CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. 0000; **00**:1–20
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/cpe

Survey on Secure Search Over Encrypted Data on the Cloud

Hoang Pham[1]*, Jason Woodworth[1], and Mohsen Amini Salehi[1]

[1] *High-Performing Cloud Computing (HPCC) Laboratory, School of Computing and Informatics, University of Louisiana at Lafayette, LA, USA*

SUMMARY

Cloud computing has become a potential resource for businesses and individuals to outsource their data to remote but highly accessible servers. However, the potential of cloud services has not been fully realized due to users concerns about data privacy and security. User-side encryption techniques can be employed to mitigate the security concerns, but once the data is encrypted, no processing (e.g., searching) can be performed on the outsourced data. Searchable Encryption (SE) techniques have been widely studied to enable searching on the data while they are encrypted. These techniques enable various types of search on the encrypted data and offer different levels of security. While these techniques enable different search types and vary in details, they share similarities in their components and architectures. In this paper, we provide a comprehensive survey on different secure search techniques, a high-level architecture for these systems, and an analysis of their performance and security level. Copyright © 0000 John Wiley & Sons, Ltd.

Received . . .

KEY WORDS: Survey, Search over Encrypted Data, Cloud Security, Encrypted Search.

1. INTRODUCTION

As cloud computing becomes prevalent, more cloud-based solutions are being developed and widely used in different applications. Companies that have adopted cloud storage solutions are reported to gain a competitive edge against those that have not [1].

Cloud computing is favored due to its many advantages, including: convenience and accessibility, consistent back ups to reducing the burden of local storage, and saving capital expenditure on in-house hardware and software maintenance [2]. However, public cloud storage services may be utilized by multi-tenant customers who store large amounts of potentially sensitive data on the cloud. Using cloud storage implies losing full control over data and delegating it to the cloud administrators, exposing the data to potential external and internal attacks [3, 4], which can be devastating for organizations that rely on confidentiality of their data (e.g financial corporations).

These problems have made businesses concerned about outsourcing their data to the cloud and utilizing its potential [5, 6]. For instance, a medical center that owns patients' health records cannot outsource its data to a cloud that is vulnerable to attacks, due to legal regulations [7]. In another instance, a law enforcement agency that keeps sensitive criminal records will hesitate to use cloud storage because of similar concerns.

One way to overcome the confidentiality problem is to encrypt data on local hardware before outsourcing it to the cloud. While this approach preserves data confidentiality, it hinders data

*Correspondence to: High-Performing Cloud Computing (HPCC) Laboratory
School of Computing and Informatics
University of Louisiana at Lafayette, LA, USA

processing. In particular, searching is of paramount importance for outsourced unstructured data [8]. In fact, when data is encrypted, search systems do not function anymore, because they are unable to compare the query to the encrypted data.

A naïve approach to enable search on encrypted data would be downloading all of the data from the cloud, decrypting them, and locally performing plain text search [9]. However, with potentially huge data (also called big data) hosted on the cloud [10] and limited network bandwidth [11], this approach remains impractical. Therefore, searchable encryption systems (*e.g.,* [12, 13, 14, 15]) have been introduced to cope with this problem. These systems ideally allow the encrypted data to be searched without revealing the data and search query. Hence, they relieve concerns about data confidentiality in the cloud.

Efforts to create searchable encryption systems date back to 2000 with work by Song *et al.* [11]. Since then, numerous research works have been undertaken to enable different types of searchable encryption. Although these systems differ in their search approaches, security level, and performance, they share certain architectural similarities. There are other survey studies over different searchable encryption systems [16, 17]. This paper complements those survey studies by providing a comprehensive survey of the existing searchable encryption systems and differentiates them in terms of their search approaches, security level, and performance. In addition, we provide a generic framework that encompasses the general components of searchable encryption systems. As more granular search systems are recently being demanded by different industries, we survey domain-specific search systems as well. Finally, we identify the shortcomings of the current research works and recommend avenues for future research and development efforts in the searchable encryption area. In summary, the contributions of this paper are as follows:

- Analyzing the commonalities and differences in various types of cloud-based searchable encryption systems.
- Providing a generic framework for cloud-based searchable encryption systems.
- Surveying and categorizing current cloud-based searchable encryption systems.
- Surveying domain-specific encryption systems.
- Identifying forthcoming challenges in searchable encryption and introducing future research avenues to address them.

The remainder of the paper is organized as follow: Section 2 introduces the background and preliminaries of searchable encryption. An overall framework for searchable encryption is given in Section 3. Section 4 reviews the security requirements and criteria to assess searchable encryption systems. Section 5 reviews the categorization of different searchable encryption schemes and Section 6 evaluates the security shortcomings of the searchable encryption system and also categorizes the existing searchable encryption systems based on their security analysis. Next, Section 7 surveys domain-specific searchable encryption schemes. Finally, Section 9 summarizes the paper and provides avenues for future research in this area.

2. ELEMENTS OF A CLOUD-BASED SEARCHABLE ENCRYPTION SYSTEM

Song *et al.* [11] are one of the pioneers of searchable encryption. They provide a system in which a client (*i.e.,* data owner) can search over her encrypted data (in the form of emails) on an email server. Once the data owner wants to search some keywords in her emails, she submits an encrypted query (termed trapdoor) to the server. The server is in charge of searching over the encrypted data and retrieving related emails for the owners. More recent research works in searchable encryption (*e.g.,* [14, 18, 19, 20, 13, 21]) describe their systems with similar elements to this work. We introduce these elements in greater detail in the rest of this section.

As depicted in Figure 1, which is inspired by works in [14, 18], searchable encryption systems are commonly composed of three main elements, as follows:

Data owner. A data owner sets up the system by granting access to data users and uploading documents to the cloud. The data owner possesses a collection of n documents $D =$

DOI: 10.1002/cpe

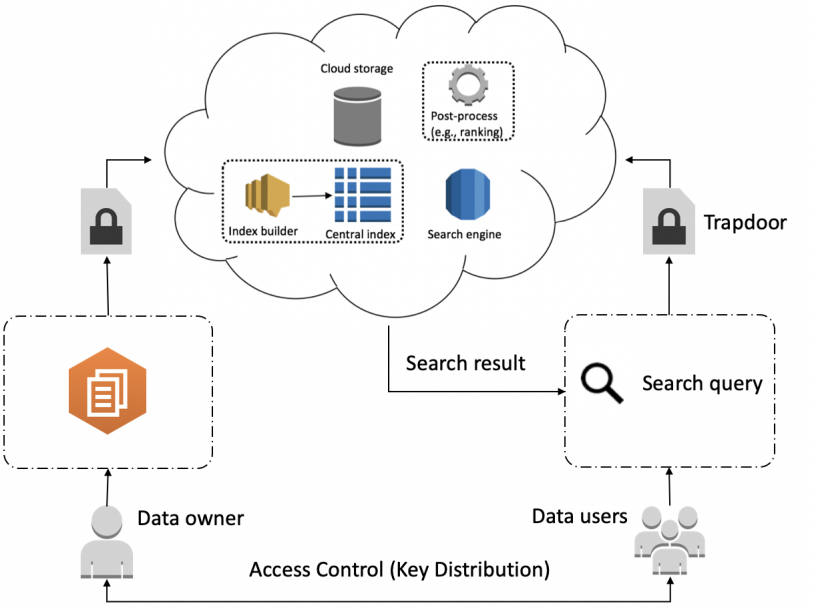

Figure 1. Main elements involved in cloud-based searchable encryption systems.

$\{d_1, d_2, d_3, \ldots, d_n\}$ and wants to outsource them to a remote public cloud server (*e.g.,* Amazon and IBM Cloud Storage) for storage or sharing purposes. In order to protect the confidentiality of the documents, the owner locally encrypts the data using her authorization key and uploads the encrypted data to the cloud.

Data user. The data users are those who are authorized to search and retrieve the uploaded documents. The data users have an encryption key that is applied to their search queries to create trapdoors[†]. The trapdoors are then sent to the cloud servers to retrieve search results in the form of a list of relevant documents (*e.g.,* in [22, 23]) or document identifiers (*e.g.,* in [24, 25]). The data user can potentially be permitted to decrypt the document locally [24, 25]. It is worth noting that, in practice, a data owner can be a data user too.

Cloud server. The cloud server receives an encrypted collection of documents uploaded by data owners and carries out three main tasks: storing uploaded documents, searching them against trapdoors, and maintaining and updating relevant data structures.

