Achieve Efficient and Privacy Preserving Range Query over Encrypted Data in Cloud

Deepigha Shree Vittal Babu (Master Student) and Rongxing Lu*

Contact Email: rlu1@unb.ca

Canadian Institute for Cybersecurity (CIC), University of New Brunswick (UNB)

ABSTRACT

Cloud computing is now evolving like never before. While cloud computing is undoubtedly beneficial to our lives, it is not without its downsides, due to the significant security challenges in cloud. It is still challenging to efficiently process encrypted data in semi-trust cloud server. Therefore, there is a high desire to design query mechanisms over encrypted data in cloud to achieve both privacy and efficiency. In this paper, we propose a range query processing scheme that achieves both privacy and efficiency at the same time over encrypted data. The elements are organised on B+ tree which helps in achieving efficiency as searching of any data in a B+ tree is very easy because all data is found in leaf nodes and these leaf node data are ordered in a sequential linked list.

System Model

Data owner:

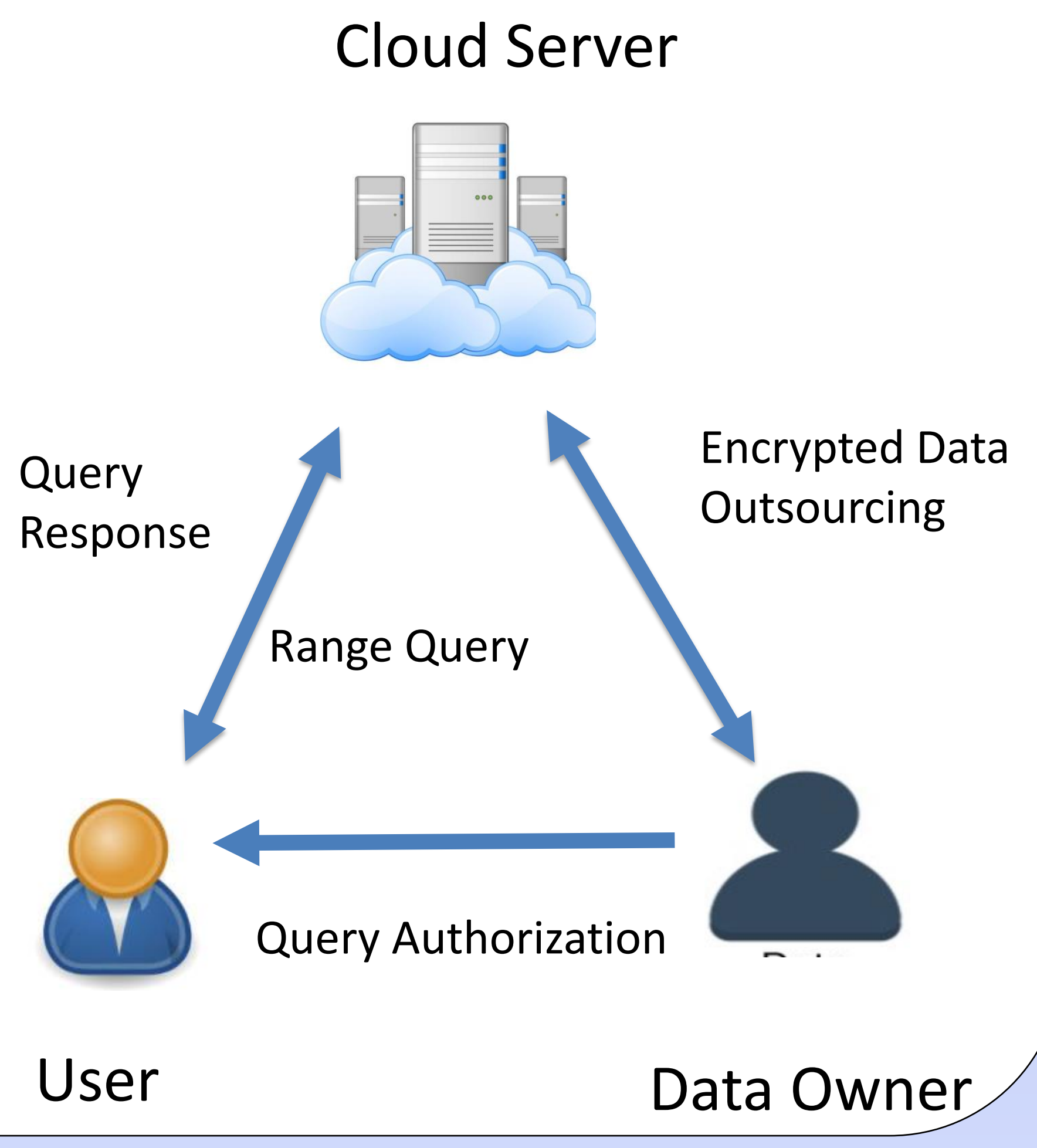
- Data owner is a server provider which owns a set of records.
- Encrypts the records outsourced to the cloud.
- Authorizes the users.

Cloud server:

- Stores the outsourced data from the data owner,
- Processes and responds the range queries from the authorized users.

