A Secure Range Query Processing Algorithm for the Encrypted Database on the Cloud

Hyeong-Il Kim, Munchul Choi, Hyeong-Jin Kim and Jae-Woo Chang

Abstract Secure range query processing algorithms have been studied as the range query can be used as a baseline technique in various fields. However, when processing a range query, the existing methods fail to hide the data access patterns which can be used to derive the actual data items and the private information of a querying issuer. The problem is that the data access patterns can be exposed even though the data and query are encrypted. So, in this paper we propose a new range query processing algorithm on the encrypted database. Our method conceals the data access patterns while supporting efficient query processing by using our proposed encrypted index search scheme. Through the performance analysis, we show that the proposed range query processing algorithm can efficiently process a query while hiding the data access patterns.

Keywords Database outsourcing · Database encryption · Encrypted index structure · Secure range query processing · Data access patterns

1 Introduction

With the development of cloud computing, a data owner can outsource his/her database and their managements to a cloud. By outsourcing the database, the data owner can flexibly utilize the resource of the cloud, thus reducing the management

H.-I. Kim · M. Choi · H.-J. Kim · J.-W. Chang (✉)
Department of Computer Engineering, Jeonbuk National University, Jeonju, South Korea
e-mail: jwchang@jbnu.ac.kr

H.-I. Kim
e-mail: melipion@jbnu.ac.kr

M. Choi
e-mail: oopsmun@jbnu.ac.kr

H.-J. Kim
e-mail: yeon_hui4@jbnu.ac.kr

J.J. (Jong Hyuk) Park et al. (eds.), *Advanced Multimedia and Ubiquitous Engineering*, Lecture Notes in Electrical Engineering 393,
DOI 10.1007/978-981-10-1536-6_14

costs. The cloud not only stores the database, but also provides an authorized user with querying services on the outsourced database. However, if the original database is outsourced to the cloud, the cloud or an attacker can abuse the private information stored in the database. For example, if a real estate agent outsources his/her original database to the cloud, the cloud or an attacker can sell the property information to the other agents. In addition, the private information of a user can be revealed to the attacker. For example, if a user sends a query with his/her location information to use location-based services, the attacker can find places where the user frequently visit.

Meanwhile, a range query, one of the most typical query types, is widely used as a baseline technique in many fields. The range query finds all the data inside a given query range. However, some privacy threat can occur when issuing the range query. This is because a range information is closely related to the interest of a user.

Therefore, researches on the range query processing algorithms which consider the data privacy have been performed [1–5]. However, all the existing works fail to hide the data access patterns during the query processing. The data access patterns are the good source to derive not only the actual data items, but also the private information of a querying issuer. This is the critical problem because the data access patterns can be exposed even though the data and query are encrypted [6]. To the best of knowledge, a scheme proposed in [7] is the only work that hides the data access patterns over the encrypted database. However, the scheme only supports kNN query processing and requires high computation cost. To solve the problem, in this paper we propose a new range query processing algorithm on the encrypted database. Our method conceals the data access patterns while supporting efficient query processing by using our proposed encrypted index search scheme.

The rest of the paper is organized as follows. Section 2 introduces the related work and Sect. 3 presents the overall system architecture and secure protocols. In Sect. 4, we propose our secure range query processing algorithm based on the encrypted index. The performance analysis of our scheme is presented in Sect. 5. Finally, we conclude the paper with some future research directions in Sect. 6.

2 Related Work

Yiu et al. [1] proposed the cryptographic transformation (CRT) method which utilizes encrypted R-tree index. However, CRT cannot preserve the data access pattern as a user hierarchically requests the required R-tree nodes to the cloud. A scheme proposed by Hore et al. [2] partitions the data into a set of buckets and builds indices for buckets. However, a data owner should store and search the indices locally. In addition, the result of schemes in [1, 2] usually contains the false-positives. Wang et al. [3] proposed a scheme which utilizes the encrypted version of R-tree. However, the scheme has a shortcoming that a result contains many false-positives. In addition, the data access patterns are revealed because the cloud returns a set of nodes which intersect the query range. Wang et al. [4]

proposed an encrypted R-tree based range query processing scheme. However, the data access patterns are revealed to a cloud because all the identifiers of the data that satisfy the query are returned by the cloud. Most recently, Kim et al. [5] proposed a range query processing scheme using the Hilbert-curve order based index. However, the scheme has a problem that a user is in charge of index traversal during the query processing. In addition, the scheme leaks the data access pattern and the query result may contain false-positives.