The research works that have been undertaken in cloud-based searchable encryption generally assume cloud servers to be *honest-but-curious*. That is, although the cloud server administrator follows necessary security procedures and does not modify nor delete data files, she is still "curious" about the content of the documents.

Another component we found common in current searchable encryption systems is a trusted computing base. Each searchable encryption system implements this element in its own way to adapt its purposes and functionality. Although a trusted computing base can be combined with other components in the searchable encryption system system, it is worth recognizing and mentioning that in the paper.

[†]For further explanation of trapdoors, please see Section 3

DOI: 10.1002/cpe

Trusted computing base (Gateway) Data owners and users in a searchable encryption system need data to be prepared and preprocessed (*e.g.,* removing stop words from the search query or extracting keywords from documents [21]) on their local hardware before proceeding to the Cloud server. Preprocessing constructs another element in the searchable encryption system, termed Trusted computing base (or Gateway) [14, 20, 12], that includes the client-side application. In particular, the job of the gateway is to prepare documents for the data owner in the setup phase and preprocess data users' queries in the retrieval phase. The gateway is generally assumed to be trusted and resides within the user's premises. There are two approaches to implement the gateway: *client-end approach* in which the gateway is part of client machine [26, 27]; and *trusted server approach* in which the gateway resides on separate trusted sever [14, 20, 12].

- **Client-end approach** The advantage of the client-end approach is that the data is secure at its origin, and thus is safe against connection interception. However, this imposes overhead on the client-side application, potentially affecting its performance. Hence, it is considered inappropriate for circumstances where data users predominantly use thin-client devices (*e.g.,* smart phones) [28].
- **Trusted server approach** The Trusted server approach is generally faster and lends itself better to the edge computing platforms. Although this approach imposes a low overhead on the client machines, it reveals data in the communication channels between the client machine and the trusted server. This approach also includes server provisioning and maintenance costs. In practice, the trusted server approach is appropriate for circumstances where clients' devices fall short in computing power and have limited energy supply [19].

3. A GENERAL ARCHITECTURE FOR SEARCHABLE ENCRYPTION SYSTEMS

3.1. Overview

The architecture of a searchable encryption system has to implement four processes involved with the elements mentioned in the previous section. In this section, we first describe these four processes, then we discuss how they are applied within different elements of the searchable encryption system.

3.2. Processes Involved in a Searchable Encryption System

The processes in a searchable encryption system are namely *Key generation (Keygen), Build-Index, Trapdoor generation,* and *Search*. These processes enable searchable encryption systems and generally have polynomial time complexity [12, 29].

Key Generation (Keygen) Process. This process creates a key to encrypt plain text documents, and later decrypts the retrieved documents. The algorithm for this process generates a key based on a set of given security parameters. Probabilistic key generation algorithms [29] are commonly used for this process.

As data owners and users transmit and store data through the server, there is a need to securely store documents and determine if a user is authorized to access them. There are two common methods for encrypting documents: symmetric and asymmetric encryption.

- Symmetric Encryption: In this method, both data owner and data user share the same secret key. This key can be used to both encrypt and decrypt the document. In other words, this key is to create noise in the documents, making them unreadable to unauthorized users, while authorized users can use that shared key (or a computable "inversed" key) to defuse and remove the noise in the document [30].
- Asymmetric Encryption: Known as Public Key Encryption. This cryptography includes two different keys, public key and private key, which are used to encrypt and decrypt a document [31, 32]. More specifically, the data owner would use one of his keys for encryption the document and the other to decrypt. Since the two key are completely different and there

DOI: 10.1002/cpe

is no computational correlation between them, even if the encryption key (public key) is compromised, the attacker can not get the data content without the private key.

Build-Index. Searchable encryption systems commonly utilize an index structure to keep track of the occurrences of keywords in documents. The process of initializing this index, called *Build-Index*, takes key K from *Keygen process* the and a collection of documents D as inputs. Then, it extracts keywords from the documents and inserts them into the index structure.

This *Build-Index* process is used by the data owner to generate a secure and searchable structure that enables search over the encrypted data. An index structure is generally implemented in form of a hash table [33], meta data (markup) [18], or inverted index [28, 34] where each unique keyword is mapped to an identifier for each document it appears in.

Trapdoor Generation. This process is used by data users to form search queries. It encrypts the user's search query using a key that is compatible with the Build-Index key K. If needed, the search query is preprocessed (*e.g.,* expanded) by the Trapdoor Generation process [27, 21, 13]. Then, the encrypted Trapdoor is sent to the cloud server.

Search. After receiving the trapdoor, the server runs the search procedure to match documents that contain the set of keywords in the trapdoor. Next, the results are sent back to the client.

3.3. General Architecture for Searchable Encryption Systems

The general architecture for a cloud-based searchable encryption system is depicted in Figure 2.

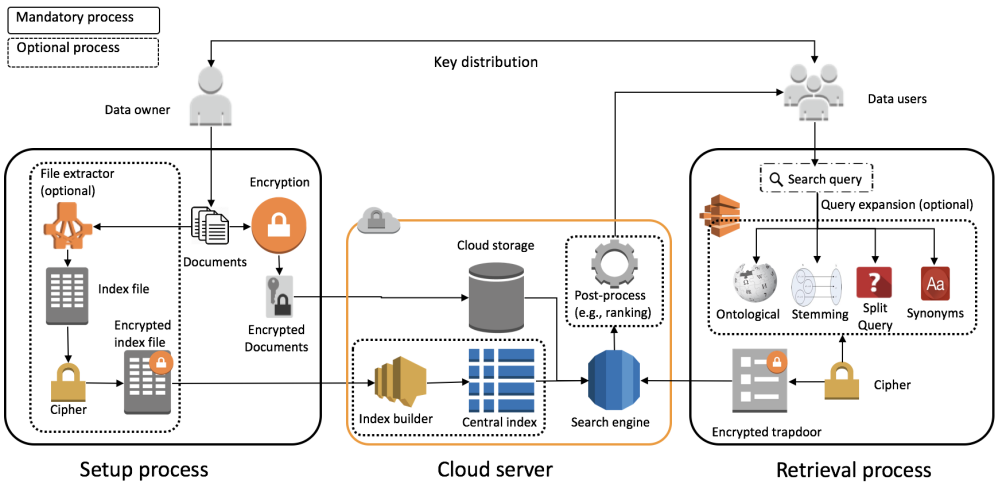

Figure 2. Architectural Overview of Cloud-based Searchable Encryption Systems.

The system architecture consists of two main mechanisms, namely *Setup* and *Retrieval*. The main job of the Setup mechanism is to prepare documents to be searched by data users. Upon receiving a search query from the data user, the job of the Retrieval mechanism, is to perform the search on the dataset, find matching documents, and send the results back to the data user. In the next subsections, we elaborate on how these mechanisms operate.

3.3.1. Setup Mechanism Before sending documents to the cloud, the Setup mechanism first extracts the useful information from the documents. The type of extracted data depends on the type of search

DOI: 10.1002/cpe

system (*e.g.,* keyword search [35, 36] versus semantic search [14, 15]). Then, the extracted data and documents are encrypted and sent to the cloud server.

The data owner initiates the search system through the *Keygen* process (see Section 3.2 for more details). The generated key is necessary to encrypt documents before outsourcing, and decrypt downloaded documents.

In some searchable encryption systems, the Setup mechanism also includes creating an "index structure" [14, 15, 21, 35]. The Index structure is also known as "meta data" [18] or "identify keywords" [31, 37]). The index is comprised of keywords that represent the essence of each uploaded document. Alternatively, some other searchable encryption systems do not rely on the index structure, instead directly encrypting each keyword individually to form an encrypted searchable document [11].

The searchable encryption system then encrypts the documents' contents as well as the index structure (if it exists), before sending them to the cloud server.

In systems where the data owner and data users are separate entities, the data owner needs to distribute the public-key to the data users. The keys are utilized by data users to create trapdoors that are compatible with the encrypted uploaded data. Methods such as public key cryptography [33] or broadcast encryption [36] are commonly used for key distribution.

3.3.2. Retrieval Mechanism After setup, the system is expected to have a collection of files ready to be searched over. Data users or owners can submit a search query, defined as a set of keywords $W = \{w_1, w_2, \ldots, w_n\}$.