End user:

- Queries those records of interest to him/her.
- Decrypts the records from the cloud



System Initialization

- The data owner sets the BGN public key $pk = (N, G, G_T, e, g, h)$ and the private key $sk = p$, where $h = g^q$.
- The data owner chooses a secure AES encryption algorithm Enc(.) and a cryptographic hash function H : $\{0, 1\}^* \rightarrow Z_N$, and also chooses three secret keys $k, s, t \in Z_N$.
- Finally, the data owner keeps (p,k,s,t) secretly, and publishes pk = (N, G, G_T, e, g, h), Enc(.), and H : $\{0, 1\}^* \rightarrow Z_N$.
- Keys (p,k,s,t) are kept secretly, and publishes pk = (N, G, G_T , e, g, h), Enc(.), and hash value set H.
- The data owner assigns the processing key $g^{pt} \in G$ to the cloud server and the access keys (k, g^s, g^{s2}) to the end user.

Hash Function

The hash function is defined as follows

- Assume the value range of key x is [0,l], then for any two keys x_i,x_j, if x_i - x_j< 0 , then range of x_i - x_j is [-l, 0)
- Define a private function PF
  PF (x) = H(e(g, g)$^{pst(s+x)}$).
- For each value x ∈ [-l,0), compute
  H(x) = PF(x),
  and form the hash value set
  H = {H1,H2,⋯ ,Hl}.

BGN Homomorphic Encryption

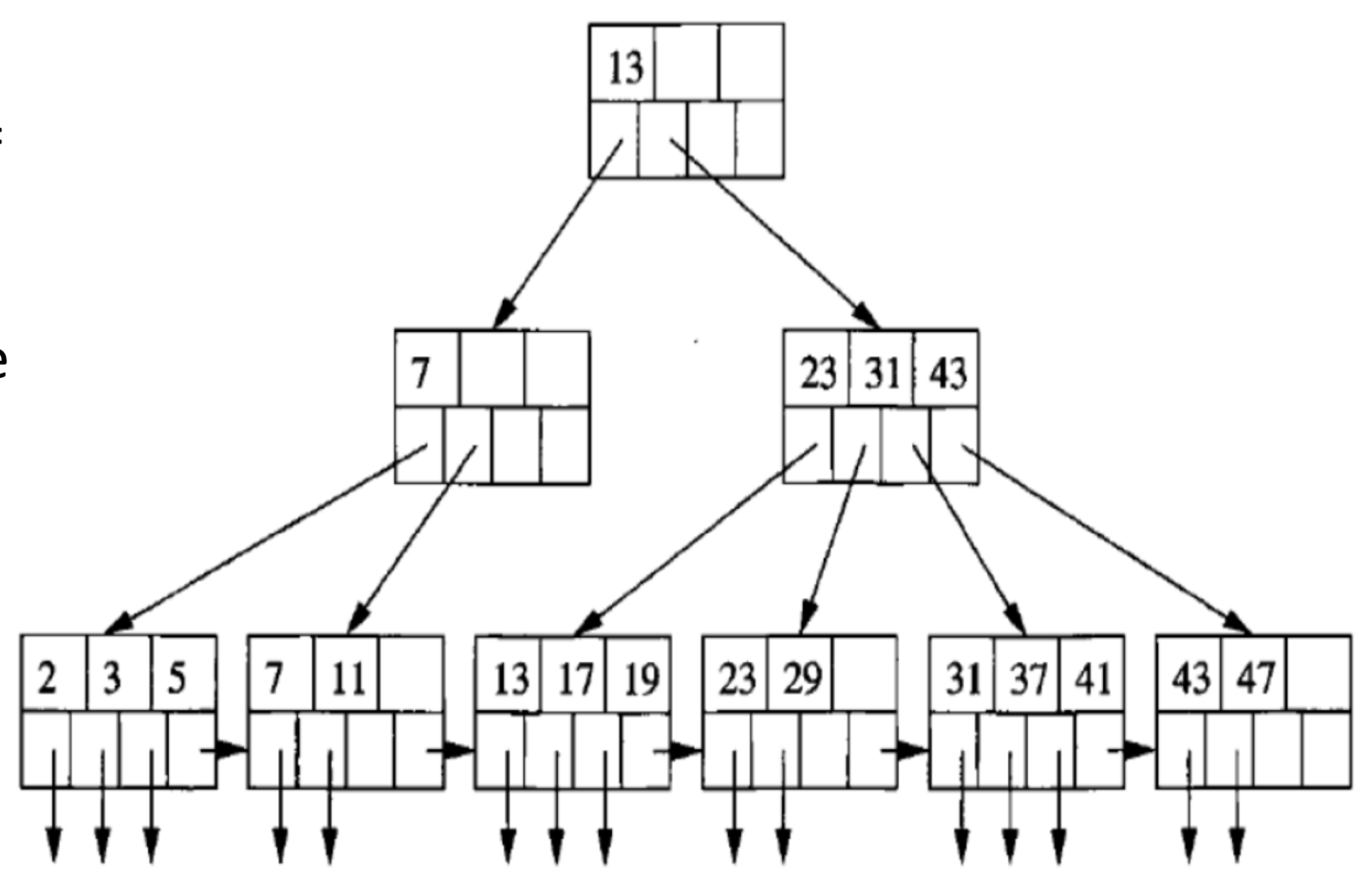
- **Key Generation:** Given the security parameter K, composite bilinear parameters (N,g,G,GT,e) are generated by C Gen(K), where N = pq and p, q are two-bit prime numbers, and g € G is a generator of order n. Set h = g^q, then h is a random generator of the subgroup of G of order p. The public key is **pk = (N, G, GT , e, g, h)**, and the corresponding private key is **sk = p**.
- **Encryption:** Compute the cipher text c = E(m, r) = $g^m h^r \in G$ where m is the message and r is a random number r ∈ Z_N.
- **Decryption**: Given c, the corresponding message can be recovered by the private key p. Observe that $c^p = (g^m h^r)^p = (g^p)^m$. Let g′ = g^p. To find m, it suffices to compute the discrete log of c^p base g′