Next, we briefly review the Paillier cryptosystem [8]. The Paillier cryptosystem is an additive homomorphic and probabilistic asymmetric encryption scheme for public key cryptography. The public key pk for encryption is given by (N, g), where N is a product of two large prime numbers p and q, and g is in $Z^*_{N^2}$. The secret key sk for decryption is given by (p, q). Let $E()$ denote the encryption function and $D()$ denote the decryption function. The Paillier crypto system has the following properties. (i) The product of two ciphertexts $E(m_1)$ and $E(m_2)$ results in the encryption of the sum of their plaintexts m_1 and m_2; $E(m_1 + m_2) = E(m_1)*E(m_2) \bmod N^2$. (ii) The bth power of ciphertext $E(m_1)$ results in the encryption of the product of b and m_1; $E(m_1*b) = E(m_1)^b \bmod N^2$. (iii) Encrypting the same plaintexts with the same public key results in distinct ciphertexts (aka 'semantic security').

3 System Architecture and Secure Protocols

The system consists of four components: data owner (DO), authorized user (AU), and two clouds (C_A and C_B, respectively). The DO owns the original database (T) of n records. A record t_i $(1 \le i \le n)$ consists of m attributes and jth attribute value of t_i is denoted by $t_{i,j}$. To provide the indexing on T, the DO partitions T by using kd-tree. If we retrieve the tree structure in hierarchical manner, the access pattern can be disclosed. So, we only consider the leaf nodes of the kd-tree and all the leaf nodes are retrieved once during the query processing. Let h denote the level of the constructed kd-tree and F be the fanout of each leaf node. The total number of leaf nodes is 2^{h-1}. From now on, a node means a leaf node. Each node is represented as the lower bound $lb_{z,j}$ and the upper bound $ub_{z,j}$ for $1 \le z \le 2^{h-1}$ and $1 \le j \le m$. Each node stores the identifiers (id) of data being located inside the node region.

To preserve the data privacy, the DO encrypts T attribute-wise using the public key (pk) of the Paillier cryptosystem [8] before outsourcing the database. So, the DO generates $E(t_{i,j})$ for $1 \le i \le n$ and $1 \le j \le m$. The DO also encrypts the kd-tree nodes so as to support efficient query processing over the encrypted database. The lb and the ub of each node are encrypted attribute-wise, so $E(lb_{z,j})$ and $E(ub_{z,j})$ are generated for $1 \le z \le 2^{h-1}$ and $1 \le j \le m$. In the system, we assume that the clouds, C_A and C_B, act as semi-honest adversaries. This is because protocols under the *semi-honest* adversaries are efficient in practice and can be used to design

protocols against malicious adversaries. So, the C_A and C_B correctly perform the given protocols and do not exchange unpermitted data. However, an adversary may try to obtain additional information from the intermediate data during executing his/her own protocol.

To support range query processing over the encrypted database, a secure multiparty computation (SMC) is required between C_A and C_B. For this, the *DO* outsources the encrypted database and its encrypted index to the C_A with *pk* while the *sk* is sent to the C_B. The encrypted index includes the region information of each node in cipher-text and the *ids* of data that are located inside the node in plain-text. The *DO* also sends *pk* to *AU*s to enable them to encrypt a query. At query time, an *AU* first encrypts a query attribute-wise. Then, the *AU* sends $E(q.lb_j)$ and $E(q.ub_j)$ for $1 \leq j \leq m$ to C_A. C_A processes the query with the help of C_B and returns a query result to the *AU*.

As an example, assume that an *AU* has 8 data in two-dimensional space (e.g., x-axis and y-axis) as depicted in Fig. 1. The data are partitioned into 4 nodes for a kd-tree. To outsource the database, the *DO* encrypts each data and the information of each node attribute-wise. For example, t_1 is encrypted as $E(t_1) = \{E(2), E(1)\}$.