The Trapdoor is produced using W and the data owner's keys. Some systems (*e.g.,* [22, 38]) also apply pre-processing of the search query when producing the Trapdoor. Once produced, it is sent to the cloud server.

The cloud server includes a search engine that carries out the search process. In systems that rely on an index structure, the index is used to match a Trapdoor against index entries to find relevant documents. At the end, the list of results, which includes matching documents or their identifiers, is sent back to the user. Upon receiving the result list, the data user can request to retrieve (*i.e.,* download) and decrypt the documents, if they are authorized.

It is worth mentioning that during the Retrieval mechanism, the cloud server could learn minimal information about the documents [36, 21]. In the next section, we provide further details on security aspects of cloud-based searchable encryption systems.

4. SECURITY REQUIREMENTS AND SHORTCOMINGS OF SEARCHABLE ENCRYPTION SYSTEMS

4.1. Security Criteria of searchable encryption Systems

It is crucial for searchable encryption systems to prove that they can preserve the confidentiality of the user's data and prevent information leakage. Therefore, the resistance of searchable encryption systems should be verified against possible internal or external attacks on an untrusted server. More specifically, the server should not be able to learn anything about the original data from the cipher text or the search process. Song *et al.* [11] defined three security properties that every searchable encryption system should maintain:

1. *Controlled searching*: Unauthorized users should not be able to search in the server. In [37, 39], an unauthorized user cannot submit a query to the server unless she has the secret key to generate a trapdoor. On the server side, the data must be kept in an encrypted manner. Also, processing the data on the server must be done without decrypting the data. Thus, without a trapdoor, nothing can be returned in response to an unauthorized search request.
2. *Hidden query*: this technique hides the unencrypted query from the untrusted server. Every searchable encryption system should be able to mask or encrypt the content of the search query to avoid the possibility of the server inferring the content from the search results. The

DOI: 10.1002/cpe

untrusted server can only learn about the relationship of a secure query to a set of document identifiers, not what that set of documents are about.
Without an encrypted query, an attacker could submit numerous queries to the searchable encryption system. Then, by analyzing the search results, documents' contents can be inferred [37]. In [37], the authors present a secure index (*e.g.,* Z-idx) that does not reveal the actual search query, by hashing it into an irreversible trapdoor, before sending it to the server. Hence, the server cannot learn any information from the query. Similarly, in [40], Ren *et al.* implement a secondary homomorphic encryption on top of the deterministically encrypted query terms to further obfuscate the original query.

3. *Query isolation*: In the search process, the server should know nothing except for the search results [11]. In searchable encryption systems (*e.g.,* [39]), if there is a match between the query and the index, the server can locate the related documents and return them to the data user. However, since the data on the server is encrypted and the secret key is not stored in the server, the server cannot understand anything other than the search result.

4.2. Shortcomings of Searchable Encryption Systems

The security requirement means that the system needs to adopt methods to protect the data. For example, documents and auxiliary indices are encrypted using the secret key so that only a user with the valid key can learn about the content, thus, its privacy and confidentiality from the untrusted server are protected. Even with cipher text, there are prevention measures against different hacking techniques, such as statistical attacks [41] and keyword-distribution attacks [41], that aim to derive the correlation between the cipher text and the plain text. These prevention measures contribute to the complexity of the searching mechanism.

The complexity of the prevention measures compels the searchable encryption system to trade-off between performance and security. In searchable encryption systems, the *access pattern* and *search pattern* are the most common security factors to consider for these trade-offs.

Search pattern. is defined as any information or pattern that an attacker can use to derive if random queries are related to keywords [24]. In some of the earlier research works (*e.g.,* [11, 25, 37]), the searchable encryption system reveals the search pattern to make the search operation more efficient. In a searchable encryption system by Curtmola *et al.* [24], the trapdoor is created using deterministic encryption, thus the system leaks the search pattern. Similarly, research works undertaken in [11, 25, 37] compromise search pattern to improve search efficiency.

Access pattern. is defined as any information that the attacker can use to determine the frequency at which files are accessed or associate to the query. For example, an observant and patient attacker could sniff the network connections to understand which files are searched for most frequently, thus determining the most important documents in a dataset.

In the context of searchable encryption, Goldreich *et al.* [42] achieved fully secure searchable encryption using Oblivious RAM (ORAM) to hide the access pattern. ORAM periodically *shuffles* the data blocks so that the user avoids accessing the same data (memory) block for the same retrieved data. However, in this work, the initial setup phase is computationally expensive and the search phase requires logarithmic rounds of interaction between the user and the server to narrow down the search results. This makes it a burden on the bandwidth and not suitable for large scale datasets.

Another work aimed at hiding the access pattern has been carried out by Boneh *et al.* [32] using Private Information Retrieval (PIR) technique. In a system consisting of n replicated servers, the user sends homomorphically encrypted queries [43, 32] to the n servers to retrieve data blocks from the servers that collectively make up the result [44]. The system has to *touch* all of the replicated servers, thus imposing a significant communication overhead to retrieve one single query.

To provide an efficient searchable encryption system, Song *et al.* [11] resorted to the weakened security guarantee, *i.e.,* revealing the access pattern and search pattern but nothing else. Later works (*e.g.,* [24, 25, 40]) attempt to find a proper method to balance a reasonable access and search pattern leakage and improve the search performance.

DOI: 10.1002/cpe

We should note that even with these security compromises, it is still difficult for an attacker to derive helpful information from the dataset. In fact, the attackers cannot learn anything beyond the importance of certain documents.

4.3. One-to-Many Order Preserving Symmetric Encryption (OM-OPSE)

In searchable encryption systems, data users generally need to see the search results ordered by their relevance to the search queries. As such, a searchable encryption system needs a method to assign relevance score to each found document and rank them for the user.

One way to achieve ranking of the search results is to assign an importance value (*i.e.,* weight) to the extracted keywords in the index structure. The weight of an extracted keyword can be obtained based on its importance in the document [30, 45]. These weights are typically hidden using deterministic encryption methods [46]. However, because deterministic encryption is a one-to-one mapping of plain text to cipher text, this encryption method can potentially leak some information about the documents [46]. In fact, an internal attacker can potentially deduce the distribution of encrypted scores in the data collection.

Given the context background of the corpus, the attacker can utilize the retrieved score distribution to gain knowledge about the documents or even break the encryption of the system [35]. To prevent this, certain research works have modified Order Preserving Symmetric Encryption (OPSE) which is a deterministic encryption method that preserves numerical order of the plain text [47].

Wang *et al.* [36] and Sun *et al.* [14] utilized OPSE to create a more efficient approach (known as one-to-many order-preserving mapping) to increase obfuscation of the original encrypted scores while maintaining plain text order. Instead of mapping the plain-text score to a single encrypted document, the encrypted score is assigned within a randomly appointed bucket within a predefined range. Therefore, the randomness of score distribution increases and the probability of being predicted decreases. Several research works have demonstrated that modifying OPSE is "as-strong-as-possible" encryption technique [36, 14].

5. TAXONOMY OF SEARCHABLE ENCRYPTION SYSTEMS

There have been various research efforts to enable different forms of search operation on encrypted data. However, a lot of these efforts are rooted in some general approaches. The goal of this section is to provide a comprehensive taxonomy of current searchable encryption systems and supply an overview of the conducted research in this domain. The taxonomy of searchable encryption systems is provided in Figure 3. As we can see in this figure, the searchable encryption systems can be broadly categorized based on the type of search they perform into three main types, namely Keyword Search; Regular Expression Search; and Semantic search.

5.1. Keyword Search

5.1.1. Sequential Scan Keyword Search Song *et al.* [11] pioneered the idea of user-side encryption and keyword-based search over encrypted documents. The user intends to retrieve encrypted documents that contain her search keyword. In the proposed scheme, each keyword W_i of a document is encrypted independently in two encryption layers. First, they pre-encrypt the keyword W_i with $E(W_i)$ into n bits. This is then split into two parts: the right part (R_i) consists of m bits and left part (L_i) consists of $n - m$ bits.

Second, the left part is encrypted with a stream cipher that can be checked for matches using XOR.

When a user requests documents that includes a set of interested keywords, she submits the encrypted version of each keyword in the query: L_i and k_i, the server performs XOR matching on $n - m$ bits of each cipher text to see whether C_i XOR L_i is of the form $(S, F_k i(S))$ for some S [11]. In this system, since every word is independently encrypted, to find all the matches, the server has to follow the above steps word-by-word for the entire document. For this reason, this method of searching is known as sequential scan.

