Received: 12 March 2020 | Revised: 19 October 2020 | Accepted: 16 November 2020

DOI: 10.1049/ise2.12010

The Institution of Engineering and Technology WILEY

ORIGINAL RESEARCH PAPER

Time-specific encrypted range query with minimum leakage disclosure

Ozgur Oksuz

Computer Engineering Department, Adiyaman University, Adiyaman, Turkey

Correspondence

Ozgur Oksuz, Computer Engineering Department, Adiyaman University, Adiyaman, Turkey.
Email: ooksuz@adiyaman.edu.tr

Abstract

A time-specific encrypted range query scheme that has the following properties is proposed. (1) The proposed scheme has trapdoor privacy and data privacy so that a semi-honest cloud is not able to get any useful information from given ciphertexts and given tokens that are used for searching ranges. (2) Unlike most of the other studies which report that the cloud server stores single encrypted keyword/element in the database, in our solution, the cloud server stores encrypted multi-keywords/ranges in the database. Therefore, the semi-honest cloud compares ciphertexts to tokens as ranges based on a predefined threshold ϕ value. This approach decreases the cloud search time since the cloud compares ranges to ranges (multi-keywords with multi-keywords) not points to points (not a keyword with a keyword). Thus, the proposed scheme is efficient based on searching ranges on ciphertexts. (3) Moreover, the communication cost between users and the cloud is decreased from $O(n)$ to $O(\log\ n)$, where n is the size of a range. Users send logarithmic size information to the cloud server instead of sending linear size information.

1 | INTRODUCTION

Wireless (mobile) devices, especially pocket-size devices such as smartphones, tablets, smart watches are effective 'human sensors' because they can be carried by their owners. These wireless devices have been used to understand user mobility patterns [1–3], mobility characteristics [4] and group behaviours [5]. They have also been used in smart buildings [6], and locating emergencies in campuses [7].

Wireless network traces have been widely used to understand human behaviours. Traces record the movement of mobile devices (users). Thus, sensitive user information embedded in these traces such as the connection location of the user, Wi-Fi access point and duration at that Wi-Fi access point. If a user uses Internet then the user's location and time period that has been spent are going to be exposed. So these devices contain very sensitive user information. To protect users' sensitive information, sanitizing the trace is one of the possible solutions. For example, the identities of the users (e.g. their user names, ages, salaries or MAC addresses) are anonymized through a one-to-one mapping. However, recent studies have shown that sanitization-based techniques leak so much information about user privacy [8,9]. The intuition is that human mobility is rather unique and predictable [1,2]. Therefore, even when the identity is anonymized, a user can still be identified easily based on her unique mobility signature.

A time-specific encrypted range query scheme for wireless network traces is used. Encryption of wireless network trace provides much stronger protection of user's data than anonymization techniques. The time range data is not disclosed to any untrusted adversaries. Using wireless network traces that users use specific Wi-Fi access points for time ranges give us very useful applications that have been used in some studies such as in the work by Zhang et al. [10]. For example, a client can learn which building is mostly visited by users when the user gives a specific time range. This information is good to know when a new building is going to be constructed. In other words, with this information, we avoid having crowded buildings during rush hours. Moreover, it helps us construct new buildings to eliminate or tackle crowdedness. For forensic use, when the police try to find a missing person, they can figure out when the person has used the Internet the last time. Thus, this information gives the police where the person was seen last. Another application that checks if a user is in a specific place or not when a specific time range is also provided. For these applications, users' online information is

stored in the database, then a client wants to check if a specific user stayed online or used the service in the database in a given time range. To have precise results, the client also gives a threshold value ϕ that the client wants the cardinality of the intersection between its query time range and database time range at least ϕ. For database search time efficiency reason, it is a good idea to put users' time range as a whole into the cloud. When a user wants to query a range in the database, the cloud is able to compare users' time range to a given query range to check if the given query range is satisfied with the database based on the predetermined threshold. For example, two users visited a web page in time interval [1−2 pm] and the first user stayed online 10 s, the second user stayed online 100 s. If the threshold is 60 s (which means that the intersection cardinality of the given query range and user's time range in the database should be at least 60), the first one is not considered as an active user of the web page but the second one is. Most probably the first user visited the web page by mistake. Therefore, there should be a mechanism that it provides range to range comparison for efficient search. Encrypting time specific ranges for wireless network traces is a challenging task. The challenge is to encrypt the ranges and also keep the order in time for some useful applications. Most of the studies uses point data encryption [11–17]. These studies do not encrypt a range as a whole. To have these applications by using these studies, the range needs to be first broken down to single points then the encryption algorithm is applied to each single point. So, these studies are not efficient based on these applications. In the study by Zhang et al. [10], they have made a range to range comparison but it leaks very critical information. The study by Zhang et al. [10] uses deterministic encryption to encrypt ranges. Deterministic encryption leaks more about user's sensitive information. By deterministic encryption [10,18–20], the same message goes to the same ciphertext all the time when it is encrypted. If the encryption scheme is deterministic, an adversary can easily figure out which users have the same information by looking at the ciphertexts of the users. By means that it allows an untrusted adversary to do statistical attacks. These attacks result in a significant amount of information leakage to the adversary. In other words, the cloud learns the common ciphertexts of the users by looking at the database. The cloud sees the repeating elements when the users' have the common ciphertexts.

Order preserving encryption (OPE) schemes [21–23] can preserve the order in time, but they reveal the order of the data, which is a significant amount of information. We have developed an effective solution to resolve these challenges.

Contribution: We develop an effective solution to resolve these challenges. Our main contributions are as follows:

- We formulate a secure time range matching problem and propose two practical techniques to address the disadvantages that are mentioned above. Our schemes do not reveal order and do not allow an adversary to do statistical attacks on the data items. Our first scheme is basic scheme based on a simple quantitative technique and the other is enhanced scheme that is very efficient (communication) than the basic scheme. We rigorously analyse the security of these two techniques, and show that both techniques provide much stronger security than deterministic and order preserving encryption schemes.
- Our scheme provides efficient range encryption and search. Our enhanced scheme provides range encryption as a whole and provides range to range comparison. By means that it allows the database server to compare multiple points (range) in database to multiple points (range) in a given query that their intersection cardinality is at least ϕ, where ϕ is a predetermined threshold. Thus, the proposed approach provides better cloud search time than trivial cloud search time since the cloud compares ranges to ranges (multi-keywords with multi-keywords) not points with points (not a keyword with a keyword). Moreover, the cloud's search time complexity is better that the scheme in the study by Zhang et al. [10] (improved scheme) where the cloud server also compares ranges to ranges.
- Moreover, the communication cost between users and the cloud is decreased from $O(n)$ to $O(\log\ n)$, where n is the size of a range.

Outline: The rest of the paper is organized as follows. In Section 2, we introduce state-of-art range query encryption schemes. Section 3 describes the problem setting and our high-level approach. Section 4 gives cryptographic definitions of our schemes. Section 5 presents the two techniques for secure time range matching and security analysis. Last, Section 6 concludes the paper and outlines future directions.

2 | RELATED WORK

Our work is broadly related to searchable encryption, which allows search over encrypted data. A rich literature is on this topic (e.g. [24–26]). A recent study [12] points out that no off-the-shelf searchable symmetric encryption scheme supports range queries. Hence the authors reduce range search to multi-keyword search, introduce the first range searchable symmetric encryption (RSSE) framework and propose several practical RSSE schemes. Our work is independent of and in parallel with the study by Demertzis et al. [12]. The key difference is that we formulate and solve a secure time range matching problem, where both the queries and database records are time ranges (in the study by Demertzis et al. [12], only queries are ranges while database records are not ranges). In the study by Demertzis et al. [27], they extended their work done in their previous study [12] by introducing a new scheme. The new scheme in the study by Demertzis et al. [27] reduces the false-positive complexity but it increases the storage complexity.

Many studies have been proposed to support queries over encrypted database. Yang et al. [28] introduced efficient and secure methods for queries on encrypted data that is stored in an outsourced database. Hacigumus et al. [18] introduced a bucketization technique that divide the domain of a column into partitions, randomly map the partitions, and then store the partition number for each data item. Hore et al. [19] explored

the performance tradeoffs in setting the bucket sizes. Popa et al. [20] developed CryptDB that executes SQL queries over encrypted data using a collection of efficient SQL-aware encryption schemes. Our study differs in scope from them in that we focus on secure time range matching in wireless network traces, instead of general databases and general SQL queries.

Lu et al. [14] proposed a logarithmic search range query protocol that uses B+ structure. In their scheme, each range trapdoor size is $O(\log^2 L)$, where L is the number of leaves in the tree. Since the scheme uses B+ data structure, the ciphertexts are sorted in order and stored in the database server, the server can guess and figure out the order between a query value and a data value.

The studies by Li et al. [13,29] proposed range query processing scheme that the encryption of a range is also done by breaking the range to the single points as that is in the study by Demertzis et al. [12]. Furthermore, the work by Li et al. [13,29] does not satisfy our trapdoor privacy definition. Given range is converted the range to a minimum set of prefixes which means that they use 'Best Range Cover' (BRC) method which is not trapdoor private.

Pappas et al. [15] introduced a scalable searchable encryption (Blind Seer) scheme that it can perform not only range queries but also arbitrary Boolean queries. The scheme relies on a different trust assumption that the server is split to two different servers (Index Server and Server) and the servers are not be able to collude. Their scheme has a large amount of client-server interactions required to finish the entire search process. Moreover, the scheme in [15] has public-key operations in the search process that is less efficient than symmetric searchable encryption. Furthermore, a range of a record is not encrypted as a whole. The range is broken down to single points, then these single points are being encrypted one at a time.

Chi et al. [11] proposed a range query scheme that each user needs to interact with the data owner to get a trapdoor to search on the data. Therefore, the data owner needs to be online all the time. Moreover, they do not consider range encryption as a whole, they encrypt single value and put the encrypted value into the cloud. Furthermore, the study by Chi et al. [11] do not provide a concrete security definition and proof.