Meanwhile, our range query processing algorithm is constructed using several secure protocols. We describe the secure protocols that are used in our range query processing algorithm. All the protocols except SBN protocol are performed through the SMC technique between C_A and C_B. SBN protocol can be executed by C_A alone. Due to the space limitation, we first briefly introduce two existing secure protocols that we adopt from [7, 9]. SM (Secure Multiplication) protocol [7] computes the encryption of $a \times b$, i.e., $E(a \times b)$, when two encrypted data $E(a)$ and $E(b)$ are given as inputs. SBD (Secure Bit-Decomposition) protocol [9] computes the encryptions of binary representation of the encrypted input $E(a)$. The output is $[a] = < E(a_1), \ldots, E(a_l) >$ where a_1 and a_l denote the most and least significant bits of a, respectively. We use symbol $[a]$ to denote the encryptions of binary representation.

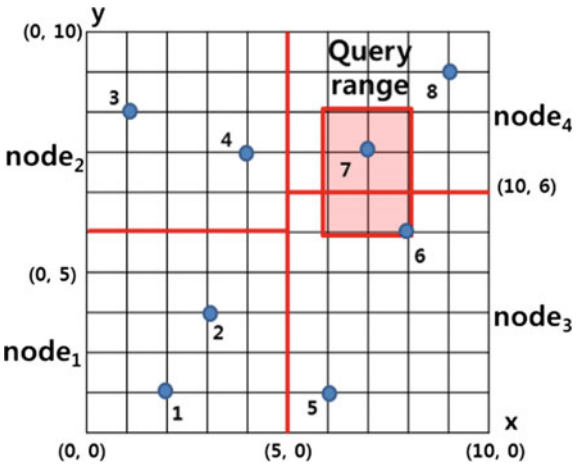

Fig. 1 An example in two-dimensional space

Next, we propose our new secure protocols. First, SBN (Secure Bit-Not) protocol performs the bit-not operation when an encrypted bit $E(a)$ is given as input. The output $E(\sim a)$ is computed by $E(a)^{N-1} \times E(1)$. Here, "-1" is equivalent to "N-1" under Z_N.

Second, SCMP (Secure Compare) protocol returns $E(1)$ if $u \leq v$, $E(0)$ otherwise, when $[u]$ and $[v]$ are given as inputs. We devise SCMP by modifying SMIN [7] protocol which outputs $[min]$ between two inputs $[u]$ and $[v]$. The variables generated during SMIN can be categorized into two folds. One set of the variables include hints about what the minimum value is. Another set of the variables is used to securely extract the minimum value. Because we only need the information about whether u is smaller or not, we only compute the former (e.g., W, G, H, Φ, L, L'). The goal of designing SCMP is to make the returned value from C_B be exactly opposite for the same inputs, based on the functionality selected by C_A.

The overall procedure of SCMP is as follows. (i) C_A appends $E(0)$ to the least significant bits of $[u]$ and $E(1)$ to the least significant bits of $[v]$. By doing so, SCMP makes u smaller than v only when two values are the same. (ii) C_A randomly chooses one functionality between F_0:$u > v$ and F_1:$v > u$. The selected functionality is oblivious to C_B. Then, C_A computes $E(u_i \times v_i)$ using SM and W_i, depending on the selected functionality. In particular, if F_0:$u > v$ is selected, C_A computes $W_i = E(u_i) \times E(u_i \times v_i)^{N-1}$. If F_1:$v > u$ is selected, C_A computes $W_i = E(v_i) \times E(v_i \times u_i)^{N-1}$. For F_0:$u > v$, $W_i = E(1)$ when $u_i > v_i$, and $W_i = E(0)$ otherwise. Similarly, for F_1:$v > u$, $W_i = E(1)$ when $v_i > u_i$, and $W_i = E(0)$ otherwise. (iii) C_A performs bit-xor between $E(u_i)$ and $E(v_i)$ and stores the result into G_i. C_A computes $H_i = (H_{i-1})^{ri} \times G_i$ and $\Phi_i = E(-1) \times H_i$ where $H_0 = E(0)$. Here, r_i is a random number in Z_N. Assume that j is the index of the first appearance of $E(1)$ in G_i. j means the first position where the minimum value between u and v can be determined. (iv) C_A computes $L_i = W_i \times \Phi_i^{ri}$ where L_i involves the information about which value is smaller between u and v at j. C_A generates L' by permuting L by using a random permutation function π_1 and sends L' to C_B. (v) C_B decrypts L' attribute-wise and checks whether there exists 0 in L_i' for $1 \leq i \leq l$. If so, C_B sets α as 1, and 0 otherwise. After encrypting α, C_B sends $E(\alpha)$ to C_A. By doing so, the returned values by C_B are exactly opposite with the selected functionalities for the same input which coincides with the goal of SCMP protocol. (vi) C_A performs $E(\alpha) = \text{SBN}(E(\alpha))$ only when the selected functionality is F_0:$u > v$ and returns the $E(\alpha)$. So, the final $E(\alpha)$ is $E(1)$ when $u \leq v$, regardless of the selected functionality. Note that the only information decrypted during SCMP is L' which is seen by C_B. However, C_B cannot obtain an additional information from $D(L')$ because the selected functionality is oblivious to C_B.