DOI: 10.1002/cpe

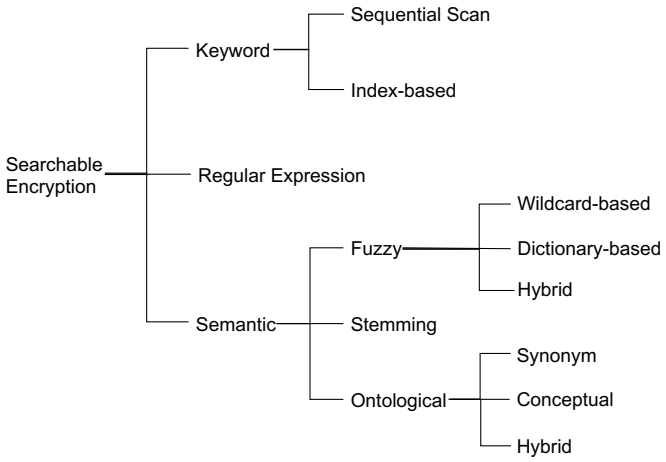

Figure 3. Taxonomy of current searchable encryption systems.

Another work in Sequential Scan category is PEKS (Public key Encryption with Keyword Search) [32]. PEKS encrypts each keyword in the uploaded document based on a public key using the Bilinear Diffie-Hellman [32] or Trapdoor Permutation [32] method. On the client side, the data owner uses her private key to verify if her query keyword occurs in a document. In this method, the server needs to scan every cipher text (*i.e.,* keyword) of each document to find the occurrences. The method is computationally expensive because the search cost is proportional to the number of the documents in the dataset.

There are not many dominant research works that follow this fashion due to the limiting time complexity for searching through the whole dataset, and given that datasets are growing in volume. More advanced and efficient searchable encryption systems are mentioned later.

5.1.2. Index-Based Keyword Search In order to deal with the inefficiency of the Sequential Scan keyword search, Song *et al.* [11] proposes a data structure called an "index" that contains a list of keywords mapped to their original documents using a document pointer (also known as document identifier [26]). The keyword W_i is encrypted as $E(W_i)$ and the document identifier is encrypted using the result of a key generator function f_k with input of the hashed keyword $E(W_i)$. In the Index-based keyword search systems, instead of searching sequentially (word-by-word) in every document, the system only needs to check the index structure for the interested keywords. From the results, the system retrieves the document identifiers and sends them back to the client.

Goh *et al.* [37] was one of the first works to introduce the secure index-based search method. In this work, the system creates an index for each document before encrypting and uploading it. The index consists of keywords that are considered relevant to the document. There are two main ways to construct the secure index: IND-CKA [37] and the efficient variant construction called Z-IDX [37]. Both of these methods use a Bloom Filter [37] as an index for each document to track its keywords. At search time, the system creates the search trapdoor and uses the Bloom Filter to check if the trapdoor is contained in the dataset.

Ren *et al.* [40] directly extended the two-layer encryption idea originally presented by Song *et al.* into a hashed index representation, seeing significant speed increases. In addition, they introduced another layer of randomized XOR-homomorphic encryption on the search query to further obfuscate the query to adversaries watching the network stream. Liu *et al.* [48] presented a searchable encryption method with multiple data sources where the cloud index is composed of multiple Indices from different sources. Research works such as [35, 36] introduced a similarity score for keyword search using formulas based on the term frequency of the keywords. Introducing a similarity score allows the result list to be ranked for the end users. We will review their scoring and ranking keyword against document and query in details in Section 5.3.3.

DOI: 10.1002/cpe

Earlier works from [37, 11, 32, 24] used index-based keyword search with their owned methods to perform single keyword or disjunctive keyword search on encrypted documents. However, as the needs of data users grow to require more accurate search results, different techniques are introduced on to the index to support multi-keyword and conjunctive keyword search. MRSE [49] is one of the first works to propose a solution to that demand. Cao *et al.* [49] define a dictionary consisting of all keywords, with each keyword having a defined location in the dictionary. Data files and search queries are represented by binary vectors which then are use "inner product computation" to measure the similarity of the data file and the query. Cao *et al.* [49] also apply their own internal ranking to return relevant ordered result. Later on, Z. Xu *et al.* [50] improve MRSE in so its fixed dictionary can be extended dynamically. Their works also improve the ranking algorithms by using access frequency in weighing the matched data file.

5.2. Regular Expression Search

One expansion to the keyword-based searchable encryption is to allow users to perform a regular expression search on encrypted data. A preliminary approach by Song *et al.* [11] proposes to create all possible variations of a given regular expression. For instance, for the query $ab[a-z]$, it would generate all 26 possible search queries, namely $aba, abb, \ldots, abz$. This approach only works for simple regular expressions and is not scalable for those with high degree of variability, *e.g.,* a^*b^*.

RESeED [20, 51] is a regular expression search system for encrypted data. RESeED operates based on two data structures: a *Column Store*, which is an unencrypted inverted index, representing keywords and the documents they appeared in, and an *Order Store* which is a fuzzy (hashed) representation of keywords within a document. For a given search phrase, RESeED builds a Nondeterministic Finite Automaton (NFA) [20]. The NFA is then partitioned into sub-NFAs that can be matched against keywords in the Column Store. For the documents found in the previous step, their Order Store is checked to confirm the keywords are in the same order as the regular expression.

5.3. Semantic Search

Searchable encryption solutions providing keyword or regular expression-based search abilities are useful when users exactly know what keywords they are searching for in the documents. However, with a growing collection of documents and the emergence of big data [52, 53], the data users may not remember the exact keywords they want to retrieve, or they might want to search for documents that are more broadly related to a topic [28, 34].

For instance, in a hospital with encrypted medical records, a doctor may desire to search for records using the query "heart disease". While the doctor is interested in documents containing the exact query terms, she is also interested in documents with semantically related terms (*e.g.,* "heart attack" or "chest pain"). Therefore, a semantic search is needed to return documents related to terms in the query and to avoid redundant searching attempts.

Many research works have been undertaken to enable different forms of semantic searchable encryption. As we can see in the taxonomy (Figure 3), these semantic search systems can be further categorized into three main types, namely Fuzzy Keyword Search; Stemming Search; and Ontological Search.

5.3.1. Fuzzy Keyword Search A Fuzzy Keyword Search improves the usability of a system by searching for close matches to the query if it fails to find sufficient matches for the exact query. Although these systems are categorized as search systems, they may not be directly used for search purposes. Fuzzy search can be used to make a system tolerant to user's typos [54].

Fuzzy keyword search systems work based on *edit distance* that measures the similarity between two strings s_1 and s_2 [55]. Edit distance is defined as the number of string operations needed to transform s_1 to s_2. The possible string operations are insertion (insert a character into a string), substitution (replace one character with another in a string), and deletion (remove a character from a string) [26]. Let $D = \{d_1, d_2, \ldots, d_n\}$, a collection of documents stored on an untrusted third-party server (*e.g.,* cloud); $W = \{w_1, w_2, \ldots, w_n\}$ a unique set of keywords with a fixed edit

DOI: 10.1002/cpe

distance d; and (s, k) a search trapdoor with threshold $k \leq d$. Then, a fuzzy keyword search system outputs a list of documents that possibly contain keyword w, if $w \in W$; else return document where $ed(w, w_i) < k$.

Li *et al.* [26] provide a fuzzy keyword search system that injects all possible words w_i that satisfy $ed(w, w_i) < d$ where w denotes the extracted keyword. For example, the fuzzy list of the keyword CAR is {*ACAR, CAAR, CARA, . . ., CARZ*}. This list of fuzzy variants is sent to the server where the index structure is located and includes those keywords and their associated documents. Similarly, in the Retrieval phase, the query is augmented with fuzzy variants, sent to the server to be matched, and a list of documents is returned. However, this approach is computationally expensive. For instance, if $d = 3$ the number of possible variants is $\frac{4}{3} \cdot k^3 \cdot 26^3$. To improve this, the authors [26] propose a wild-card-based technique that inserts wild card (*) to represent the fuzzy character or omit a letter from w_i. For instance, the fuzzy key set of CAR is $\{*CAR, C*AR, CA*R, CAR*, CAR\}$ and $\{CAR, AR, CR, CA\}$. These techniques significantly reduce the size of the index structure, thus improving the search time.