Boldyreva et al. [22] presented the first formal treatment of OPE. After they found that their scheme in their study [22] reveals half of the plaintext besides the order of the plaintext [30]. Popa et al. [23] proposed Order Preserving Encoding scheme, which is an interactive protocol between the data owner and the server. The study by Roche et al. [31] proposed a partial order preserving encoding that leaks the order of some of the inserted elements. Moreover, in the study by Roche et al. [31] searching phase is an interactive protocol between users and the server that introduces $O(\log_L n)$ rounds of communication, where L is the storage size and n is the insertion elements. The study by Rocheet al. [31] also leaks the order of some elements. The study by Kerschbaum and Tueno [32] provides a solution that does not leak the order but the server interactively traverses a structure like a binary tree asking client for a direction to go in inserting or searching phase. This results multiple round of interactions between the server and the client. Our basic and improved schemes provide stronger privacy than OPEs and do not require interaction.

Zuo et al. [16,17] proposed a searchable symmetric encryption that provides forward and backward privacy. The studies by Zuo et al. [16,17] do not solve our problem since these schemes are also based on point encryption. As an example, once a trapdoor range [0,2] is generated and sent to the server, the server sends a record that satisfies single points (0,1 or 2), not the record that satisfies a range (multiple points) that is a subset of [0,2] ([0,1], [1,2] or [0,2]). We would like to interest in intersection cardinality of a ciphertext with a trapdoor range which is at least $\phi = r$, where $r > 1$ and we want to solve this efficiently. Moreover, in these works, each keyword is mapped to a numeric value. Their schemes are not efficient for client side storage since the client needs to store all the keywords, $O(2W)$, where W is the number of keywords. Then the authors eliminate keeping linear number of keywords by introducing another construction based on Paillier cryptosystem. This new construction is based on public key encryption that introduces more computational cost than symmetric encryption. In our scheme, the client does not need to store keywords. Moreover, our scheme does not have public key encryption and has range to range comparison for efficient search.

The work by Miao et al. [33] presents an attribute-based keyword search with multi-authority. With multi-authority architecture, the proposed work eliminates single point performance bottleneck and expensive secret key distribution cost. This work does not focus on range query in encrypted database.

The study by Yang et al. [34] presents an expressive query over encrypted data. The proposed work supports single/conjunctive keyword query, range query, Boolean query and mixed Boolean query. There are bunch of range query protocols. In all these protocols, the data owner encrypts data as single keyword points so the ciphertexts are not as ranges. Moreover, there are two entities, cloud platform (CP) and computing service provider (CSP), work together to test if the trapdoor range covers a keyword point in encrypted database. Thus, the entities are not doing range (in database) to range (in a trapdoor) comparison for fast search.

The study by Qin et al. [35] presents a public key authenticated encryption with keyword search (PAEKS). The paper proposes a new security model for PAEKS that captures chosen ciphertext attack and keyword guessing attacks. This work also encrypts single keywords so the ciphertexts are not as ranges. Moreover, this study does not focus on range search in encrypted database.

The work by Peng et al. [36] presents a secure range query on geographical data. They have proxy re-encryption mechanism. Thus, the protocol has public key encryption. Moreover, the study by Peng et al. [36] also encrypts data as single points not as ranges. Each multi-dimensional point map to the single point (integer) with the help of LSH functions. Then the mapped points put into the buckets that these buckets contain

neighbour points. Once a query is issued by the client, all the points are retrieved from suitable buckets in the encrypted database. These bucket. also include fake points that they are sent back to the client. Then, the client checks if each point is covered by the range. Thus, in the study by Peng et al. [36], the client compares single data points in encrypted database to the range in a trapdoor. Furthermore, in the study by Peng et al. [36], the client has more false-positive results from cloud to eliminate. Thus, this introduces heavy communication bandwidth between the client and the cloud server. To eliminate this heavy communication, the authors introduce another protocol that has multiple cloud servers. This work by Peng et al. [36] is different than ours since their focus is for an efficient range query solution in two-dimensional geographical data while we focus on secure time range matching in wireless network traces.

Our solution is different than the above [33–36] solutions since in our work the data is encrypted as ranges for fast search. In other words, in our work an authorized user sends a range query to the public cloud and the cloud server does equality check on ranges. Moreover, our protocol uses lightweight symmetric key encryption. Our protocol uses only hash functions and AES encryption while the studies [33–36] use public key encryption which computationally more expensive than symmetric key encryption.

As can be seen from the above discussion, most of the studies have focused on encrypting a single element and storing it on the cloud. With a given range as trapdoor, the cloud service checks if the given range covers the encrypted single element in encrypted database. Our work differs from these studies since we want to encrypt a range as a whole (not a single value) and we want the cloud service to find the intersection of the encrypted range with the given trapdoor range with a given predetermined threshold. Comparing the range to range mechanism provides a very efficient search mechanism to the server. This mechanism provides a more efficient range search scheme than the above studies.

3 | PROBLEM SETTING

We consider a wireless network trace that can be a cellular network trace or Wi-Fi network trace that contains identity, spatial and temporal information. In other words, the trace consists of identity of a user, spatial information of the user's connection and the user's temporal information of the connection. In a Wi-Fi network, the information can express when a mobile device is associated with an access point and the duration of the association; in a cellular network, the information can express when a phone is connected to a cellular tower and the duration of the connection. For simplicity, we assume it is of the form depicted in Table 1 and consists of four parts (assuming the user uses a mobile device):

- an identity of the user, *ID*,
- an access point of the user, *AP*,
- a starting time of the connection, *t*,
- a duration of the connection.

TABLE 1 An example of a Wi-Fi network trace

ID	Location	Start time	Duration (s)
ID_1	AP_1	t_1	10
ID_2	AP_2	t_2	100

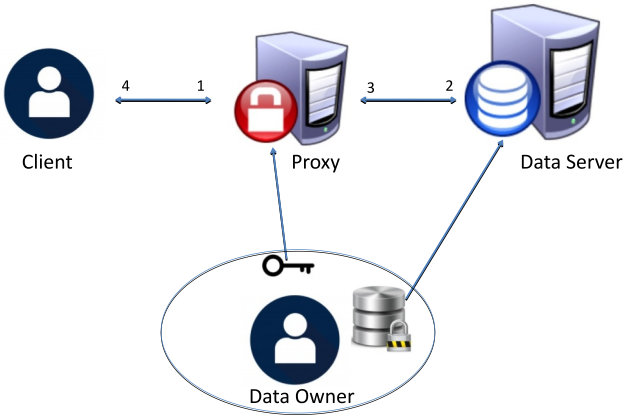

FIGURE 1 System architecture and workflow

Instead of any anonymization technique, we use an encryption-based approach (which provides much stronger protection) to protect user privacy. Our focus is to achieve an efficient and secure time-specific encrypted range query scheme.

3.1 | System architecture and work-flow

We use a proxy-based architecture to make the data encryption transparent to the client. Proxy-based architecture is not new and has been used several works [20,28]. Our architecture is illustrated in Figure 1. There are four entities in our system: a data owner, a client/clients, a proxy and a data server.

The data server stores the encrypted data. As the data are stored as ciphertext, a proxy equipped with encryption keys is introduced as an intermediary between a client application and the data server. Multiple clients can send queries simultaneously to the data server (through the proxy). We treat client and user interchangeably.

The work-flow of our system is illustrated in Figure 1 as follows:

- In the initialization step, the data owner encrypts the data, stores the encrypted data in the database and sends secret keys to the proxy through a secure channel.
- In the query step: (1) The client submits query to the proxy as plaintext; (2) The proxy converts the query into ciphertext, and sends it to the data server; (3) The data server retrieves the data from the database, and performs necessary processing, and returns the results to the proxy and (4) The proxy decrypts the data and returns the plaintext results to the client.

Instead of using deterministic data that reveals the same plaintexts are mapped to the same ciphertexts in the studies by Boldyreva et al. [22,30] and Zhang et al. [10], we propose a randomized approach that avoids the adversary to do statistical attack. The adversary cannot be able to figure out if the two ciphertexts have the same plaintext by looking at the encrypted data. Basically, it provides a stronger security and encrypts wireless network traces, performs analysis over the encrypted data and then decrypts the output of the analysis to reveal the final results to end users. In this way, data will be encrypted once captured and never revealed in the plaintext. Our goal is twofold: providing provable and a stronger security, while being still simple and practical. As an example, fully homomorphic encryption allows an arbitrary operation to be performed on encrypted data. It is, however, not practical since existing homomorphic encryption techniques are prohibitively slow. We choose light-weight symmetric-key encryption schemes that provide stronger security, and use a single proxy-based architecture to make the data encryption transparent to the client, and thus allowing legacy client applications to be used directly in our system.

3.2 | Focus of the study

In general, we are dealing with wireless network data that provide temporal and spatial information. Our first goal is to mine wireless network data to extract useful information and to have useful applications, for example, how many devices (people) are connected to an access point; whether a person is in a building or not; to find a missing person. To examine and analyse these information and to have these applications, time-specific range query is important. To have precise result for these applications, a threshold needs to be set. The threshold provides realistic results to the querier (client). The importance of time-specific range query and the need of the threshold are given in Section 1.

Our second goal is developing practical encryption techniques for time range that do not reveal order and do not reveal if two users share the same time range while maintaining utility of the data.

A small example of a data owner's database is illustrated in Table 1. In this table, time range field is represented by multiple values (ranges) while the other fields are represented as a single value. Since our focus is secure time matching problem with a predefined threshold value, one primary challenge is how to encrypt time range an efficient and a privacy-preserving approach. There have been several studies [10,18,19,23] which provides some level of privacy with deterministic encryption schemes but the untrusted server is easily able to do statistical attacks. Even if the adversary sees the encrypted database, it is able to say if two users are in the same building at the same time. Moreover, the adversary extracts the people that share the same time range and building by looking the encrypted database. This leakage is very huge that the adversary is able to map the encrypted range to the plaintext. For example, a cafeteria of a university is more crowded than other buildings at the noon time. one can analysis the deterministically encrypted database by mapping a single time value to the number of students. By doing statistical analysis, easily one can map noon time to corresponding encrypted time.