Third, SRO (Secure Range Overlapping) protocol returns $E(1)$ when $range_1$ overlaps $range_2$, $E(0)$ otherwise, when the encryptions of binary representation of two ranges $[range_1]$ and $[range_2]$ are given as inputs. Assuming that both $range_1$ and $range_2$ consist of $[lb_j]$ and $[ub_j]$, where $1 \leq j \leq m$, the two ranges overlap only if two following conditions are satisfied; (i) $E(range_1.lb_j) \leq E(range_2.ub_j)$ for $1 \leq j \leq m$, (ii) $E(range_2.lb_i) \leq E(range_1.ub_j)$ for $1 \leq j \leq m$. SRO determines the conditions by using our SCMP. The overall procedure of SRO is as follows.

(i) C_A initializes $E(\alpha)$ as $E(1)$. (ii) C_A obtains $E(\alpha')$ by performing SCMP([$range_1.lb_j$], [$range_2.ub_j$]) and updates $E(\alpha)$ by executing SM($E(\alpha)$, $E(\alpha')$). C_A repeats this step for $1 \leq j \leq m$. Similarly, C_A computes $E(\alpha')$ by performing SCMP([$range_2.lb_j$], [$range_1.ub_j$]) and updates $E(\alpha)$ by executing SM($E(\alpha)$, $E(\alpha')$) for all attribute values. Only when all conditions are satisfied, the value of $E(\alpha)$ remains $E(1)$. (iii) C_B returns the final $E(\alpha)$. Note that no decryption is performed during SRO except performing SCMP and SM protocols.

Finally, SPE (Secure Point Enclosure) protocol returns $E(1)$ when p is inside the range or on a boundary of the range, $E(0)$ otherwise, when the encryptions of binary representation of a point [p] and [$range$] are given as inputs. The overall procedure of SPE is identical to SRO. This is because if a low bound and an upper bound of a range is the same, the range can be considered as a point. However, we also define SPE protocol to make the relations between inputs clear.

4 Secure Range Query Processing Algorithm

In this section, we present our secure range query processing algorithm (SRange_I) using the kd-tree on the encrypted database. SRange_I consists of two steps; encrypted kd-tree search step and result retrieval step.

First, the procedure of the encrypted *k*d-tree search step is as follows.

(i) C_A computes [$q.lb_j$] and [$q.ub_j$] for $1 \leq j \leq m$ by using the SBD. C_A also computes [$node_z.lb_j$] and [$node_z.ub_j$] for $1 \leq z \leq num_{node}$ and $1 \leq j \leq m$ by using SBD where num_{node} is the total number of *k*d-tree leaf nodes. Then, C_A securely finds nodes which overlap the query range by executing $E(\alpha_z) \leftarrow \text{SRO}([q], [node_z])$ for $1 \leq z \leq num_{node}$. Note that the nodes with $E(\alpha_z) = E(1)$ overlaps the query range, but both C_A and C_B cannot know whether the value of each $E(\alpha_z)$ is $E(1)$.

