add(w1, ID3)

TSET

Location	Value
$UT_2[w1]$	$CT_1, DT_1, \alpha_{1T}, \alpha_{1F}, \alpha_2$
$UT_3[w1]$	$CT_2, DT_2, \alpha_{1T}, \alpha_{1F}, \alpha_2$

XSET

Location	Value
$xtag_2[w1]$	1
$xtag_1[w1]$	1

Our **Solution:**

Operation: Add(w2, ID1)

TSET

Location	Value
$UT_2[w2]$	$CT_1, DT_1, \alpha_{1T}, \alpha_{1F}, \alpha_2$
$UT_2[w1]$	$CT_1, DT_1, \alpha_{1T}, \alpha_{1F}, \alpha_2$
$UT_3[w1]$	$CT_2, DT_2, \alpha_{1T}, \alpha_{1F}, \alpha_2$

XSET

Location	Value
$xtag_2[w1]$	1
$xtag_1[w1]$	1
$xtag_1[w2]$	1

Our **Solution:**

Operation: Add(w2, ID2)

TSET

Location	Value
$UT_2[w2]$	$CT_1, DT_1, \alpha_{1T}, \alpha_{1F}, \alpha_2$
$UT_2[w1]$	$CT_1, DT_1, \alpha_{1T}, \alpha_{1F}, \alpha_2$
$UT_3[w2]$	$CT_2, DT_2, \alpha_{1T}, \alpha_{1F}, \alpha_2$
$UT_3[w1]$	$CT_2, DT_2, \alpha_{1T}, \alpha_{1F}, \alpha_2$

XSET

Location	Value
$xtag_2[w1]$	1
$xtag_1[w1]$	1
$xtag_2[w2]$	1
$xtag_1[w2]$	1

Our Solution:

Operation: Add(w3, ID2)

TSET

Location	Value
$UT_2[w2]$	$CT_1, DT_1, \alpha_{1T}, \alpha_{1F}, \alpha_2$
$UT_2[w1]$	$CT_1, DT_1, \alpha_{1T}, \alpha_{1F}, \alpha_2$
$UT_3[w2]$	$CT_2, DT_2, \alpha_{1T}, \alpha_{1F}, \alpha_2$
$UT_3[w1]$	$CT_2, DT_2, \alpha_{1T}, \alpha_{1F}, \alpha_2$
$UT_2[w3]$	$CT_1, DT_1, \alpha_{1T}, \alpha_{1F}, \alpha_2$

XSET

Location	Value
$xtag_2[w1]$	1
$xtag_1[w1]$	1
$xtag_2[w2]$	1
$xtag_1[w3]$	1
$xtag_1[w2]$	1

Our **Solution:**

Operation: Add(w3, ID3)

TSET

Location	Value
$UT_2[w2]$	$CT_1, DT_1, \alpha_{1T}, \alpha_{1F}, \alpha_2$
$UT_3[w3]$	$CT_2, DT_2, \alpha_{1T}, \alpha_{1F}, \alpha_2$
$UT_2[w1]$	$CT_1, DT_1, \alpha_{1T}, \alpha_{1F}, \alpha_2$
$UT_3[w2]$	$CT_2, DT_2, \alpha_{1T}, \alpha_{1F}, \alpha_2$
$UT_3[w1]$	$CT_2, DT_2, \alpha_{1T}, \alpha_{1F}, \alpha_2$
$UT_2[w3]$	$CT_1, DT_1, \alpha_{1T}, \alpha_{1F}, \alpha_2$

XSET

Location	Value
$xtag_2[w1]$	1
$xtag_1[w1]$	1
$xtag_2[w2]$	1
$xtag_1[w3]$	1
$xtag_1[w2]$	1
$xtag_2[w3]$	1

Our Solution:

Operation: Search(w1 ∧ w2 ∧ w3)

$UT_{CNT[w1]} = H_1(K_T, ST_{CNT[w1]})$

$xtoken_{2+} = \exp[\, g, F_p(K_X, w2||add) * F_p(K_Z, w1||CNT[w1])\,]$

$xtoken_{3+} = \exp[\, g, F_p(K_X, w3||add) * F_p(K_Z, w1||CNT[w1])\,]$

$xtoken_{2-} = \exp[\, g, F_p(K_X, w2||del) * F_p(K_Z, w1||CNT[w1])\,]$

$xtoken_{3-} = \exp[\, g, F_p(K_X, w3||del) * F_p(K_Z, w1||CNT[w1])\,]$

CLIENT

Permute

Permute

SERVER

Our Solution:

TSET

Location	Value
$UT_2[w2]$	$CT_1, DT_1, \alpha_{1T}, \alpha_{1F}, \alpha_2$
$UT_3[w3]$	$CT_2, DT_2, \alpha_{1T}, \alpha_{1F}, \alpha_2$
$UT_2[w1]$	$CT_1, DT_1, \alpha_{1T}, \alpha_{1F}, \alpha_2$
$UT_3[w2]$	$CT_2, DT_2, \alpha_{1T}, \alpha_{1F}, \alpha_2$
$UT_3[w1]$	$CT_2, DT_2, \alpha_{1T}, \alpha_{1F}, \alpha_2$
$UT_2[w3]$	$CT_1, DT_1, \alpha_{1T}, \alpha_{1F}, \alpha_2$