Another insecure encryption method is OPE. OPE schemes preserves the order in time, the order in time that they reveal may expose the identity of a user. For instance, suppose that we can infer that a user visits buildings X, Y, X and Z (in the order of time) every Friday, where X is the Computer Science building, and Y and Z are classroom buildings. Suppose only one faculty in Computer Science teaches in building Y and then in building Z every Monday, then we can infer that the user is likely to be this faculty even though we do not know exactly what time the user visits these buildings.

The problem we focus on is secure time range matching when given a trapdoor range and each row's time range field in the database. The goal is to extract the rows that have time range overlapping with the queried time range for at least a threshold value ϕ. As an example, suppose the query is to find the number of users that were seen between time t_1 and t_2. Then the corresponding secure time range matching problem is to find each row in the database with its time range t_1' and t_2' so that $[t_1, t_2] \cap [t_1', t_2'] \geq \phi$. The server computes all these operations (comparisons) in ciphertexts. Suppose another query is to find all the connections that are at location AP_1 between time t_1 and t_2. In this case, besides the time range matching as before, an additional condition is that the location needs to equal to AP_1.

3.3 | Security model

In our system, the proxy is fully trusted. The users that are able to search ranges are also trusted. The server is semi honest adversary. It follows the protocol specification but tries to learn the authorized users' queries and data owner's database. The server does not change the database and the users' queries. Moreover, it sends the results to the users without changing the results.

We focus on data privacy and trapdoor privacy. Data privacy means that any sensitive information (plaintext) and any statistical information about database are not leaked to the server. Query privacy means that the server is not able to figure out what an authorized user searches in the data base when the user issues a query.

4 | DEFINITIONS

In this section, we give some technical preliminaries, secure range index and its security definitions.

Definition 1 (Pseudo-Random Generator): A pseudo-random generator (PRG) outputs strings that are computationally indistinguishable from random strings. More precisely, we say that a function G: $\{0,1\}^n \rightarrow \{0,1\}^\lambda$, where $\lambda > n$ is a (t, ϵ, q)-pseudo-random generator if

1. G is efficiently computable by a deterministic algorithm,
2. for all t time probabilistic algorithm A that makes at most q adaptive queries, $|\Pr A(G(s)) = 0 | s \leftarrow \{0,1\}^n] - \Pr A(r) = 0 | r \leftarrow \{0,1\}^\lambda]| \leq \epsilon$

Definition 2 (Pseudo-Random Function): A pseudo-random function (PRF) is computationally indistinguishable from a random function—given pairs $(x_1, f_k(x_1)), \ldots, (x_m, f_k(x_m))$, an adversary cannot predict $f_k(x_{m+1})$ for any x_{m+1}. More precisely, we say that a function f: $\{0,1\}^s \times \{0,1\}^n \rightarrow \{0,1\}^n$ is a (t, ϵ, q)-pseudo-random function if

1. $f(k, x) = f_k(x)$ can be computed efficiently from input $x \in \{0,1\}^s$ and key $k \in \{0,1\}^n$.
2. for any t time oracle algorithm A that makes at most q adaptive queries, $|\Pr A^{f_k(.)} = 0 | k \leftarrow \{0,1\}^n] - \Pr A^g = 0 | g \leftarrow F : \{0,1\}^s \rightarrow \{0,1\}^n]| \leq \epsilon$

Definition 3 (Secure Range Indexes): A secure range index scheme consists of the following four algorithms.

KeyGen(n): Given a security parameter n, it outputs the master private key Kpriv.

RanBInd(Kpriv, DR): Given a data range DR and the master key Kpriv, outputs range index IDR.

Trap(Kpriv, TR): Given the master key Kpriv and a trapdoor range TR, it outputs the trapdoor TTR for TR.

RanSInd(TTR, IDR): Given trapdoor TTR and data range index IDR, outputs 1 if $|\text{DR} \cap \text{TR}| \geq \phi$, where ϕ is predefined threshold that is the number *of common elements in given data range and trapdoor range; and 0 otherwise (*$|DR \cap TR| < \phi$). Except for KeyGen, which is a randomized algorithm, the other three algorithms are deterministic.

Definition 4 (IND-CKA): Our users' data privacy is achieved by using semantic security against adaptive chosen keyword attack, IND-CKA as short. This model is proposed in the study by Goh [37] and widely used in many studies [13,29,37]. This model aims to capture the notion that an adversary A cannot deduce a range contents from its index. The security game works as follows: Suppose the challenger C gives the adversary A two equal length data ranges DR_0 and DR_1, each containing some (possibly unequal) number of elements, together with an index. Here, A's challenge is to determine which range is encoded in the index. If the problem of distinguishing between the index for DR_0 and DR_1 is hard, then deducing at least one of the elements that DR_0 and DR_1 do not have in common from the index must also be hard. If A cannot determine which document is encoded in the index with probability non-negligibly different from 1/2, then the index reveals nothing about its contents. We use this formulation of index indistinguishability (IND) to prove the semantic security of indexes. We note that secure indexes do not hide information such as range sizes that can be obtained by examining the encrypted ranges. More precisely, let (KeyGen, Trap, RanBInd, RanSInd) be a range index scheme. We use the following game between a challenger C and an attacker A to define semantic security against an adaptive chosen keyword attack (IND-CKA).

Setup: Challenger C picks a threshold ϕ value, creates a data range DR of q (consecutive) elements and gives this to adversary A. A chooses a number of range subsets from DR. The collection of range subsets is called DR^* and is returned to C. Upon receiving DR^*, C runs KeyGen to generate a master secret key K_{priv}, and for each range subset in DR^*, C encodes its contents into an index using RanBInd algorithm. Finally, C sends all indexes with their associated range subsets to A.

Queries: A is allowed to make trapdoor queries. For a trapdoor range query TR, C runs Trap algorithm and sends T_{TR} it to A. With T_{TR}, A can invoke RanSInd on an index $I \in DR^*$ to determine if $|T_{TR} \cap I| \geq \phi$.

Challenge: After making some trapdoor queries, A decides on a challenge by picking a non-empty range subset $DR_0 \in DR^*$, and generating another non-empty range subset DR_1 from DR such that $|DR_0 - DR_1| \neq 0$, $|DR_1 - DR_0| \neq 0$, and the total length of elements in DR_0 is equal to that in DR_1. Lastly, A must not have queried C for the trapdoor of any element in $DR_0 \triangle DR_1$, where $\triangle$ is the symmetric difference of two range DR_0 and DR_1. Next, A gives DR_0 and DR_1 to C who chooses $b \leftarrow \{0, 1\}$, invokes RanBInd(DR_b, K_{priv}) to obtain the index I_{DR_b} for DR_b, and returns I_{DR_b} to A. The challenge for A is to determine (b) After the challenge is issued, A is not allowed to query C for the trapdoors of any point or range $TR \in DR_0 \triangle DR_1$.

Response: A eventually outputs a bit b', representing its guess for (b). The advantage of A in winning this game is defined as $Adv_A = |\Pr b = b'] - 1/2|$, where the probability is over A and C's coin tosses.

We say that an adversary A (t, ϵ, q)-breaks a range index if Adv_A is at least ϵ after A takes at most t time and makes q trapdoor queries to the challenger. We say that I is an (t, ϵ, q)-IND-CKA secure range index if no adversary can (t, ϵ, q)-break it.

Remark 1: Even if the study by Goh [37] defines data privacy for his scheme but he does not examine trapdoor privacy since the trapdoor contains only one element. In our scheme, since our trapdoor contains multiple keywords/range we should define trapdoor privacy. We want to make sure that our scheme does not leak anything about intended keywords/range that it wants to be searched by an authorized user to the adversary.

Definition 5 (Trapdoor Privacy): Trapdoor privacy assures that the trapdoor does not leak the intended sensitive keyword (s) (or ranges) to the adversary. The privacy game between an attacker A and a challenger C is given as follows.

Setup: The challenger C randomly selects k from K.

Queries (phase 1): The adversary A is allowed to interact with C and asks two types of queries:

- encryption/trapdoor queries for a value x; to those queries C responds with $T_x \leftarrow Trap(k, x)$ for the trapdoor and $I_x \leftarrow RanBInd(k, x)$ for encryption.
- encryption/trapdoor queries for a range TR; to those queries C responds with $T_{TR} \leftarrow Trap(k, TR)$ for the trapdoor and $I_{TR} \leftarrow RanBInd(k, TR)$ for encryption.

Challenge: A submits two ranges TR_0, TR_1 to be challenged on. C responds as follows: it flips a coin $b \in \{0, 1\}$ and sends T_{TR_b} to A

Queries (phase 2): A makes the queries as that are in phase 1.

Response: A terminates by returning a single bit b'. The advantage of A in winning this game is defined as $Adv_A = |\Pr b = b'] - 1/2|$, where the probability is over A and C's coin tosses.

Definition 6 (Bloom Filter): A Bloom Filter is data structure that represents a set of $R = \{t_1, \ldots, t_{m'}\}$ of m' (or range) elements (we use time range instead of using elements) and its represented by an array of λ bits that all are initially set to 0. The filter uses r independent hash functions $h_1, \ldots, h_r$, where $h_i : \{0,1\}^* \rightarrow [1, m]$ for $i \in [1, r]$. For each point/range t_i in R, where $i = 1, \ldots, m'$, the array bits at positions $h_1(t_i), \ldots, h_r(t_i)$ are set to 1. A location can be set to 1 multiple times, but only the first is noted. There is, however, some probability of a false positive, in which t_i appears to be in R but actually is not. False positives occur because each location may have also been set by some element other than t_i.

5 | MINING OVER ENCRYPTED WIRELESS DATA

In this section, we describe two schemes: Basic and Enhanced schemes. We first describe how we encrypt wireless network traces for both of the schemes, with particular attention to encrypting time range with a stronger security. We then describe query over the encrypted data with a single proxy setting, improvement techniques, and security analysis of our approach.

5.1 | Overview

To encrypt time column is more complicated if we want the searchable encryption scheme is more secure than the previous studies [10,22,23,30] that allow the adversary to do statistical attacks on encrypted data and the scheme that is as efficient as that are in the studies by Boldyreva et al., Popa et al. and Zhang et al. [10,22,23,30]. We will show that we have obtained these properties (secure and efficient) by presenting two searchable encryption schemes in this section. Our first scheme (basic) is not communication efficient. The trapdoor sizes are linear in the range size. Then we introduce second scheme (enhanced) that has logarithmic size trapdoors in the size of range. Moreover, it provides better search time than the trivial case and the improved scheme in the study by Zhang et al. [10].