(ii) C_A generates $E(\alpha')$ by permuting $E(\alpha)$ using a random permutation function π and sends $E(\alpha')$ to C_B. For example, SRO returns $E(\alpha) = \{E(0), E(0), E(1), E(1)\}$ in Fig. 1 as the query range overlaps the node_3 and node_4. Assuming that π permutes data in reverse way, C_A sends the $E(\alpha') = \{E(1), E(1), E(0), E(0)\}$ to C_B.

(iii) Upon receiving the $E(\alpha')$, C_B obtains α' by decrypting the $E(\alpha')$ and counts the number of α' with the value of 1. The number of $\alpha' = 1$ is stored into c. So, c means the number of nodes that overlaps the query range.

(iv) C_B creates c number of node groups (e.g., *NG*). C_B assigns to each *NG* a node with $\alpha' = 1$ and $num_{node}/c-1$ nodes with $\alpha' = 0$. Then, C_B computes *NG*′ by randomly shuffling the *ids* of nodes in each *NG* and sends *NG*′ to C_A. For example, C_B can obtain $\alpha' = \{1, 1, 0, 0\}$. However, C_B cannot correctly point out *ids* of the nodes overlapping the query range because the values in α' were permutated by C_A. As two node groups are required, C_B assigns $node_1$ and $node_2$ to NG_1 and NG_2, respectively. In this example, $num_{node}/c-1$ is

calculated as 1 because $num_{node} = 4$ and c = 2. So, C_B randomly assigns a node to each node group. Assume that C_B assigns $node_3$ to NG_1 and $node_4$ to NG_2. So, $NG_1 = \{1, 3\}$ and $NG_2 = \{2, 4\}$. Then, C_B randomly shuffles the *ids* of the nodes in each *NG*. The result can be like $NG_1' = \{1, 3\}$ and $NG_2' = \{4, 2\}$.

(v) C_A obtains NG^* by permuting the *ids* of nodes using π^{-1} in each NG'. In each NG^*, there exists only one node overlapping the query range. However, C_A cannot know the correct *id* of the node because the *ids* of the nodes in NG^* are shuffled by C_B. Sixth, C_A gets access to one datum in each node (e.g., $node_z$) for each NG^* and performs $E(t'_{i,j}) \leftarrow \text{SM}(node_z.t_{s,j}, E(\alpha_z))$ for $1 \le s \le F$ and $1 \le j \le m$. Here, α_z is the outputs of SPE, corresponding to the $node_z$. If a node has the less number of data than *F*, it performs SM by using $E(max)$, instead of using $node_z.t_{s,j}$, where $E(max)$ is the largest value in the domain. When C_A accesses one datum from every node in a NG^*, C_A performs a homomorphic addition such as $E(cand_{cnt,j}) \leftarrow \prod_{i=1}^{num} E(t'_{i,j})$, where *num* means the total number of nodes in the selected NG^*. By doing so, a datum in the node overlapping the query range is securely extracted without revealing the data access patterns. By repeating these steps, all the data in the nodes are safely stored into the $E(cand_{cnt,j})$ where *cnt* means the total number of data extracted during the index search.

As an example, C_A obtains $NG_1^* = \{2, 4\}$ and $NG_2^* = \{1, 3\}$ by permuting $NG_1' = \{3, 1\}$ and $NG_2' = \{4, 2\}$ using π^{-1}. Then, C_A accesses $E(t_3)$ in $node_2$, $E(t_7)$ in $node_4$ for NG_1^*. The results of SM using $E(t_3)$, e.g., $E(t'_1)$, are $E(0)$ for every attribute because the $E(\alpha)$ value of $node_2$ is $E(0)$. However, the results of SM using $E(t_7)$, e.g., $E(t'_2)$, become $E(7)$ and $E(7)$ for *x* and *y* dimension, respectively. So, the results of the attribute-wise homomorphic addition of $E(t'_1)$ and $E(t'_2)$ are $E(7)$ and $E(7)$ for *x* and *y* dimension, respectively. Thus, one datum $E(t_7)$ in node_4 is securely extracted into $E(cand_1)$. Similarly, values of $E(t_8)$ can be securely extracted into $E(cand_2)$ by using $E(t_4)$ and $E(t_8)$. In the same way, for NG_2^*, all the data in the $node_3$ (e.g., $E(t_5)$ and $E(t_6)$) are securely extracted into $E(cand_3)$ and $E(cand_4)$, respectively.