In a later study, Liu *et al.* [56] proposes a Dictionary-based Fuzzy search method. The method takes a dictionary and a predefined edit distance value and generates a set of fuzzy keywords for each keyword in the search queries. However, instead of inserting a wild-card, this technique only injects words that exist in the given dictionary. At the search time, the query goes through the same process to get the fuzzy set extension from the dictionary and encrypts it before sending it to the server to be searched.

In fuzzy keyword search, the system needs to search for the entire list of fuzzy keywords, which imposes an extensive overhead for a large edit distance. Therefore, to further enhance the fuzzy searching performance, Wang *et al.* [12] build a tree from the fuzzy keyword set that reduces the search to $O(log(n))$ of the fuzzy list's size.

5.3.2. Stemming Search Utilizing the Fuzzy Keyword Search makes searchable encryption systems more resistant to minor typos, but in many cases does not exactly cover the semantic perspective. In fact, two keywords having a close lexicographical distance does not necessitate that they are semantically related. Different words (e.g. "student" and "studying"), despite having a large edit distance (4), are highly semantically related. The stemming search method aims at solving this problem based on the belief that semantically related words tend to start from the same root (stem).

The model is the same as other searchable encryption systems: a user encrypts documents and extracts keywords as an encrypted index. The difference in this system is the additional step it takes to convert the keyword set into a set of stem words. When a user searches for a query, each query keyword is replaced by its stem [13]. There are three ways to extract the stem of a word:

- Affix stripping: This method applies well-known stemming algorithms to find the stem of a word. Prominent methods include the J.B. Lovins [57] and Porter stemming algorithms [58]. These algorithms remove the suffix and prefix to get the root of a word. However, these algorithms require knowledge of the language and can be computationally costly.
- Statistical Stemming: known as n-gram stemming, statistically finds the frequency of a contiguous sequence of n items from a given sequence of text in the whole document [59]. The least frequent n-gram is considered to be the stem (also called root).
- Hybrid: Utilizing both of the aforementioned methods to find the stem.

Both the uploaded documents and the search query extension go through the same process to get the stems of the extracted keywords. These stemmed keywords are stored in the index on of the third-party server (*e.g.,* public cloud) and used in the search process.

5.3.3. Ontological Semantic Search Fuzzy keyword and stemming searchable encryption methods cannot truly capture the semantic essence in searching. For example, if a user intends to search for robbery, she may be interested in seeing results about "burglary" or "break in" as well. However, neither the fuzzy keyword nor the stemming method can capture this type of semantic. In fact, the semantically related words neither share the same stem nor have a close edit distance. To resolve this problem, ontological semantic search was invented to find more meaningfully related data to the

DOI: 10.1002/cpe

original query [28, 34]. As we can see in the taxonomy of Figure 3, Ontological semantic search can be achieved using synonym semantics, conceptual semantics, or a combination of these semantics. In this part, we explore these schemes, however, we first revisit some of the preliminary concepts that enabler Ontological Searching:

- *Semantic Relationship:* Psycholinguists, Church *et al.* [60] propose that word association can be inferred from a statistical description of semantic relationship between words defined by the co-occurrence of those words. To calculate the similarity score between two keywords, Sun *et al.* [14] use data mining methods to effectively find out the co-occurrence degree between terms in a dataset. For two string x and y, the similarity score information $I(x, y)$ is defined based on Equation 1.

$$I(x, y) \equiv \log_2(\frac{P(x, y)}{p(x)p(y)}) \tag{1}$$

  where $P(x, y)$ is the probability that x and y appear together; and $p(x)$ and $p(y)$ are the probabilities that x and y appear independently in the collection. Higher values of the similarity score express more relevance between x and y.
  Another method to measure the similarity is based on Cosine similarity [14, 61] that can be calculated using the vector representation of documents and queries to get a relevance score of the closeness between the queries and documents.
- *Inverted Index:* A data structure that maps each unique keyword to the documents that contain them. This structure keeps track of a list of keywords throughout the whole dataset, with each keyword associated with list of documents that it appears in. For some work [14, 18, 35] in order to further assist ranking functionality, a normalized numerical relevance score (value between [0,1]) is often also given along with each document to indicate the relevance of it to the keyword.
- *Ranking function:* In a large dataset, it is common for an abundant number of documents to match a certain semantic search query with different degrees of relevance. Therefore, it is necessary for the user to receive the list of documents ranked in order of relevance. This introduces the need for a ranking function that measures the relevance of a matching document to the search query. The most common ranking function is known as $TF \times IDF$ (term frequency, inverted document frequency) [14, 15] where TF indicates the significance of that keyword in the document and IDF indicates the significance of the keyword over all documents in the dataset.

Different methods have been developed to measure the relevance of a given keyword in searchable encryption systems. Woodworth *et al.* [28] extend the Okapi BM25 standardized text retrieval method [28] which uses term frequency and inverse document frequency to calculate the semantic relevance of a given query to a document. This score is later used to rank documents in the result set.

Sun *et al.* and Xia *et al.* [14, 21] propose different mechanisms for semantic searchable encryption by extending the query keywords and ranking the result set. The data owner constructs *metadata* for each document and sends the encrypted *metadata* to a trusted server (*e.g.,* a private cloud) to build an inverted index and a semantic relationship library (SRL). The inverted index is then sent to reside in the public cloud. Upon receiving a search query, the trusted server extends the query using the semantic relationship library to get ontologically related keywords and synonyms. Then, it encrypts the extended keywords and sends them to the public cloud to look up the index structure. The returned documents are ranked based on the relevance ranking functions and are sent back to the user.

In another study, Moh *et al.* [15] introduce three schemes to learn the semantic meaning of a search query in order to produce accurately related result. Particularly, they propose Synonym-Based Keyword Search (SBKS), Wikipedia-Based Keyword Search (WBKS), and a combination of the two schemes called WBSKS. In the *SBKS* scheme, alongside the process of extracting important keywords of the document, synonyms that represent the semantic of a document are collected. The

DOI: 10.1002/cpe

encrypted version of these keywords and synonyms are sent to the cloud to form a searchable index. Similarly, the search query is also extended with its synonyms to form the trapdoor. Then the trapdoor is compared with the cloud index structure to find documents that semantically match the search query. In *WBKS* scheme, a pre-defined set of Wikipedia articles (WKS) are collected and a vector representation (VR) [15] of them are created using the term frequency and inverse document frequency ($TF \times IDF$) technique.

When a document is uploaded, its VR is then compared against the VR in WKS and the obtained score is stored in the index structure. In the search phase, the user query is converted to a VR and is compared to the WKS library using cosine similarity method. This score is then added to the trapdoor to let the server know how semantically related the query is to the WKS library. The cloud server computes the cosine similarity score of the trapdoor and the existing entries in the index to create a ranked list of documents which it sends to the user. In *WBSKS* scheme, both SBKS and WSKS techniques are used. SBKS is used to expand the query in the Search phase and WSKS is used to construct the expanded index of the uploaded documents of the Setup phase.

Another way to get the semantic meaning is by using WordNet [18]. WordNet is a tool created by Princeton University that contains a dictionary including word definitions and their synonyms. For example, Yang *et al.* [62] utilize WordNet to construct a semantic keyword set. Before the query is encrypted and searched with, each query keyword is expanded with the provided synonyms.

Woodworth *et al.* [28] further propose work that reduces the number of terms put in the index by extracting only the most important terms from an uploaded document, and extends the query using Wikipedia and Synonyms to capture the broader meaning of the query. The result is a space-efficient index which still achieves an accurate semantic search.

Table I provides a summary of categorization of different searchable encryption approaches in terms of components mentioned in Figure 2. We point out references to the works in each searchable encryption type, so readers can easily compare the difference between these works.

Table I. Categorizing different studied searchable encryption systems in terms of components they contain.

Search Approach	File Extractor	Using Index File	Multi-keyword Searching	Query Expansion	Ranking Search Result
Keyword Search	N/A	[37, 48, 35, 36, 40]	[49, 50]	N/A	[35, 36]
Regular Expression Search	N/A	N/A	N/A	[20, 51]	[13, 11, 36, 25, 37, 32]
Semantic Search	[28, 54, 14, 18, 35]	[28, 14, 18, 35]	[28]	[28, 54, 26]	[28, 14, 21, 15]

6. SECURITY ANALYSIS OF SEARCHABLE ENCRYPTION SYSTEMS

Based on security criteria of searchable encryption systems (mentioned in Section 4.1) and the level of information leakage, we can categorize current searchable encryption systems into four security levels. The differences between these security levels are summarized in Table II and are explained below.