5.2 | Basic scheme

Our secure range index for basic scheme construction is similar to the scheme that is in the study by Goh [37]. There are some differences between our basic scheme and the scheme that is in the study by Goh [37]. The first difference is that our scheme encrypts multi-keywords (range) at a time instead of encrypting one element at a time in the study by Goh [37]. The second difference is that, a user makes multi-keyword queries (range) at a time instead of making only a single keyword query in [37]. The third difference is that we use GGM-tree to implement a pseudo-random function (PRF) in our schemes.

The GGM-based PRF [38] is built upon the well-known tree-based GGM-PRF family. This PRF is based on the hierarchical application of any length-doubling pseudo-random generator (PRG) according to the structure induced by a tree, where input values are uniquely mapped to root-to-leaf paths. Specifically, Let G be a publicly known PRG that takes a n-bit string $k \in \{0,1\}^n$ (secret random seed) as input, and outputs a $2n$-bit string, $G(k)$. Let $G_0(k)$ and $G_1(k)$ denote, respectively, the first and second half of $G(k)$. The GGM-PRF [38] is defined as $F = \{f_k : \{0,1\}^n \rightarrow \{0,1\}^n\}_{k \in \{0,1\}^n}$ such that $f_k(x_{n-1} \ldots x_0) = G_{x_0}(G_{x_1}(\cdots(G_{x_{n-1}}(k))))$, where $x_i = \{0, 1\}$.

In our basic scheme construction, range is represented in an index by codeword derived by applying PRFs twice: the first PRF (GGM-PRF) takes the key and the time range as input. The second PRF (GGM-PRF) takes the output of the first PRF as input and the index which is basically the identity of the bloom filter. In this technique, the codewords representing a time range t are going to be different for each record.

Using PRFs twice is the trick that our schemes break the determinism of the ciphertexts and eliminates any statistical attacks that the adversary does. Even the two records, R_1, R_2 have the same plain range $[r_1, r_2]$ value in their rows, the encryption of $[r_1, r_2]$ is going to be different for rows R_1, R_2 due to using PRFs twice.

5.2.1 | Basic scheme

KeyGen(λ): Given a security parameter λ, the data owner chooses a length doubling PRG G: $\{0,1\}^\lambda \rightarrow \{0,1\}^{2\lambda}$, chooses PRG seeds (secret values) $K_{priv} = (k_1, \ldots, k_r) \in \{0,1\}^{r\lambda}$ and a threshold value ϕ. Then, the data owner gives K_{priv}, G and ϕ to the proxy and gives (G, ϕ) to the server.

RanBInd(R, K_{priv}): The input is the range $R = \{t_1, \ldots, t_w\}$ comprising an unique identifier $R_{id} \in \{0,1\}^\lambda$ and a list of time points $(t_1, \ldots, t_w)$, and $K_{priv} = (k_1, \ldots, k_r)$.

1. For each unique time point t_j for $j \in [1, w]$, the data owner
   a. Computes $q_j = ((x_1)_j = f_{k_1}(t_j), \ldots, (x_r)_j = f_{k_r}(t_j))$,
   b. Sets the codeword for $t_j \in R$: $((y_1)_j = f_{(x_1)_j}(R_{id}), \ldots, (y_r)_j = f_{(x_r)_j}(R_{id}))$,
   c. Inserts the codeword $(y_1)_j, \ldots, (y_r)_j$ into document R_{id}'s Bloom filter BF.
2. Computes an upper bound u on the number of time points in R. For example, an extreme value for u assumes one time point for every byte in R (after encryption).
3. Let v be the number of unique words among the set of m time points $(t_1, \ldots, t_w) \in R$. Blind the index by inserting $(u - v).r$ number of 1's uniformly at random in the Bloom

filter (possibly with replacement); This is equivalent to adding $u - v$ random words into the index, but without any pseudo-random function computations.
4. Outputs $I_{R_{id}} = (R_{id}, BF)$ as the index for R_{id}.

Trap (K_{priv}, R^*): Given the master key $K_{priv} = (k_1, \ldots, k_r) \in \{0,1\}^{r\lambda}$ and a range $R^* = \{t_1^*, \ldots, t_\xi^*\}$, the proxy computes $(t_i)^*$ in R^* as $Tr_{t_i^*} = ((x_1)_i^* = f((t_i)^*, k_1), \ldots, (x_r)_i^* = f((t_i)^*, k_r))$. Then, it gives $Tr_{t_i^*}$, where $i = 1, \ldots, \xi$ to the server.

RanSInd($Tr_{R^*}, I_{R_{id}}, \phi$): Given the trapdoor $Tr_{t_i^*}$, where $i = 1, \ldots, \xi$, and the index $I_{R_{id}} = (R_{id}, BF)$, the server

1. Sets $sum = 0$ and, compute the codeword for each $(t_i)^*$ in R_{id} : $(y_1)_i^* = f_{(x_1)_i^*}(R_{id}), \ldots, (y_r)_i^* = f_{(x_r)_i^*}(R_{id})$.
2. Test if BF contains 1 in all r locations denoted by $(y_1)_i^*, \ldots, (y_r)_i^*$.
3. If so, $sum \leftarrow sum + 1$ and check if $sum \geq \phi$,
4. If this inequality holds, outputs 1; Otherwise, repeat the steps [1–3] for t_{i+1}^*,
5. If this inequality does not hold for all t_i^*, where $i = 1, \ldots, \xi$, it outputs 0.

5.3 | Enhanced scheme

In this subsection, we introduce the main idea of the enhanced scheme. The basic scheme has some disadvantages especially results a significant communication cost. This is because each time range is going to be broken into single time values then each single time value maps to a single PRF value. The enhanced scheme reduces the communication cost of the basic scheme. Our enhanced scheme is based on Delegatable Pseudorandom Function (DPRF) [39] which is a cryptographic primitive that a user delegates the evaluation of a PRF to an untrusted server. The DPRF construction is based on GGM framework to generate PRF. The server is able to evaluate PRF on a range using a trapdoor derived from the DPRF secret key. The trapdoor is generated based on a certain policy that determines the subset of input values that the proxy is allowed to compute. The study by Kiayias et al. [39] introduces two efficient constructions. One of them which is Uniform Range Cover (URC) construction has query privacy on 1-dimensional ranges. We adopt URC algorithm for generating trapdoor ranges and encrypting time ranges in our scheme. The encryption of a time range is a bit complicated that the basic scheme because in the enhanced scheme the internal nodes of the tree also used to be encrypted so the level of these nodes also are getting involved. In the basic scheme, only the leaf nodes are encrypted.

The main idea of the trapdoor generation algorithm in enhanced scheme is that every internal node in GGM tree covers a set of leaf nodes, which corresponds to a time range. As an example, in Figure 2, the internal nodes {01, 001} covers the range [2, 7]. Therefore, instead of representing time range [2, 7] using $f_m(0010)$, $f_m(0011)$, …, $f_m(0111)$, one can simply represent it as two partial PRFs $f_m(001) = G_1(G_0(G_0(m)))$

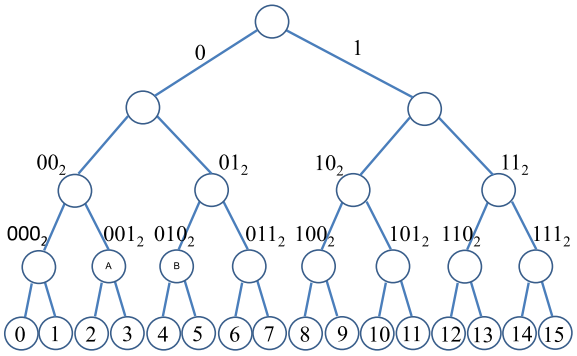

FIGURE 2 Illustration of a 4-level GGM tree

and $f_m(01) = G_1(G_0(m))$. This provides very efficient communication between a client and the proxy. In other words, for time range [2, 7], if we use four bytes to represent $f_m(\cdot)$ and one byte to represent heights, then the communication overhead is to $4 + 1 + 4 + 1 = 10$ bytes, significantly lower than the overhead of $6 \times 4 = 24$ bytes in the basic scheme.

In general, for a time range $[a, b]$, the enhanced scheme finds a set of internal nodes that covers the time range, and uses the set of partial PRF values for the internal nodes (along with their heights) to represent the time range. Since there may exist multiple sets of internal nodes that can cover a time range, we will find a set that incurs the minimum trapdoor size, that is containing the minimum number of internal nodes that covers the time range. The *Best Range Cover* (BRC) scheme [39] achieves this goal. Specifically, for time range $[a, b]$, it first finds the common path from the root to leaves a and b Suppose the common path ends at node u. It then recursively goes through the left and right subtrees of u to find the internal nodes that cover $[a, b]$ (see more details of the algorithm in the study by Kiayias et al. [39]). For time range $[a, b]$ with size $r = b - a + 1$, the communication overhead under BRC is $O(\log\ l)$ [39], only logarithmic of the communication under the basic scheme.