The BGN encryption enjoys the following homomorphic properties:

- **<u>Addition in G</u>**: E(m_1) · E(m_2) = E(m_1 + m_2).
- **Multiplication in G**: E(m_1)m_2 = E(m_1 · m_2).
- **Multiplication from G to G_T** : e(E(m_1),E(m_2)) = E_T(m_1 ·m_2) ∈ G_T, where E_T (·) denotes a ciphertext in G_T .
- **Addition in G_T** : E_T(m_1)·E_T(m_2)=E_T(m_1 +m_2).
- **Multiplication in G_T** : E_T (m_1)m_2 = E_T (m_1 · m_2).

B+ Tree Implementation

- For each record Ri ∈ R, the data owner first obtains its key x = R.key. Then, based on the keys (x1, x2, · · ·), the data owner constructs a B+ Tree as shown in figure.
- Each record Ri ∈ R, is encrypted using AES encryption and for its key x_i = R_i.key, BGN BGN encryption algorithm is used [x_i] = C_{xi} = $(g^s)^{xi} \cdot h^{ri}$ where $r_i \in Z_N$ is a random number.
- The data owner replaces each record (R_i,x_i) with ([R_i],[x_i]) on the B+ Tree and obtains the corresponding encrypted B+ Tree, as shown in.
- In B+ tree, leaf node data are ordered in a sequential linked list. Therefore, the data owner can get a set of sorted encrypted records ER = {ER1, ER2, · · · } by traversing through the leaf nodes.
- The data owner outsources the encrypted B+ tree and the sorted encrypted records ER = {ER1,ER2,⋯} to the cloud server.
- Searching of any data in a B+ tree is very easy because all data is found in leaf nodes.
- Upon receiving the range query, the cloud server determines the upper bound and lower bound in the encrypted tree.
- Once the upper bound and lower bound are localized the encrypted records between them can be easily retrieved.



Range Query

In this paper, we generalize the above two type of range queries as one

select * from table where [age]+[y] between [a] and [b]

where (a,b,y) are integer parameters provided in the range query

Privacy-Preserving Comparison Algorithm

```
Input: processing key $g^{pt}$, encrypted key $[x_i] = C_{x_i}$ and case-1:
        $[a] = C_a, [y] = C_y$ or case-2: $[b] = C_b, [y] = C_y$
Output: 1 (result is >=) or 0 (result is <)
compute $C_x = C_{x_i} \cdot C_y = g^{s(x_i+y)} \cdot h^{r_y+r_i}$
if case-1: [a], [y] then
    compute $X_i = e(\frac{C_a}{C_x}, g^{pt}) = e(g,g)^{pst(s+a-(x_i+y))}$
    compute the hash value $H(X_i)$
    if $H(X_i)$ in the set $\mathbb{H}$ then
        return 0 // showing $a < (x_i + y)$
    else
        return 1 // showing $a \geq (x_i + y)$
else if case-2: [b], [y] then
    compute $X_i = e(\frac{C_b}{C_x}, g^{pt}) = e(g,g)^{pst(s+b-(x_i+y))}$
    compute the hash value $H(X_i)$
    if $H(X_i)$ in the set $\mathbb{H}$ then
        return 0 // showing $b < (x_i + y)$
    else
        return 1 // showing $b \geq (x_i + y)$
```

The privacy preserving comparison algorithm is used while localizing the the upper bound and lower bound while query processing.

- Starting from the root node, the cloud server runs this algorithm. If the returned value is 0, continue to use the algorithm to compare the upper bound/lower bound with the left child in the encrypted B+ Tree.
- Otherwise, continue to use the algorithm to compare with the right child in the encrypted B+ Tree, until the location that it should be inserted is found.
- Once the location is found, the upper bound/lower bound can be localized.

Conclusion

- In this paper, we have proposed an efficient range query processing scheme that guarantees strong privacy over encrypted data in cloud.
- Since the proposed scheme is integrated with the encrypted B+ Tree and privacy-preserving comparison techniques, which is not only privacy-preserving against the cloud server but also high efficient in the range query execution.
- This scheme can efficiently support real-time range queries.