Second, the procedure of the result retrieval step is as follows. (i) C_A computes $\{([cand_{i,j}] \mid 1 \le i \le cnt,\ 1 \le j \le m\}$ by using the SBD. Here, *cnt* is equal to $F \times$ (the number of node groups).

(ii) C_A securely finds data inside the query range by executing $E(\alpha_i) \leftarrow \text{SPE}([cand_i], [q])$ for $1 \le i \le cnt$. Note that the data with $E(\alpha_i) = E(1)$ are included in the query range. However, both C_A and C_B cannot know whether the value of each $E(\alpha_i)$ is $E(1)$. For example, when $E(cand) = \{E(t_7), E(t_8), E(t_5), E(t_6)\}$ is given from the step 1, SPE returns $E(\alpha) = E(1)$ for $E(t_7)$ and $E(t_6)$, which are located inside the query range. The data with $E(\alpha) = E(1)$ should be sent to the user. To minimize the computation cost at the user side, it is required to send decrypted results. However, if the cloud decrypts the results, the actual content of the data are revealed to the cloud.

So, (iii) C_A computes $E(\gamma_{i,j}) = E(result_i) \times E(r_{i,j})$ for $1 \le i \le cnt$ and $1 \le j \le m$ by generating a random value $r_{i,j}$. C_A generates $E(\alpha')$, $E(\gamma')$, and r' by permuting $E(\alpha)$, $E(\gamma)$, and r, using a random permutation function π_1. Then, C_A sends $E(\alpha')$ and $E(\gamma')$ to C_B, and r' to AU, respectively. iv) C_B decrypts $E(\alpha_i')$ and $E(\gamma'_{i,j})$ for $1 \le i \le cnt$ and for $1 \le j \le m$. Then, C_B sends α' and γ' corresponding to the $\alpha' = 1$ to AU. Finally, AU computes $\gamma'_{i,j} - r'_{i,j}$ for $1 \le i \le cnt$ and $1 \le j \le m$ only for the corresponding $\alpha_i = 1$.

5 Performance Analysis

There is no existing range query processing algorithm that hides the data access patterns. So, in this section, we compare our SRange_I with a baseline algorithm SRange_B which only performs result retrieval step by considering all the data without using an index. We do the performance analysis of both schemes in terms of query processing time with different parameters. We used the Paillier cryptosystem to encrypt a database for both schemes. We implemented both schemes by using C++. Experiments were performed on a Linux machine with an Intel Xeon E3-1220v3 4-Core 3.10 GHz and 32 GB RAM running Ubuntu 14.04.2. To examine the performance under various parameters, we randomly generated synthetic datasets by following [7]. We set the size of the range as 0.1 which means the relational portion of the range compared to the total domain size (i.e., l). In addition, we set the domain size (l) as 12 and the encryption key size (K) as 1024.

Figure 2 shows the performance of SRange_I for varying the level of kd-tree. Figure 2a shows the performance of SRange_I for varying h and n. Overall, as the n becomes larger, the query processing time increases. Meanwhile, when the n increases, the h that shows the best performance becomes larger. For all cases, when the h is increased, the query processing time decreases for the smaller h while the query processing time increases for the larger h. Figure 2b shows the performance of SRange_I for varying h and m. As the m becomes larger, the query processing time linearly increases. This is because all the protocols including SRO and SPE should process additional data as the m increases. We set the h as 7 in the following

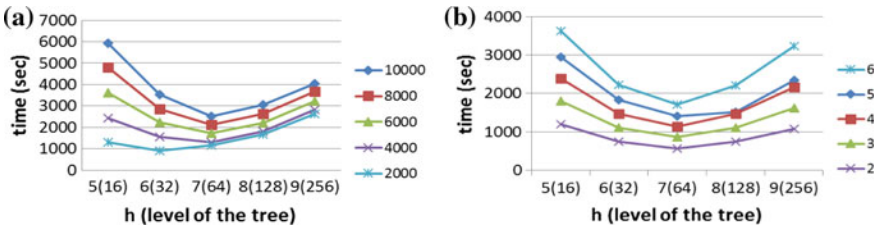

Fig. 2 Performance of SRange_I for varying h. **a** $m = 6$, $K = 1024$, **b** $n = 6$ k, $K = 1024$