While BRC is optimal in trapdoor generation, it may leak information. As an example, two ranges [2,11,21,38] have the same length $l = 6$, while BRC generates two distinguishable representations, $\{(f_m(001), 1), (f_m(01), 2)\}$ and $\{(f_m(101), 1), (f_m(1001), 0), (f_m(110), 1)), ((f_m(1110), 0)\}$, thus leaking information. The *URC* scheme [39] solves the above problem. It modifies the output of BRC to generate a uniform output for a given range size l (see more details of URC in the study by Kiayias et al. [39]). Using URC, the set of PRF values generated for the studies by Agrawal et al. and Chi et al. [11,21] is. $\{(f_m(010), 1), (f_m(0010), 0), (f_m(011), 1), (f_m(0011), 0)\}$, which becomes indistinguishable from that for the studies by de Montjoye et al. and Goldreich et al. [2,38]. Indeed, these two representations have the same number of elements with pairwise equal heights. It is shown in the study by Kiayias et al. [39] that URC preserves the same communication overhead $O(\log\ l)$. We adopt URC scheme. Instead of storing plain partial PRFs in the database that is used in the study by Zhang et al. [10], we use the secure range index scheme that we showed in the basic scheme to avoid having statistical attacks from an adversary. To have an efficient range

search scheme, however, the integration of using DPRF algorithm for generating trapdoor range and using secure index algorithm for generating range encryption are *notstraightforward*. In the reality, it is very complicated. The problem is that how the server compares given trapdoors that consist of partial PRF and its level to given ciphertexts that consist of bloom filters in database. The work by Zhang et al. [10] allows the server to delegate partial PRF value with its level in both sides (trapdoors and encrypted database) to find matches since trapdoors and ciphertexts consist of partial PRFs and their levels. But it is not secure since the adversary can do statistical attacks. Our new scheme does not have this functionality to eliminate statistical attacks from the cloud server. To have an efficient search mechanism, the cloud server somehow needs to compare partial PRF value and its level from given trapdoor to partial PRF value and its level from given encrypted database. In our new scheme, trapdoors consist of partial PRFs with their levels and database consists of bloom filters. To allow the server to compare trapdoor values to database values in an efficient and a secure way, we use Multi-Bloom Filters (MBF) for each record to allow the server to do partial PRF (from trapdoor) to partial PRF (from encrypted database) and level (from trapdoor) to level (from encrypted database) comparisons since each PRF is defined with its level. In order to have a match in trapdoor and database, partial PRFs and their levels should be the same. Therefore, we also assign a level to a bloom filter. This provides efficient search mechanism.

5.3.1 | Construction

Keygen(λ): Given a security parameter λ, the data owner chooses a length doubling PRG G: $\{0,1\}^{\lambda} \rightarrow \{0,1\}^{2\lambda}$, a threshold value ϕ and chooses PRG seeds (secret values) $K_{priv} = (k_1, \ldots, k_r) \in \{0,1\}^{r\lambda}$. Then, the data owner gives K_{priv}, G, ϕ to the proxy and gives (G, ϕ) to the server.

RanBInd(R_i, K_{priv}): Given the range $R_i = [a', b']$ (for record i), and $K_{priv} = (k_1, \ldots, k_r)$, the data owner,

1. For record i, creates h Bloom filters $BF_{i,0}$, .., $BF_{i,h-1}$ that each Bloom filter is m-bits. Then the data owner sets all the entries to 0, where h is the height of GGM-tree, T_q, where $q = 1, \ldots, r$.
2. Assigns $id_{i,j} \in \{0,1\}^{\lambda}$ for each Bloom Filter $BF_{i,j}$, where $j = 0, \ldots, h-1$.
3. Takes each k_q, where $q = 1, \ldots, r$ as a seed of GGM tree T_q and $R_i = [a', b']$ (for record i), then
   a. Computes $(((x_1)_1 = f_{k_1}(R_{i,1})), \ell_{u_{i,1}}), \ldots, (((x_1)_s = f_{k_1}(R_{i,s})), \ell_{u_{i,s}}), \ldots, (((x_r)_1 = f_{k_r}(R_{i,1})), \ell_{u_{i,1}}), \ldots, (((x_r)_s = f_{k_r}(R_{i,s})), \ell_{u_{i,s}})$ by running URC trapdoor generation algorithm in the study by Kiayias et al. [39], where $R_{i,1} \cup \ldots \cup R_{i,s} = R_i$, $s = O(\log R_i)$, $\ell_{u_{i,s'}}$ ($s' = 1, \ldots, s$) is the level information of partial PRFs $f_{k_q}(R_{i,s'})$ ($q = 1, \ldots, r$) and $(h-1) \geq \ell_{u_{i,1}} \geq \ell_{u_{i,2}} \geq \ldots \geq \ell_{u_{i,s}} \geq 0$.
   b. Takes the highest level of the partial PRFs from step-[a] that these are $(x_1)_1, \ldots, (x_r)_1$, and computes the code-word for level-$\ell_{u_{i,1}}$: $((y_1)_1 = f_{(x_1)_1}(id_{i,\ell_{u_{i,1}}}), \ldots, (y_r)_1 = f_{(x_r)_1}(id_{i,\ell_{u_{i,1}}}))$,
   c. Inserts the code-word $(y_1)_1, \ldots, (y_r)_1$ into $BF_{i,\ell_{u_{i,1}}}$.
   d. Derives level-$\ell_{u_{i,1}}$ partial PRFs $(x_1)_1, \ldots, (x_r)_1$ by using PRG G (this results two level-($\ell_{u_{i,1}}$-1) partial PRFs for each $(x_q)_1$, where $q = 1, \ldots, r$) and follow the steps—[b], [c] and [d] until level-0 is obtained from partial PRFs $(x_q)_1$.
   e. Applies step—[b], [c] and [d] for remaining partial PRFs$(x_1)_2, \ldots, (x_r)_2, \ldots, (x_1)_s, \ldots (x_r)_s$.
4. Assigns an upper bound $UB_{\ell_{u_j}} = max(v_{\ell_{u_{i,j}}})_{i=1}^{n}$ (assuming there are n users) on the number of partial PRF values to each record's level ℓ_{u_j}, where $\ell_{u_j} \in \{0, \ldots, h-1\}$, and $v_{\ell_{u_{i,j}}}$ is the number of unique partial PRFs at level-ℓ_{u_j} for record i.
5. Blinds the index by inserting $(UB_{\ell_{u_j}} - v_{\ell_{u_{i,j}}}).r$ number of 1's uniformly at random in $BF_{i,\ell_{u_{i,j}}}$ (possibly with replacement); This is equivalent to adding $UB_{\ell_{u_j}} - v_{\ell_{u_{i,j}}}$ random elements into the level-$\ell_{u_{i,j}}$ index, but without any pseudo-random function computations.
6. Outputs $I_i = (id_{i,\ell_{u_{i,j}}}, BF_{i,\ell_{u_{i,j}}})$, where $i = 1, \ldots, n$ and $0 \leq \ell_{u_{i,j}} \leq (h-1)$ as the index for record i. Then, I_i is given to the server.

Trap (K_{priv}, $R^* = [a, b]$): Given the master key $K_{priv} = (k_1, \ldots, k_r) \in \{0,1\}^{r\lambda}$ and a range $R^* = [a, b]$, the proxy computes, Tr_{R^*} for R^* as $(((x_1^*)_1 = f_{k_1}(R_1^*)), \ell_1^*), \ldots, (((x_1^*)_s = f_{k_1}(R_s^*)), \ell_s^*), \ldots, (((x_r^*)_1 = f_{k_r}(R_1^*)), \ell_1^*), \ldots, (((x_r^*)_s = f_{k_r}(R_s^*)), \ell_s^*)$, where $R_1^* \cup \ldots \cup R_s^* = R^*$, and $(h-1) \geq \ell_1^* \geq \ldots \geq \ell_s^* \geq 0$. Then, Tr_{R^*} is given to the server.

RanSInd(Tr_{R^*}, I_i): The server sets $sum = 0$, then it takes $Tr_{R^*} = ((x_1^*)_1, \ell_1^*), \ldots, ((x_1^*)_s, \ell_s^*), \ldots, ((x_r^*)_1, \ell_1^*), \ldots, ((x_r^*)_s, \ell_s^*)$, and index $I_i = (id_{i,\ell_{u_{i,1}}}, BF_{i,\ell_{u_{i,1}}})$ for range R_i, then it

7. Finds the corresponding Bloom Filter for record i that it holds $\ell_1^* = \ell_{u_{i,s'}}$, where $s' = [1, s]$. Assuming that $\ell_1^* = \ell_{u_{i,1}}$.
8. Computes the code-word $\left((y_{i,1}^*)_1\right) = f_{(x_1^*)_1}(id_{i,\ell_{u_{i,1}}}), \ldots, (y_{i,r}^*)_1 = f_{(x_r^*)_1}(id_{i,\ell_{u_{i,1}}})$.
9. Tests if $BF_{i,\ell_{u_{i,1}}}$ contains 1's in all r locations denoted by $(y_{i,1}^*)_1, \ldots, (y_{i,r}^*)_1$.
10. If so, computes $sum \leftarrow sum + 2^{\ell_{u_{i,1}}}$ and checks if $sum \geq \phi$,
11. If the inequality holds, outputs 1; otherwise, uses other trapdoor values to follow steps- [7-11]. If all the positions do not contain 1's in the Bloom filter in step [9], derives level-$\ell_{u_{i,1}}$ partial PRFs $(x_1^*)_1, \ldots, (x_r^*)_1$ by using PRG G. This results two level-($\ell_{u_{i,1}} - 1$) partial PRFs e_q and f_q such that $G((x_q^*)_1) = e_q \| f_q$, where $q = 1, \ldots, r$ and follows steps- [7-11] until $sum \geq \phi$ and level-0 is obtained from all the partial PRFs in trapdoor. If $sum < \phi$, it outputs 0.

5.4 | False-positive analysis

The false-positive analysis (FPA) is going to be done in a similar way as that is in the study by Goh [37]. In the study by Goh [37], the FPA is based on only one keyword but in our scheme, we have a multiple keywords (range) that should be considered.

The FPA for a keyword is done as follows: Assuming that there are r hash functions, $\delta = \{\delta_1, \ldots, \delta_\upsilon\}$ υ items that needs to be stored in m bit array. The probability that a keyword is not in δ but it appears in the array that r-hashed positions are 1 in the array is $\Pr_{FPA} = (1 - (1 - 1/m)^{r\upsilon})^r \approx \left(1 - e^{-r\upsilon/m}\right)^r$. When $r = (\ln 2)m/\upsilon$, the false-positive rate is going to be $(1/2)^r$.