Table II. Security level of current secure search systems.

Security Level	Leak Plain Text Document	Using Trusted Computational Base	Leak Index Data	Plain Data Stored Remotely	Leak Access Pattern	Leak Search Pattern	Related Works
Somewhat secure	No	Yes	No	Yes	Yes	Yes	[14, 20, 12]
Semi secure	No	No	Partially	No	Yes	Yes	[28, 18]
Secure	No	No	No	No	Yes	Yes	[13, 11, 36, 25, 37, 32]
Fully secure	No	No	No	No	No	No	[42]

Somewhat Secure The searchable encryption systems in this category often deploy a trusted server (also known as a private cloud [14] or a gateway [20]) in between the third-party server

DOI: 10.1002/cpe

(*e.g.,* public cloud) and the client device. The searchable encryption systems that leverage edge/fog computing [63] paradigms also fall under this category.

In Somewhat Secure searchable encryption systems, unencrypted documents are sent to the private cloud, where they are parsed and encrypted to form the index structure. These systems not only leak access and search patterns, but also expose the documents' contents to an internal attacker of the trusted server. Sun *et al.* [14] make use of the computational capacity of private clouds to build a Semantic Relations Library (SRL). The SRL quantifies the semantic distance of each term to other terms based on their co-occurrence in the dataset. However, SRL is maintained unencrypted on the private cloud, and thus is susceptible to attacks on the private cloud.

Semi Secure In searchable encryption systems with this level of security, the auxiliary index structure is partially encrypted. That is, some information about the documents or the keywords are not encrypted and can be leaked from the index structure.

In [18], the index structure keeps track of encrypted keywords, and for each keyword it contains a list of documents that include the keyword. However, the weight (*i.e.,* score) of the keyword in each document is maintained in an unencrypted manner. Therefore, the index structure is partially encrypted and reveals the relevance score that represents the relatedness of a given keyword to a document. In [28], Woodworth *et al.* provided a semantic search system for encrypted data in the cloud. Their system, named S3C, also exposes the frequency of a keyword within the original documents. These unencrypted data are considered security holes in the searchable encryption system that attackers can take advantage of and conduct a statistical attack.

Secure Such searchable encryption systems do not trust any part of the system, except the client's device. Also, the auxiliary index is properly secured and does not expose any plain text data to the server. Keywords in the index structure are hashed using SHA-1 or HMAC-SHA [36, 64] methods. Also, relevance scores are encrypted using OM-OPSE method (as explained in Section 4.3).

Because of the aforementioned reasons, Secure searchable encryption systems are less prone to internal attacks, in compare to those in the Somewhat or Semi Secure categories. However, the Secure searchable encryption systems still leak the access and search patterns. Although there are methods (*e.g.,* Private Information Retrieval [65, 31], Oblivious RAM [42]) to avoid such leaks, they slow down the search process. Thus, Secure searchable encryption systems prefer to expose these leaks in favor of the search performance. Instances of Secure searchable encryption systems can be found in [13, 11, 36, 25].

Fully Secure The searchable encryption systems in this category provide full security of the data and the search operation. Systems in this category do not even reveal the search and access pattern to the server.

Research undertaken by Goldreich *et al.* [42] falls into the fully secure category. It provides search and access pattern security through the use of ORAM (see Section 4.2). This strong security, however, comes with poly-logarithmic rounds of communication between the server and the user, which restrains its usability. In the same research, the authors introduced another method that needs only two-rounds of communication between the users and the server. Nonetheless, it induces a square root complexity overhead in the set up phase.

7. APPLICATION DRIVEN DOMAIN

Certain regulations [66] require some businesses and industries (*e.g.,* health care) to store their data in a secure a manner. As such, they need to store their data in an encrypted format, if they were to use a third-party servers or a public cloud for storage. Therefore, there have been several efforts from those businesses to tailor searchable encryption solutions for their specific domains. This section provides an overview of such works. In the rest of this section, we investigate such domain-specific searchable encryption systems.

DOI: 10.1002/cpe

Health care For health care providers, all Personal Health Records (PHRs) have to be encrypted to guarantee a patients privacy [67, 68]. Health care providers focus on maintaining the privacy of PHRs in an emerging patient-centric model that uses public clouds to easily store, retrieve, and share health information between medical centers. Although the public cloud solutions are convenient and attractive for health care providers, privacy concerns restrain its potential [69].

The owner of the PHR has the utmost right to share and distribute the decryption key to other users for personal or professional purposes [67]. However, the encrypted PHR makes key functionalities, such as keyword search by multiple users challenging. Multiple data users need to request the decryption key to access the encrypted PHR. However, the key requests by users are generally unpredictable, making it challenging for the data owner to manage key distribution, as they are not always online [67]. To overcome this challenge, an attribute-based encryption (ABE) scheme is proposed [70]. In ABE, there is a set of attributes that manages the access policies. Only users with the valid decryption key that matches the attribute can decrypt the patient's PHR and users with a revoked key cannot do anything with the encrypted PHRs. The ABE encryption scheme also enables a patient to exchange PHRs with a group of users without prior knowledge of the complete list of users.

In [19], Naseeruddin *et al.* developed an ABE-based schema that allows the system to audit user access, allowing it to identify the source of a breach and detect if information is distributed properly or if there was illegal access to the data. An encrypted PHR will be delegated to n trusted authorities from the data owner (patient). To access and search on the encrypted data, a data user needs to acquire the approval of k trusted authorities. Their work modifies ABE with a signature threshold $A(k, n)$ where a valid message is guaranteed if there are at least k valid signature shares out of n verified parties. In the Key generation algorithm of ABE, the key is generated based on an ABE-modification to encrypt the data before outsourcing it to the public cloud, which can be searched later to retrieve necessary information.

Another modification of ABE is proposed in [67, 71] to add another layer of fine-grained privacy protection beyond the underlying cryptographic mechanisms to enable searchable encryption or data access control. The study develops a hierarchical predicate encryption (HPE), in which the data owner generates a key and distributes it to a local trusted authority (LTA). These LTAs are able to delegate the key to allow authorized users to search to that patient's PHR. In the search phase, the server has to verify that the user has a valid signature from a registered LTA before performing search. It is worth noting that this study only supports keyword search, and not other forms of search.

These methods are appropriate for normal medical procedures, but may hamper first-aid treatment in emergency situations in which a patient's life is at risk and an authorized user is not present. To combat this challenge, Yang *et al.* [72] implement adaptive dual-layer access control, in which normal approved users are allowed to see all of the patient's health care data, while outside users can view a subset of the data in emergency situations using a password-based break-glass access mechanism. This approach provides data security while limiting danger in emergency scenarios.

Law Enforcement Other industries, such as law enforcement, can potentially have privacy restrictions in granting access to their datasets. Data owners, in this industry, require data that is stored in external sources (*e.g.,* in a cloud) to be encrypted, and searchable encryption systems may only be used to enable search over data without providing direct access to the data. For example, a detective needs to know if the subject has any matching record in the court reports. The detective, however, does not need to (or cannot) access the information of all people in the court system.

Text Editors Li *et al.* [26] propose a secure search system which could handle user typos through a fuzzy keyword search. Wang *et al.* [12] used a similar approach to find matches for similar keywords to the user's query by using edit distance as a similarity metric, allowing for words with similar structures and minor spelling differences to be matched. However, these methods reveal the topic of the external sources that an attacker can use to categorize the data sources based on query retrieval.

DOI: 10.1002/cpe

Other Domains Financial and military domains generally have strict data privacy and confidentiality rules [73]. However, there is currently no dedicated study on the particular use of searchable encryption in these domains.

8. FUTURE RESEARCH DIRECTIONS

In this section, we suggest various directions that we envisage are necessary and researchers in the searchable encryption domain could find their interest in. Each of these topics or even a combination of multiple suggestions will push the current boundaries of searchable encryption.