XSET

Location	Value
$xtag_2[w1]$	1
$xtag_1[w1]$	1
$xtag_2[w2]$	1
$xtag_1[w3]$	1
$xtag_1[w2]$	1
$xtag_2[w3]$	1

Our Solution:

TSET

Location	Value
$UT_2[w2]$	$CT_1, DT_1, \alpha_{1T}, \alpha_{1F}, \alpha_2$
$UT_3[w3]$	$CT_2, DT_2, \alpha_{1T}, \alpha_{1F}, \alpha_2$
$UT_2[w1]$	$CT_1, DT_1, \alpha_{1T}, \alpha_{1F}, \alpha_2$
$UT_3[w2]$	$CT_2, DT_2, \alpha_{1T}, \alpha_{1F}, \alpha_2$
$UT_3[w1]$	$CT_2, DT_2, \alpha_{1T}, \alpha_{1F}, \alpha_2$
$UT_2[w3]$	$CT_1, DT_1, \alpha_{1T}, \alpha_{1F}, \alpha_2$

$ST_2 = H_2(K_T, ST_3) \oplus CT_2$

$\varepsilon(ID1,add) = H_3(K_T, ST_3) \oplus DT_2$

$UT_2 = H_1(K_T, ST_2)$

Then:

$\varepsilon(ID1,add) = H_3(K_T, ST_2) \oplus DT_1$

SERVER

Our Solution:

$xtag_{+T} = \exp(xtoken_{2+}, \alpha_{1T})$

XSET

Location	Value
$xtag_2[w1]$	1
$xtag_1[w1]$	1
$xtag_2[w2]$	1
$xtag_1[w3]$	1
$xtag_1[w2]$	1
$xtag_2[w3]$	1

Our Solution:

$xtag_{+T} = exp(xtoken_{2+}, \alpha_{1T})$

$Cnt_1[2] = 1$

$xtag_{+T} = exp(xtoken_{3+}, \alpha_{1T})$

$Cnt_1[2] = 1$

XSET

Location	Value
$xtag_2[w1]$	1
$xtag_1[w1]$	1
$xtag_2[w2]$	1
$xtag_1[w3]$	1
$xtag_1[w2]$	1
$xtag_2[w3]$	1

Our Solution:

$xtag_{+F} = \exp(xtoken_{2+}, \alpha_{1F})$

$Cnt_2[2] = 0$

XSET

Location	Value
$xtag_2[w1]$	1
$xtag_1[w1]$	1
$xtag_2[w2]$	1
$xtag_1[w3]$	1
$xtag_1[w2]$	1
$xtag_2[w3]$	1

Our Solution:

$xtag_{+F} = exp(xtoken_{2+}, \alpha_{1F})$

$Cnt_2[2] = 0$

$xtag_{+F} = exp(xtoken_{3+}, \alpha_{1F})$

$Cnt_2[2] = 0$

XSET

Location	Value
$xtag_2[w1]$	1
$xtag_1[w1]$	1
$xtag_2[w2]$	1
$xtag_1[w3]$	1
$xtag_1[w2]$	1
$xtag_2[w3]$	1

Our Solution:

$xtag_{-T} = exp(xtoken_{2-}, \alpha_{1T})$

$Cnt_3[2] = 0$

XSET

Location	Value
$xtag_2[w1]$	1
$xtag_1[w1]$	1
$xtag_2[w2]$	1
$xtag_1[w3]$	1
$xtag_1[w2]$	1
$xtag_2[w3]$	1

Our Solution:

$xtag_{-T} = exp(xtoken_{2-}, \alpha_{1T})$

$Cnt_3[2] = 0$

$xtag_{-T} = exp(xtoken_{3-}, \alpha_{1T})$

$Cnt_3[2] = 0$

XSET

Location	Value
$xtag_2[w1]$	1
$xtag_1[w1]$	1
$xtag_2[w2]$	1
$xtag_1[w3]$	1
$xtag_1[w2]$	1
$xtag_2[w3]$	1

Our Solution:

$xtag_{-F} = exp(xtoken_{2-}, \alpha_{1F})$

$Cnt_4[2] = 0$

XSET

Location	Value
$xtag_2[w1]$	1
$xtag_1[w1]$	1
$xtag_2[w2]$	1
$xtag_1[w3]$	1
$xtag_1[w2]$	1
$xtag_2[w3]$	1

Our Solution:

$xtag_{-F} = exp(xtoken_{2-}, \alpha_{1F})$

$Cnt_4[2] = 0$

$xtag_{-F} = exp(xtoken_{3-}, \alpha_{1F})$

$Cnt_4[2] = 0$

XSET

Location	Value
$xtag_2[w1]$	1
$xtag_1[w1]$	1
$xtag_2[w2]$	1
$xtag_1[w3]$	1
$xtag_1[w2]$	1
$xtag_2[w3]$	1

Our Solution:

Server Computes:

$Cnt_1[2] = 1$ $Cnt_2[2] = 0$ $Cnt_3[2] = 0$ $Cnt_4[2] = 0$

And

$Cnt_1[1] = 1$ $Cnt_2[1] = 0$ $Cnt_3[1] = 0$ $Cnt_4[1] = 0$

Our Solution:

Server Computes:

$Cnt_1[2] = 1$ $Cnt_2[2] = 0$ $Cnt_3[2] = 0$ $Cnt_4[2] = 0$

$Cnt_1[1] = 1$ $Cnt_2[1] = 0$ $Cnt_3[1] = 0$ $Cnt_4[1] = 0$

Server Sends Client:

$$(\varepsilon(ID2,add), Cnt_1[2], Cnt_2[2], Cnt_3[2], Cnt_4[2])$$

And

$$(\varepsilon(ID1,add), Cnt_1[1], Cnt_2[1], Cnt_3[1], Cnt_4[1])$$

Our Solution:

Client Decrypts: $\varepsilon(ID2,add)$

Operation: Add

Chcek: $Cnt_1[2] = 2$ and $Cnt_4[2] = 0$

Our Solution:

Client Decrypts: $\varepsilon(ID2,add)$

Operation: Add

Chcek: $Cnt_1[2] = 2$ and $Cnt_4[2] = 0$

But

$$Cnt_1[2] = 1$$

So,

ID2 is NOT ADDED in Final Result

Our Solution:

Client Decrypts: $\varepsilon(ID1,add)$

Operation: Add

Chcek: $Cnt_1[1] = 2$ and $Cnt_4[1] = 0$

Our Solution:

Client Decrypts: $\varepsilon(ID1,add)$

Operation: Add

Chcek: $Cnt_1[1] = 2$ and $Cnt_4[1] = 0$

But

$$Cnt_1[1] = 1$$

So,

ID1 is NOT ADDED in Final Result

What we have Achieved:

	Computation on Client Side		Communication	Forward Privacy	Backward Privacy
	Update	Search			
Our Proposed Scheme	• 5 PRF evaluation • 1 Group Exponentiation • 3 Group Multiplication • 1 Group Inverse • 3 Hash Computation				

What we have Achieved:

	Computation on Client Side		Communication	Forward Privacy	Backward Privacy
	Update	Search			
Our Proposed Scheme	• 5 PRF evaluation • 1 Group Exponentiation • 3 Group Multiplication • 1 Group Inverse • 3 Hash Computation	• 3(n-1) PRF evaluations • 2(n-1) Group Exponentiations			

What we have Achieved:

Our Proposed Scheme

Computation on Client Side		Communication	Forward Privacy	Backward Privacy
Update	Search			
• 5 PRF evaluation • 1 Group Exponentiation • 3 Group Multiplication • 1 Group Inverse • 3 Hash Computation	• 3(n-1) PRF evaluations • 2(n-1) Group Exponentiations	• Client Sends: 2(n-1) group elements • Server Sends: |matchw1| many tuples containing encrypted identifiers		

What we have Achieved:

	Computation on Client Side		Communication	Forward Privacy	Backward Privacy
	Update	Search			
Our Proposed Scheme	• 5 PRF evaluation • 1 Group Exponentiation • 3 Group Multiplication • 1 Group Inverse • 3 Hash Computation	• 3(n-1) PRF evaluations • 2(n-1) Group Exponentiations	• Client Sends: 2(n-1) group elements • Server Sends: |matchw1| many tuples containing encrypted identifiers	Achieved for TSETand probably for XSET (Working)	Provavbly Achieved Type II Privacy

What about Our Target:

- Very Less Communication
- Average Computation
- Probably Complete Forward Privacy
- Type II Backward Privacy
- Correctness

What about Our Target:

- Very Less Communication
- Average Computation
- Probably Complete Forward Privacy
- Type II Backward Privacy
- Correctness

What about Our Target:

- Very Less Communication
- Average Computation
- Probably Complete Forward Privacy
- Type II Backward Privacy
- Correctness

Future Direction:

- Try to achieve Less Computation
  - Planned to Get Rid Off Large Group Operations

Future Direction:

- Try to achieve Less Computation
  - Planned to Get Rid Off Large Group Operations

- Achieve the Complete Forward Privacy
  - Planned to use Private Membership Test

Future Direction:

- Achieve Less Computation
  - Planned to Get Rid Off Large Group Operations
- Achieve the Complete Forward Privacy
  - Planned to use Private Membership Test
- Extend for General Boolean Query
  - Use the Four Different Cnts according to Query

Quick Questions ?

Snacks are Waiting