In our construction (enhanced scheme), we assume that there are h' Bloom filters. Bloom Filter $BF_{h'-1}$ has m bit positions and has distinct υ data items, $BF_{h'-2}$ has $2m$ bit positions and has distinct 2υ data items, and BF_0 has $m2^{h'-1}$ bit positions and has distinct $\upsilon 2^{h'-1}$ data items. Moreover, there are r hash functions. Then, the probability that a given range $R = [r_1, r_2]$, $|R| = 2^{h'-1}$ (assuming that $\phi = |R|$), is not an element of $\delta_{h'-1} = \{\delta_{(h'-1,1)}, \ldots, \delta_{(h'-1,\upsilon)}\}$ (level-$\left(h' - 1\right)$'s δ set) but r positions are set to 1, is $(1/2)^r$. Then, we further derive R by using PRG G, we have 2 levels—$(h' - 2)$ ranges $R_1 = [r_1, r_{11}]$ and $R_2 = [r_{12}, r_2]$. The probability that two given ranges R_1, R_2 ($R = R_1 \cup R_2$) are not elements of $\delta_{h'-2} = \{\delta_{(h'-2,1)}, \ldots, \delta_{(h'-2,2\upsilon)}\}$ (level-$\left(h' - 2\right)$'s δ set) but each r positions are set to 1 is $2(1/2)^r$. When we go further to level 0, then we have $2^{h'-1}$ elements. The probability that given $2^{h'-1}$ elements are not elements of $\delta_0 = \{\delta_{(0,1)}, \ldots, \delta_{(0,\upsilon)}\}$ (level-0's δ set) but each r positions are set to 1 is $2^{h'-1}(1/2)^r$. The total false positive is $(1/2)^r + 2(1/2)^r + \cdots + 2^{h'-2}(1/2)^r + 2^{h'-1}(1/2)^r = \left(2^{h'} - 1\right)/2^r$.

According to this result, we can have small false positive rate, when we have smaller range R to search.

5.5 | Eliminating false-positive results

We have described two schemes that introduce false-positive results to the users who are authorized to make range queries. To eliminate false-positive records from the given result by the server, the data owner makes another independent column (*Time*) for each record's time range that is encrypted by AES algorithm. During any search phase, the cloud sends not only the results (probably results consist of wrong records because of false positive) but also it sends corresponding time range row of the record in *Time* column. Then the proxy filters out the false-positive results by comparing with given the user's query range and given corresponding time range rows of the records.

5.6 | Example

We give an example to describe the enhanced scheme.

Example: Let [4, 5] be a range to be encrypted, and let [4, 7] be a trapdoor to be searched in enhanced scheme. Also, we assume that threshold $\phi = 2$.

Setup: The data owner picks a random PRG G: $\{0,1\}^n \rightarrow \{0,1\}^{2n}$, and r secret keys $K_{priv} = \{k_1, \ldots, k_r\} \in \{0,1\}^{rn}$.

RanBInd: The data owner

- Computes the partial PRF values for the range [4, 5] that is $\left((a_i)_1 = f_{k_i}(010), 1\right)$, where $i = 1, \ldots, r$, (here k_i value is used as the seed for tree T_i),
- Assigns id_j for bloom filter BF_j, where $j = 0, 1$,
- Uses tree structure using a_i (level-1) as the new seed of the tree and follows the path id_1 to get new PRF value which is $y_i = f_{(a_i)_1}(id_1)$, where $i = 1, \ldots, r$,
- Puts an 1 into the array BF_1 of location y_i, where $i = 1, \ldots, r$,
- Applies G using a_i as the seed (since the level of a_i is 1), gets two level 0 PRFs, assuming that there are a_i0 (the first half of $G(a_i)$) and a_i1 (the second half of $G(a_i)$),
- Follows previous steps to fill BF_0,
- Outputs $I_{[4,5]} = \left(id_j, BF_j\right)_{j=0,1}$.
- Sends $\left(K_{priv}, G, \phi\right)$ to the proxy and $\left(G, I_{[4,5]}, \phi\right)$ to the server.

Trap: The proxy uses URC algorithm to build a trapdoor for range [4, 7]. The proxy

- Computes the partial PRF values as $Tr_{[4,7]} = \left(\left((a_i')_1 = f_{k_i}(011), 1\right), \left((b_i')_1 = f_{k_i}(010), 1\right)\right)_{i=1,\ldots,r}$, and sends these values to the server. (Ranges randomly mixed to make the order hidden)

RanSInd: The server once it gets $Tr_{[4,7]}$ and $I_{[4,5]}$,

- Sets $sum = 0$, and takes the highest level of partial PRFs, a'_i, where $i = 1, \ldots, r$,
- Uses tree structure using a'_i (level-1) as a seed of the tree and follows the path id_1 to get new PRF value which is $y'_{i1} = f_{(a'_i)_1}(id_1)$, where $i = 1, \ldots, r$,
- Checks if 1s are seen in the locations of y'_{i1}s in the array BF_1, where $i = 1, \ldots, r$,
- This case it is false,
- Takes the another partial PRF $\left((b'_i)_1 = f_{k_i}(010), 1\right)$, and uses tree structure using b'_i (level-1) as a seed of the tree and follows the path id_1 to get new PRF value which is $y'_{i2} = f_{(b'_i)_1}(id_1)$, where $i = 1, \ldots, r$,
- Checks if 1s are seen in the locations of y'_{i2}s in the array BF_1, where $i = 1, \ldots, r$,
- this case it is true,
- Sets $sum = 2 \leftarrow 0 + 2^1$ and checks if $2 \geq \phi = 2$,(YES) Outputs 1.

Based on the given example above, the cloud needs to make two comparisons (worst case) for our scheme. If we use the trivial scheme (leaf-to-leaf node comparison) which is one leaf node (from trapdoor) to one leaf node (from database) comparison each time, there are going to be $4 \times 2 = 8$ (worst case) comparisons. Thus, the proposed solution decreases search time from eight comparisons to two comparisons for this example. This provides very efficient search mechanism.

In general, in the trivial scheme/case, if trapdoor $[a, b]$ that $k = b - a + 1$ and database $l = [c, d]$ that $l = d - c + 1$. The cloud needs to compare each element in trapdoor set to each

element in database set to find common elements. Thus, the cloud needs to make kl comparisons.

5.7 | Search time complexity

In this subsection, we would like to compare search time complexities of our enhanced scheme (keywords/ranges in database are encrypted as ranges) to improved scheme in the study by Zhang et al. [10] (keywords/ranges in database are encrypted as ranges) and trivial scheme (keywords/ranges in trapdoor and database are not encrypted as ranges). We want to compare our enhanced scheme to improved scheme in the study by Zhang et al. [10] since the keywords in database is encrypted as ranges. The cloud server also compares ranges in trapdoor set to ranges in database in the study by Zhang et al. [10]. This comparison is shown in Table 2. The cloud compares given trapdoor set elements to client's database set elements. In the table, *Trapdoor* is the trapdoor set column and *Database* is the database set column that these sets consist of ranges. *Search Time Complexity (Number of Comparisons)* column is the number of comparisons between trapdoor set and database set that the cloud server makes. There is an example for the cloud's search time given in subsection 5.6.

In the table, TR_0 is a trapdoor set that consists of only a single leaf node (level-0) while $TR_{0,1}$ is a trapdoor set that consists of one leaf node and one level-1 node. $TR_{0,1,2}$ is a trapdoor set that consists of one leaf node, one level-1 node and one level-2 node while $TR_{0,1,2,\ldots,k}$ is a trapdoor set that consists of one leaf node, one level-1 node, one level-2 node, …, and one level-k node.

At the same time, in the table, DB_0 is a database set that consists of only a single leaf node (level-0) while $DB_{0,1}$ is a database set that consists of one leaf node and one level-1 node. $DB_{0,1,2}$ is a database set that consists of one leaf node, one level-1 node and one level-2 node while $DB_{0,1,2,\ldots,k}$ is a database set that consists of one leaf node, one level-1 node, one level-2 node, …, and one level-k node.

From the table, we can conclude that the search time of our enhanced scheme is better than the improved scheme in the study by Zhang et al. [10]. The search cost of our scheme is half of the search cost of the improved scheme in the study by Zhang et al. [10]. In the trivial case, the cloud server has a quadratic comparison/search time while in the enhanced scheme and improved scheme ([10]) the cloud has linear comparison/search time.

Our enhanced scheme is more efficient than the improved scheme in the study by Zhang et al. [10] in terms of cloud search time since in the study by Zhang et al. [10] the cloud needs to derive not only trapdoor set partial PRFs but also derive database set PRFs to find common elements. As an example, assuming that a trapdoor set consists of nodes A and four and a database set consists of nodes B and 3, where A and B nodes are level-1 nodes in Figure 2. In the study by Zhang et al. [10], the cloud compares each node in trapdoor set to each node in database as follows:

TABLE 2 Cloud search timSe complexity comparisons between Trivial scheme [24,26] and Improved Scheme [10] and Enhanced scheme

	Trapdoor	Database	Search time complexity (number of comparisons)
Trivial scheme [24,26]	TR_0	DB_0	1
Improved scheme [10]	TR_0	DB_0	1
Our enhanced scheme	TR_0	DB_0	1
Trivial scheme [24,26]	$TR_{0,1}$	$DB_{0,1}$	9
Improved scheme [10]	$TR_{0,1}$	$DB_{0,1}$	6
Our enhanced scheme	$TR_{0,1}$	$DB_{0,1}$	4
Trivial scheme [24,26]	$TR_{0,1,2}$	$DB_{0,1,2}$	49
Improved scheme [10]	$TR_{0,1,2}$	$DB_{0,1,2}$	19
Our enhanced scheme	$TR_{0,1,2}$	$DB_{0,1,2}$	11
Trivial scheme [24,26]	$TR_{0,1,2,3}$	$DB_{0,1,2,3}$	225
Improved scheme [10]	$TR_{0,1,2,3}$	$DB_{0,1,2,3}$	48
Our enhanced scheme	$TR_{0,1,2,3}$	$DB_{0,1,2,3}$	26
Trivial scheme [24,26]	$TR_{0,1,2,3,4}$	$DB_{0,1,2,3,4}$	961
Improved scheme [10]	$TR_{0,1,2,3,4}$	$DB_{0,1,2,3,4}$	109
Our enhanced scheme	$TR_{0,1,2,3,4}$	$DB_{0,1,2,3,4}$	57
Trivial scheme [24,26]	$TR_{0,1,2,\ldots,k}$	$DB_{0,1,2,\ldots,k}$	$\left(\sum_{i=0}^{k} 2^i\right)^2$
Improved scheme [10]	$TR_{0,1,2,\ldots,k}$	$DB_{0,1,2,\ldots,k}$	$2^0(k+1) + 2^1(2k) + 2^2(2k-2) + 2^3(2k-4) + \cdots + 2^k(2)$
Our enhanced scheme	$TR_{0,1,2,\ldots,k}$	$DB_{0,1,2,\ldots,k}$	$2^0(k+1) + 2^1(k) + 2^2(k-1) + 2^3(k-2) + \cdots + 2^k$

1. The cloud compares node A in trapdoor set to B in database set (level-1 nodes),
2. The cloud compares node 4 in trapdoor set to 3 in database set (level-0 nodes),
3. The cloud compares node 2 in trapdoor set (after derivation) to 3 in database set (level-0 nodes),
4. The cloud compares node 3 in trapdoor set (after derivation) to 3 in database set (level-0 nodes),
5. The cloud compares node 4 in trapdoor set to 5 in database set (after derivation) (level-0 nodes),
6. The cloud compares node 4 in trapdoor set to 4 in database set (after derivation) (level-0 nodes).