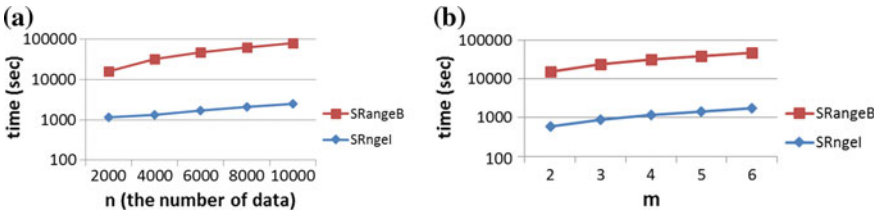

Fig. 3 Comparison of $SRange_I$ and $SRange_B$. **a** $m = 6$, $K = 1024$, **b** $n = 6$ k, $K = 1024$

performance evaluation because $SRange_I$ shows good performance when $h = 7$ in our parameter settings.

Figure 3a shows the performance of both $SRange_I$ and $SRange_B$ schemes varying the n. Overall, as the n becomes larger, the query processing time linearly increases for both schemes. Overall, $SRange_I$ shows much better performance than $SRange_B$ because $SRange_I$ filters irrelevant data by using the index. On average, $SRange_I$ shows about 25 times better performance than $SRange_B$. Meanwhile, Fig. 3b shows the performance of both schemes varying the m. Overall, as the m becomes larger, the query processing time linearly increases for both schemes. However, on average, $SRange_I$ shows about 27 times better performance than $SRange_B$.

6 Conclusion

With the popularity of the outsourced databases, researches on the range query processing methods over the encrypted database have been actively performed. They can preserve the data privacy and the query privacy, but there is no work that hides the data access patterns during the query processing. To solve the problem, we proposed a new secure range query processing algorithm on the encrypted database. Our method conceals the data access patterns while supporting efficient query processing by using our proposed encrypted index search scheme. We showed from our performance analysis that our algorithm achieves efficient query processing performance while hiding the data access patterns. As a future work, we plan to expand our work to support other query types, such as Top-k and skyline queries.

Acknowledgments This work was supported by the Human Resource Training Program for Regional Innovation and Creativity through the Ministry of Education and National Research Foundation of Korea (NRF-2014H1C1A1065816)

References

1. Yiu ML, Ghinita G, Jensen CS, Kalnis P (2010) Enabling search services on outsourced private spatial data. VLDB J 19(3):363–384
2. Hore B, Mehrotra S, Canim M, Kantarcioglu M (2012) Secure multidimensional range queries over outsourced data. VLDB J 21(3):333–358
3. Wang P, Ravishankar CV (2013) Secure and efficient range queries on outsourced databases using R-trees, ICDE, pp 314–325
4. Wang B, Hou Y, Li M, Wang H, Li H (2014) Maple: scalable multi-dimensional range search over encrypted cloud data with tree-based index, ASIACCS, pp 111–122
5. Kim H, Hong S, Chang J (2015) Hilbert curve-based cryptographic transformation scheme for spatial query processing on outsourced private data. Data Knowl Eng. doi:10.1016/j.datak.2015.05.002
6. Vimercati S, Foresti S, Samarati P (2012) Managing and accessing data in the cloud: privacy risks and approaches, CRiSIS, pp 1–9
7. Elmehdwi Y, Samanthula BK, Jiang W (2014) Secure k-nearest neighbor query over encrypted data in outsourced environments, ICDE, pp 664–675
8. Paillier P (1999) Public-key cryptosystems based on composite degree residuosity classes, EUROCRYPT, pp 223–238
9. Samanthula BK, Chun H, Jiang W (2013) An efficient and probabilistic secure bit-decomposition, ASIACCS, pp 541–546
10. Carmit H, Lindell Y (2010) Efficient secure two-party protocols: techniques and constructions, Springer Science & Business Media