1. Clustering Encrypted Data Since the field's inception, searchable encryption systems have primarily strived to improve search accuracy, performance, and security. Despite many provable improvements to security, searchable encryption solutions still need to enhance their performance and maintain the real-time quality of the search operation. This is particularly important as datasets get substantially larger, and as more businesses look to outsource their big data to the cloud. One promising approach to dealing with big data is to cluster the data and restrict the search to relevant clusters. This approach presents several challenges: clustering encrypted keywords based on some meaningful semantic data, determining which clusters are most relevant at search time, and determining how many of those clusters should be searched. A novel feature of distributing and arranging encrypted data into meaningful clusters is presented by Chen *et al.* [74]. By combining affinity propagation and K-means clustering method, called CAK-means model, they can partition related data into a certain number of clusters. This model maintains the closeness of relevant data files and supports I/O operations that further improves the uploading process and system as a whole.

2. Searchable Encryption Across Multi-source Datasets As datasets grow larger and more organizations opt to use remote cloud storage, it becomes increasingly likely that users and organizations will need to search across multiple sources. For example, a law enforcement officer who needs to obtain information on a suspect may need to search across datasets from other jurisdictions. One challenge to this is that each of these organizations can potentially have their own privacy and confidentiality policies and use various encryption standards [75]. The datasets may also have different characteristics, such as document structure or length. Further research is needed to deal with a wider variety of data, and combining multiple levels of security. One of the recent works following this direction is from Gou *et al.* [76], where they set up a scenario of trusted clients issuing queries on a cluster of server nodes. They proposed EncKV, a system to store encrypted data and support rich query retrieval utilizing standard data partitioning algorithm. To provide rich query across distributed data storage, the client maintains a small hashing structure to keep track of metadata.

3. Edge Computing for Searchable Encryption Most current searchable encryption systems have a two-tier architecture, utilizing primarily a user-end and cloud-end. The user-end includes the client machine, often in the form of a thin-client (*e.g.,* smart phone or tablet) with limited computational power. This often creates a performance bottleneck, as much of the document and query pre-processing must occur on the client-end since it is the only trusted component. An edge computing paradigm can be utilized to bypass this bottleneck. Use of this paradigm would add additional architectural tiers along the network between the server and client. Off-loading computational work to an edge node can substantially improve performance, but weakens security, as edge nodes are more vulnerable to attacks than clients. Further research efforts should determine what processes can be performed on edge nodes while minimally compromising security. ENSURE is an edge computing based searchable encryption system schema [77]. It introduces a way to provide the searchable encryption system functionality through mobile devices. This is achieved via outsourcing heavy tasks to edge devices, releasing the end device from intense computation

DOI: 10.1002/cpe

workload and saving energy consumption. The article also improves security by relying on the trustworthiness of the edge devices rather than on the public (cloud) servers.

4. Utilizing Blockchain in Searchable Encryption With recent breakthroughs, blockchain technology and the idea of decentralization are becoming more promising aspects and gaining more attention from the searchable encryption community. There are several proposals for utilizing blockchain in searchable encryption. Hu *et al.* [78] introduce a decentralized privacy-preserving search scheme leveraging smart contracts in which a blockchain verifies the accuracy and fairness in the contract between users and cloud provider. In other research, Cai *et al.* [79]suggest a decentralized storage platform, a concept in which people lease their hardware capacity as a service. However, it is questionable that outsourcing data to other storage providers could create threats to data integrity, which is mentioned in [80], and thus affect autonomous payment of people involving the service [79]. The system proposed by Cai *et al.* [79] utilizes crypto-currencies and cryptographic incremental hashing to solve the mentioned concerns. This is one of the very first works integrating blockchain with searchable encryption systems which starts a promising direction. Nevertheless, it would need more investigation and experimentation to develop more mature approaches in the future.

9. SUMMARY

In this work, we surveyed current searchable encryption systems and techniques that perform various forms of search operation on encrypted data located on an untrusted third-party cloud. We identified common components of the searchable encryption systems and the overall architecture through which these components interact with each other. Then, we provided a taxonomy that categorizes the searchable encryption systems based on the type of search operation supported by them. We analyzed the security of the current searchable encryption systems and categorized them into four security levels. We investigated particular demands of searchable encryption systems in various domains, such as health care and law-enforcement. Finally, we offered suggested directions for future research, including clustering data for improved search speed, searching across multiple datasets, and utilizing edge computing and blockchains.

ACKNOWLEDGMENTS

We would like to acknowledge anonymous reviewers of the manuscript. This research was supported by the Louisiana Board of Regents under grant number LEQSF(2017-20)-RD-B-06, and Perceptive Intelligence, LLC.

REFERENCES

1. S. Marston, Z. Li, S. Bandyopadhyay, J. Zhang, A. Ghalsasi, Cloud computingthe business perspective, Decision support systems 51 (1) (2011) 176–189.
2. M.-G. Avram, Advantages and challenges of adopting cloud computing from an enterprise perspective, Procedia Technology 12 (2014) 529–534.
3. Z. Yan, X. Li, M. Wang, A. V. Vasilakos, Flexible data access control based on trust and reputation in cloud computing, IEEE Transactions on Cloud Computing (TCC) 5 (3) (2017) 485–498.
4. A. S. Sohal, R. Sandhu, S. K. Sood, V. Chang, A cybersecurity framework to identify malicious edge device in fog computing and cloud-of-things environments, Computers & Security 74 (2018) 340 – 354.
5. M. Nakayama, C. Chen, C. Taylor, The effects of perceived functionality and usability on privacy and security concerns about cloud application adoptions, Journal of Information Systems Applied Research 10 (2) (2017) 529–534.
6. V. Chang, Y.-H. Kuo, M. Ramachandran, Cloud computing adoption framework: A security framework for business clouds, Future Generation Computer Systems 57 (2016) 24 – 41.
7. J. Singh, T. Pasquier, J. Bacon, H. Ko, D. Eyers, Twenty security considerations for cloud-supported internet of things, IEEE Internet of Things Journal 3 (3) (2016) 269–284.
8. A. Meharwade, G. Patil, Efficient keyword search over encrypted cloud data, Procedia Computer Science 78 (C) (2016) 139–145.
9. Y. Wang, J. Wang, X. Chen, Secure searchable encryption: a survey, Journal of Communications and Information Networks 1 (4) (2016) 52–65.