If the cloud server uses our enhanced scheme there are going to be 4 comparisons as follows:

1. The cloud checks if node A in trapdoor set is in level-1 Bloom Filter database set,
2. The cloud checks if node 4 in trapdoor set is in level-0 Bloom Filter database set,
3. The cloud checks if node 2 (after derivation) in trapdoor set is in level-0 Bloom Filter database set,
4. The cloud checks if node 3 (after derivation) in trapdoor set is in level-0 Bloom Filter database set.

The cloud makes 9 comparisons using trivial scheme, 6 comparisons using improved scheme in the study by Zhang et al. [10], and 4 comparisons using our enhanced scheme.

5.8 | Security analysis

In this section, we give the security analysis of both of the scheme. First, we show that our schemes have data privacy (IND-CKA) under the security of the pseudorandom generator of the underlying GGM construction. Second, we show that our schemes have trapdoor privacy.

Theorem 1 *Our both schemes satisfy data privacy (IND-CKA) if G is a length doubling PRG and f is a pseudo random function* that is *constructed from G.*

Proof We first prove the theorem for the basic scheme. We prove the theorem using its contrapositive. Suppose the basic scheme is not an IND-CKA. Then, there exists an algorithm A that breaks the basic. We build an algorithm B that uses A to determine if strings are generated from a pseudo-random generator G or from a totally random function.

Before showing this, we first show that A breaks the basic scheme. We build an algorithm $B^{'}$ that uses A to determine if f is a PRF or a totally random function. Second, we show that if $B^{'}$ is able to distinguish PRF from a totally random function then B distinguished pseudo-random generator (PRG) from a totally random function. The second part is immediate and straightforward from GGM construction [38].

$B^{'}$ has access to an oracle $\circ_f$ for the unknown f that takes as input $x \in \{0,1\}^n$ and returns $f_k(x) \in \{0,1\}^n$. This oracle is maintained by B. When running any of the four algorithms, $B^{'}$ substitutes evaluations of f with queries to the oracle $\circ_f$. Algorithm $B^{'}$ simulates A as follows.

Setup: Algorithm $B^{'}$ picks a set DR from $\{0,1\}^n$ uniformly at random and sends DR to A. In response, A returns a polynomial number of subsets of DR. We call this collection of subsets DR^*. We assume that the size of each subset $r = |D|$. For each subset $D \in DR^*$, $B^{'}$ assigns a unique id D_{id}, and runs RanBInd on D with identifier D_{id} to obtain index $I_{D_{id}}$. After creating all the indexes, $B^{'}$ gives all indexes and their associated subsets to A.

Queries: On receiving A's query for the trapdoor of $|x| = 1$ (single point), $B^{'}$ runs Trap on x and replies with trapdoor T_x. When the trapdoor is a range $|TR| = d$ $(d > 1)$, $B^{'}$ runs Trap on TR and replies with trapdoor T_{TR} $(|T_{TR}| = d)$.

Challenge: After making some trapdoor queries, decides on a challenge by picking a subset $DR_0 \in DR^*$, and generating another subset DR_1 from DR such that $|DR_0 - DR_1| \neq 0$, $|DR_1 - DR_0| \neq 0$, and the total length of words in DR_0 is equal to that in DR_1. Furthermore, A must not have queried $B^{'}$ for the trapdoor of any word in $DR_0 \Delta DR_1$. Next, A gives DR_0 and DR_1 to $B^{'}$ who chooses $b \in \{0, 1\}$ and also chooses a new (unique) identifier DR_{id}. $B^{'}$ runs RanBInd on DR_b with identifier DR_{id} to get index I_{DR_b}, which is given to A. After the challenge is issued, A is not allowed to query $B^{'}$ for the trapdoors of any single point $x \in DR_0 \Delta DR_1$.

Response: A eventually outputs a bit b', representing its guess for (b) If $b' = b$, then $B^{'}$ outputs 0, indicating that it guesses that f is a PRF. Otherwise, $B^{'}$ outputs 1. When f is a PRF, then.

$* = \Pr B^{'}(f) = 0 | k \leftarrow \{0,1\}^n] - 1/2| \geq \epsilon$

When f is a random function, then.

$** = \Pr[B^{'}(r) = 0 | r \leftarrow \{0,1\}^n] = 1/2$ *** It follows from these two that $*-**\geq\epsilon$. From [38] (Theorem 3—Main theorem) $\Pr B(G(k)) = 0 | k \leftarrow \{0,1\}^n] - \Pr B(r) = 0 | r \leftarrow \{0,1\}^{2n}] \geq \epsilon^{'}$ thus proving the theorem.

The proof of data privacy for enhanced scheme is a bit challenging than the basic scheme. This is because

- there are $l-$index to describe a range that is to be encrypted. This is because the range is represented by nodes consist of not only the leaves but also consist of internal nodes in the GGM tree. For this reason, each j level node is mapped to the corresponding bloom filter, level-j, that stores the level j internal nodes, where $0 \leq j \leq l$ and l is the height of the tree.
- the adversary obtains not only leaf PRFs but also can obtain also partial PRF values in the GGM tree (via trapdoor queries).

Because of these two important information above, we need to introduce new restrictions for the adversary in the challenge phase.

Setup: Algorithm $B^{'}$ picks a set DR from $\{0,1\}^n$ uniformly at random and sends DR to A. In response, A returns a polynomial number of subsets of DR. We call this collection of subsets DR^*. We assume that the size of each subset $r = |D|$. For each subset $D \in DR^*$, $B^{'}$ assigns a unique id D_{id}, and runs RanBInd on D with identifier D_{id} to obtain the index $I_{D_{id}}$. In this step, $B^{'}$ uses URC algorithm in the study by Kiayias et al. [39]. Then, it creates a BF_j for node with level-j, where $0 \leq j \leq l$. After creating all the indexes, $B^{'}$ gives all indexes and their associated subsets to A.

Queries: On receiving A's query for the trapdoor of $|x| = 1$ (single point), $B^{'}$ runs Trap on x and replies with trapdoor T_x. When the trapdoor range is $|TR| > d$ $(d > 1)$, $B^{'}$ runs Trap on TR and replies with trapdoor T_{TR} ($|T_{TR}| = O(logd)$) using URC construction in [39].

Challenge: After making some trapdoor queries, A decides on a challenge by picking a subset $DR_0 \in DR^*$, and generating another subset DR_1 from DR^* such that $|DR_0 - DR_1| \neq 0$, $|DR_1 - DR_0| \neq 0$, and the total length of words in DR_0 is equal to that in DR_1. Furthermore, A can query a range TR' ($|T_{TR'}| = 1$ or $|T_{TR'}| = d$). Next, A gives DR_0 and DR_1 to $B^{'}$ who chooses $b \in \{0, 1\}$ and also chooses a new (unique) identifier DR_{id}. $B^{'}$ runs RanBInd on DR_b with identifier DR_{id} to get index I_{DR_b}, which is given to A. After the challenge is issued, A

1. Is not allowed to query B' for the trapdoors of any single point $x \in DR_0 \Delta DR_1$.
2. Is not allowed to query B' for the trapdoors of a range $T_{TR'} = \left(f_{k_i}(TR_{\ell_1}), \ell_1\right), \left(f_{k_i}(TR_{\ell_2}), \ell_2\right), \ldots, \left(f_{k_i}(TR_{\ell_l}), \ell_l\right)$, where $\ell_1 > \ldots > \ell_l$ and $TR_{\ell_1} \cup \ldots \cup TR_{\ell_l} = TR'$ if TR' satisfies.
   A. $(T_{TR'}) \cap (T_{DR_0}) = \emptyset$ but $(T_{TR'}) \cap (T_{DR_1}) \neq \emptyset$.
   B. $f_{f_{k_i}\left(TR_{\ell_j}\right)}\left(id_{\ell_j}\right)$ positions are 1 in BF_{DR_0,ℓ_j} but not in BF_{DR_1,ℓ_j}, where $i = 1, \ldots, r, j = 1, \ldots, l$.
   C. $f_{G_1\left(f_{k_i}\left(TR_{\ell_j}\right)\right)}\left(id_{\ell_j-1}\right)$ positions are 1 in BF_{DR_0,ℓ_j-1} but not in BF_{DR_1,ℓ_j-1}, where $i = 1, \ldots, r, j = 1, \ldots, l$.
   D. $f_{G_0\left(f_{k_i}\left(TR_{\ell_j}\right)\right)}\left(id_{\ell_j-1}\right)$ positions are 1 in BF_{DR_0,ℓ_j-1} but not in BF_{DR_1,ℓ_j-1}, where $i = 1, \ldots, r, j = 1, \ldots, l$. [C] and [D] are going to be applied until each partial PRF's level (the level of TR_{ℓ_j} is ℓ_j) is 0 (leaf level), where $j = 1, \ldots, l$.

Response: A eventually outputs a bit b', representing its guess for (b) If $b' = b$, then $B^{'}$ outputs 0, indicating that it guesses that f is a PRF and B concludes that strings are generated by G that is a PRG. Otherwise, $B^{'}$ outputs 1.

The first restriction is immediate since any trapdoor query of any single point $x \in DR_0 \Delta DR_1$ is going to help the adversary to distinguish the index. The second restriction says that the adversary can be ask trapdoor queries only if the trapdoor range TR' is common or not common in both DR_0 and DR_1.

For both of the schemes, A wins the game if $b = b'$, then B concludes that G is a PRG that the sequence of bits generated with G by distinguishing two consecutive games in GGM proof [38] since A can be able to distinguish underlying function f is a PRF. We finally show that B can determine if it is having strings that are generated from a PRG G or totally random strings with advantage at least ϵ'. This can be shown in the same way above by following $*$, $**$ and $***$.

Remark 2. *In the challenge phase, A's choices for data ranges DR_0 and DR_1 should also satisfy [1] and [2] (restrictions above) with trapdoor range query TR in Queries step. Otherwise, A wins the game with probability 1.*

5.8.1 | Query privacy

Theorem 2 *Basic and enhanced schemes have trapdoor privacy if G is a PRG.*

Proof Recall that query privacy means two different but the same size range queries are not distinguishable by the adversary. Consider two range queries A_{TR_0}, A_{TR_1} which are given in the challenge phase.

Query privacy for enhanced scheme can be attained using DPRF policy privacy security proof [39] (Theorem 4) which relies on the security of the PRG of the underlying GGM construction. Since the basic construction is a special construction of the improved scheme, query privacy can be proved similarly. The only difference in our proof from the study by Kiayias et al. [39] is that we allow the adversary to make range encryption queries but this does not make the proof different since any range encryption query can be seen as a trapdoor range query. Therefore, the restrictions in the study by Kiayias et al. [39] still apply. The restrictions are $A_{TR_0} \neq A_{TR_1}, |A_{TR_0}| = |A_{TR_1}|, A_{TR_j} \cap (A_{TR_0} \cup A_{TR_1}) = \emptyset$ and. $A_{DR_j} \cap (A_{TR_0} \cup A_{TR_1}) = \emptyset$where, TR_j is trapdoor query j, DR_j is encryption query j, $j = 1, \ldots, q$ and A_{TR_j} or A_{DR_j} is the all leaf values (nodes) that T_{TR_j} or T_{DR_j} is able to have after sequentially applying PRG G.

6 | CONCLUSION

We have introduced two symmetric range encryption schemes: Basic and Enhanced. Our constructions have trapdoor privacy and data privacy. The enhanced scheme is more efficient than the basic scheme. Our enhanced scheme is efficient in terms of communication between the cloud server and the user. Our enhanced scheme is more efficient than the trivial scheme since it allows the cloud server to make range to range comparisons. In other words, the cloud is able to compare multi-keywords in given trapdoor to multi-keywords in given database. Furthermore, our enhanced scheme is search efficient than the improved scheme in the study by Zhang et al. [10], where it also allows the cloud server to make range to range

comparisons. As a future work, we would like to implement our schemes (basic and enhanced) to see their computational and communication costs.

REFERENCES

1. Song, C., Qu, Z., Blumm, N., Barabási, A.L.: Limits of predictability in human mobility. Sci. 327(5968), 1018–1021 (2010)
2. de Montjoye, Y.A., et al.: Unique in the crowd: the privacy bounds of human mobility. Proc. of Nature Sci. Rep. 3, 1–5 (2013)
3. Kumar, U., Yadav, N., & Helmy, A.: Gender based grouping of mobile student societies. In Proc. of MODUS, IPSN workshop (2008)
4. Hsu, W., et al.: Tvc: Modeling spatial and temporal dependencies of user mobility in wireless mobile networks. Proc. of IEEE/ACM Trans. Netw. 17, 1564–1577 (2009)
5. Hsu, W., Dutta, D., Helmy, A.: Extended abstarct: Mining behavioral groups in large wireless lans. In Proc. of ACM MobiCom (2007)
6. Balaji, B., et al.: Sentinel: Occupancy based HVAC actuation using existing WiFi infrastructure within commerical buildings. In Proc. of 11th ACM Conference on Embedded Networked Sensor Systems (2013)
7. Farhan, A.A., Bamis, A., Wang, B.: Locating emergencies in a campus using wi-fi access point association data. In: Proc. Of ACM Workshop on Pervasive Urban Crowdsensing architecture and applications (PUCAA) (2013)
8. Kumar, U., Helmy, A.: Human behavior and challenges of anonymizing wlan traces. In Proc. of GLOBECOM, pp. 1–6 (2009)
9. Tan, K., et al.: Privacy analysis of user association logs in a large-scale wireless lan. In Proc. of INFOCOM, pp. 31–35 (2011)
10. Zhang, L., et al.: Encrypting wireless network traces to protect user privacy: a case study for smart campus. In 2th IEEE International Conference on wireless and mobile computing, Networking and Communications, WiMob 2016, New York, NY, USA, October 17-19, 2016, pp. 1–8 (2016)
11. Chi, J., et al.: Privacy-enhancing range query processing over encrypted cloud databases. In Web Information Systems Engineering - WISE 2015 - 16th International Conference, Miami, FL, USA, November 1-3, 2015, Proceedings, Part II, pp. 63–77 (2015)
12. Demertzis, I., et al.: Practical private range search revisited. In Proceedings of the 2016 International Conference on Management of data, SIGMOD '16, New York, NY, USA, pp. 185–198 (2016). ACM
13. Li, R., et al.: Fast range query processing with strong privacy protection for cloud computing. Proc. VLDB Endow. 7(14), 1953–1964 (2014)
14. Lu, Y.: Privacy-preserving logarithmic-time search on encrypted data in cloud. In 19th Annual network and distributed system security Symposium, NDSS 2012, San Diego, California, USA, February 5-8, 2012 (2012)
15. Pappas, V., et al.: Blind seer: a scalable private DBMS. In 2014 IEEE Symposium on Security and Privacy, SP 2014, Berkeley, CA, USA, May 18-21, 2014, pp. 359–374 (2014)
16. Zuo, C., et al.: Dynamic searchable symmetric encryption with forward and stronger backward privacy. In: Sako, K., Schneider, S., Ryan, P. Y. A. (eds.) Computer security - ESORICS 2019 - 24th European Symposium on Research in computer security, Luxembourg, September 23-27, 2019, Proceedings, Part II, volume 11736 of Lecture Notes in Computer Science, pp. 283–303. Springer (2019)
17. Zuo, C., et al.: Dynamic searchable symmetric encryption schemes supporting range queries with forward (and backward) security. In: Lopez, J., Zhou, J., Soriano, M. (eds.) Computer security, pp. 228–246. Springer International Publishing, Cham (2018)
18. Hacigumus, H., et al.: Executing sql over encrypted data in the database service provider model. Proc. of SIGMOD, pp. 216–227 (2002)
19. Hore, B., Mehrotra, S., Tsudik, G.: A privacy-preserving index for range queries. Proc. of VLDB (2004)
20. Popa, R.A., et al.: Protecting confidentiality with encrypted query processing. Proc. of SOSP (2011)
21. Agrawal, R., et al.: Order preserving encryption for numeric data. In Proc. of ACM SIGMOD (2004)
22. Boldyreva, A., Chenette, N., & Lee, Y.: Order preserving symmetric encryption. In Proc. of EUROCRYPT (2009)
23. Popa, R.A., Li, F.H., Zeldovich, N.: An ideal-security protocol for order-preserving encoding. Proc. of Security and Prvacy (2013)
24. Boneh, D., et al.: Public key encryption with keyword search. In Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings, pp. 506–522 (2004)
25. Curtmola, R., et al.: Searchable symmetric encryption: improved definitions and efficient constructions. In Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006, Alexandria, VA, USA, Ioctober 30 - November 3, 2006, pp. 79–88 (2006)
26. Song, D. X., Wagner, D., & Perrig, A.: Practical techniques for searches on encrypted data. In 2000 IEEE Symposium on Security and Privacy, Berkeley, California, USA, May 14-17, 2000, pp. 44–55 (2000)
27. Demertzis, I., et al.: Practical private range search in depth. ACM Trans. Database Syst. 43(1):2:1–2:52 (2018)
28. Yang, Z., Zhong, S., Wright, R. N.: Privacy-preserving queries on encrypted data. In Proceedings of the 11th European Conference on Research in computer security, ESORICS'06 (2006)
29. Li, R., et al.: Fast and scalable range query processing with strong privacy protection for cloud computing. IEEE/ACM Trans. Netw, 24(4): 2305–2318 (2016)
30. Boldyreva, A., Chenette, N., O'Neill, A.: Order preserving symmetric encryption revisited: improved security analysis and alternative solutions. In Proc. of CRYPTO (2011)
31. Roche, D.S., et al.: Pope: Partial order preserving encoding. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, CCS '16, New York, NY, USA, pp. 1131–1142 (2016). ACM
32. Kerschbaum, F. & Tueno, F.: An efficiently searchable encrypted data structure for range queries. CoRR, abs/1709.09314 (2017)
33. Miao, Y., et al.: Multi-authority attribute-based keyword search over encrypted cloud data. IEEE Trans Dependable Secure Comput, pp. 1 (2019)
34. Yang, Y., Liu, X., Deng, R.H.: Expressive query over outsourced encrypted data. Inf Sci. 442-443, 33–53 (2018)
35. Qin, B., et al.: Public-key authenticated encryption with keyword search revisited: security model and constructions. Inf. Sci. 516, 515–528 (2020)
36. Peng, Y., et al.: Ls-rq: A lightweight and forward-secure range query on geographically encrypted data. IEEE Trans Dependable Secure Comput, pp. 1 (2020)
37. Goh, E.-J.: Secure indexes. In submission (2004)
38. Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions. J ACM. 33, 792–807 (1986)
39. Kiayias, A., et al.: Delegatable pseudorandom functions and applications. In Proc. of CCS (2013)

How to cite this article: Oksuz O. Time-specific encrypted range query with minimum leakage disclosure. *IET Inf. Secur.* 2021;15:117–130. https://doi.org/10.1049/ise2.12010