DOI: 10.1002/cpe

10. M. Salehi, R. Buyya, Adapting Market-Oriented Scheduling Policies for Cloud Computing, in: Algorithms and Architectures for Parallel Processing, Vol. 6081 of ICA3PP '10, 2010, pp. 351–362.
11. D. X. Song, D. Wagner, A. Perrig, Practical techniques for searches on encrypted data, in: Proceedings of IEEE Symposium on Security and Privacy, 2000. S&P 2000., 2000, pp. 44–55.
12. J. Wang, H. Ma, Q. Tang, J. Li, H. Zhu, S. Ma, X. Chen, Efficient verifiable fuzzy keyword search over encrypted data in cloud computing, Computer science and information systems 10 (2) (2013) 667–684.
13. T. Moataz, A. Shikfa, N. Cuppens-Boulahia, F. Cuppens, Semantic search over encrypted data, in: Proceedings of 20th International Conference on Telecommunications (ICT), 2013, pp. 1–5.
14. X. Sun, Y. Zhu, Z. Xia, L. Chen, Privacy-preserving keyword-based semantic search over encrypted cloud data, International journal of Security and its Applications 8 (3) (2014) 9–20.
15. T.-S. Moh, K. H. Ho, Efficient semantic search over encrypted data in cloud computing, in: Proceedings of 12th International Conference on High Performance Computing & Simulation (HPCS), 2014, pp. 382–390.
16. C. Bösch, P. Hartel, W. Jonker, A. Peter, A survey of provably secure searchable encryption, ACM Comput. Surv. 47 (2) (2014) 18:1–18:51. doi:10.1145/2636328.
URL http://doi.acm.org/10.1145/2636328
17. G. S. Poh, J.-J. Chin, W.-C. Yau, K.-K. R. Choo, M. S. Mohamad, Searchable symmetric encryption: Designs and challenges, ACM Comput. Surv. 50 (3) (2017) 40:1–40:37. doi:10.1145/3064005.
URL http://doi.acm.org/10.1145/3064005
18. M. Saleem, M. Warsi, N. S. Khan, Secure metadata based search over encrypted cloud data supporting similarity ranking, International Journal of Computer Science and Information Security 15 (3) (2017) 353–361.
19. N. Ali, N. Pathan, S. P. Dubey, Privacy and protection of mobile health data on secure cloud storage, Imperial Journal of Interdisciplinary Research 3 (4).
20. M. Amini Salehi, T. Caldwell, A. Fernandez, E. Mickiewicz, D. Redberg, E. W. D. Rozier, S. Zonouz, RESeED: Regular Expression Search over Encrypted Data in the Cloud, in: Proceedings of the 7th IEEE Cloud conference, Cloud '14, 2014.
21. Z. Xia, Y. Zhu, X. Sun, L. Chen, Secure semantic expansion based search over encrypted cloud data supporting similarity ranking, Journal of Cloud Computing 3 (1) (2014) 8–17.
22. N. Cao, C. Wang, M. Li, K. Ren, W. Lou, Privacy-preserving multi-keyword ranked search over encrypted cloud data, IEEE Transactions on parallel and distributed systems 25 (1) (2014) 222–233.
23. Z. Fu, X. Sun, Q. Liu, L. Zhou, J. Shu, Achieving efficient cloud search services: multi-keyword ranked search over encrypted cloud data supporting parallel computing, IEICE Transactions on Communications 98 (1) (2015) 190–200.
24. R. Curtmola, J. Garay, S. Kamara, R. Ostrovsky, Searchable symmetric encryption: improved definitions and efficient constructions, Journal of Computer Security 19 (5) (2011) 895–934.
25. Y.-C. Chang, M. Mitzenmacher, Privacy preserving keyword searches on remote encrypted data, in: Proceedings of the 3rd International Conference on Applied Cryptography and Network Security, ACNS '05, 2005, pp. 442–455.
26. J. Li, Q. Wang, C. Wang, N. Cao, K. Ren, W. Lou, Fuzzy keyword search over encrypted data in cloud computing, in: Proceedings of the 29th Conference on Information Communications, INFOCOM '10, 2010, pp. 441–445.
27. Z. Fu, F. Huang, K. Ren, J. Weng, C. Wang, Privacy-preserving smart semantic search based on conceptual graphs over encrypted outsourced data, IEEE Transactions on Information Forensics and Security 12 (8) (2017) 1874–1884.
28. J. Woodworth, M. A. Salehi, V. Raghavan, S3C: An architecture for space-efficient semantic search over encrypted data in the cloud, in: Proceedings of 5th IEEE International Conference on Big Data (Big Data), 2016, pp. 3722–3731.
29. L. Ballard, S. Kamara, F. Monrose, Achieving efficient conjunctive keyword searches over encrypted data, in: Proceedings of the 7th International Conference on Information and Communications Security, ICICS '05, 2005, pp. 414–426.
30. Z. Fu, K. Ren, J. Shu, X. Sun, F. Huang, Enabling personalized search over encrypted outsourced data with efficiency improvement, IEEE transactions on parallel and distributed systems 27 (9) (2016) 2546–2559.
31. J. Baek, R. Safavi-Naini, W. Susilo, Public key encryption with keyword search revisited, in: Proceeding of the International Conference on Computational Science and Its Applications, Part I, ICCSA '08, 2008, pp. 1249–1259.
32. D. Boneh, E. Kushilevitz, R. Ostrovsky, W. Skeith, Public key encryption that allows pir queries, Advances in Cryptology-CRYPTO 2007 (2007) 50–67.
33. D. Cash, J. Jaeger, S. Jarecki, C. S. Jutla, H. Krawczyk, M.-C. Rosu, M. Steiner, Dynamic searchable encryption in very-large databases: Data structures and implementation., in: NDSS, Vol. 14, 2014, pp. 23–26.
34. J. Woodworth, M. Amini Salehi, S3BD: Secure Semantic Search over Encrypted Big Data in the Cloud, Concurrency and Computation: Practice and Experience (CCPE).
35. C. Wang, N. Cao, K. Ren, W. Lou, Enabling secure and efficient ranked keyword search over outsourced cloud data, IEEE Transactions on parallel and distributed systems 23 (8) (2012) 1467–1479.
36. C. Wang, N. Cao, J. Li, K. Ren, W. Lou, Secure ranked keyword search over encrypted cloud data, in: Proceedings of 30th IEEE International Conference on Distributed Computing Systems, ICDCS '10, 2010, pp. 253–262.
37. E.-J. Goh, Secure indexes, Cryptology ePrint Archive, report 2003/216 (2003).
URL http://eprint.iacr.org/
38. Z. Xia, X. Wang, X. Sun, Q. Wang, A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data, IEEE transactions on parallel and distributed systems 27 (2) (2016) 340–352.
39. M. Ding, F. Gao, Z. Jin, H. Zhang, An efficient public key encryption with conjunctive keyword search scheme based on pairings, in: Proceedings of 3rd IEEE International Conference on Network Infrastructure and Digital Content, IC-NIDC '12, 2012, pp. 526–530.
40. S. Q. Ren, B. H. M. Tan, S. Sundaram, T. Wang, Y. Ng, V. Chang, K. M. M. Aung, Secure searching on cloud storage enhanced by homomorphic indexing, Future Generation Computer Systems 65 (2016) 102 – 110.
41. P. Kocher, J. Jaffe, B. Jun, Introduction to differential power analysis and related attacks (1998).

DOI: 10.1002/cpe

42. O. Goldreich, R. Ostrovsky, Software protection and simulation on oblivious rams, Journal of the ACM (JACM) 43 (3) (1996) 431–473.
43. K. Banawan, S. Ulukus, The capacity of private information retrieval from coded databases, IEEE Transactions on Information Theory 64 (3) (2018) 1945–1956.
44. T. Mayberry, E.-O. Blass, A. H. Chan, Efficient private file retrieval by combining oram and pir., in: Network and Distributed System Security Symposium, NDSS '13, 2014.
45. Z. Fu, X. Wu, Q. Wang, K. Ren, Enabling central keyword-based semantic extension search over encrypted outsourced data, IEEE Transactions on Information Forensics and Security 12 (12) (2017) 2986–2997.
46. K. Li, W. Zhang, C. Yang, N. Yu, Security analysis on one-to-many order preserving encryption-based cloud data search, IEEE Transactions on Information Forensics and Security 10 (9) (2015) 1918–1926.
47. A. Boldyreva, N. Chenette, Y. Lee, A. O'Neill, Order-preserving symmetric encryption, in: Proceedings of the 28th Annual International Conference on Advances in Cryptology: The Theory and Applications of Cryptographic Techniques, EUROCRYPT '09, 2009, pp. 224–241.
48. C. Liu, L. Zhu, J. Chen, Efficient searchable symmetric encryption for storing multiple source data on cloud, in: Proceedings of the 2015 IEEE Trustcom/BigDataSE/ISPA - Volume 01, TRUSTCOM '15, 2015, pp. 451–458.
49. N. Cao, C. Wang, M. Li, K. Ren, W. Lou, Privacy-preserving multi-keyword ranked search over encrypted cloud data, in: 2011 Proceedings IEEE INFOCOM, 2011, pp. 829–837.
50. Z. Xu, W. Kang, R. Li, K. Yow, C. Xu, Efficient multi-keyword ranked query on encrypted data in the cloud, in: 2012 IEEE 18th International Conference on Parallel and Distributed Systems, 2012, pp. 244–251.
51. M. A. Salehi, T. Caldwell, A. Fernandez, E. Mickiewicz, E. W. D. Rozier, S. Zonouz, D. Redberg, RESeED: A secure regular-expression search tool for storage clouds, Software: Practice and Experience (SPE) 47 (9) (2017) 1221–1241.
52. S. Zobaed, M. Amini Salehi, Big Data in the Cloud, in: A. Zomaya, S. Sakr (Eds.), Encyclopedia of Big Data, Springer, 2018.
53. M. Pusala, M. Amini Salehi, J. Katukuri, Y. Xie, V. Raghavan, Massive Data Analysis: Tasks, Tools, Applications and Challenges, in: S. Pyne, B. L. . S. P. Rao, S. B. Rao (Eds.), Big Data Analytics: Methods and Applications, Springer, 2016, ISBN: 978-8-132-23626-9.
54. Z. Fu, X. Wu, C. Guan, X. Sun, K. Ren, Toward efficient multi-keyword fuzzy search over encrypted outsourced data with accuracy improvement, IEEE Transactions on Information Forensics and Security 11 (12) (2016) 2706–2716.
55. B. Wang, S. Yu, W. Lou, Y. T. Hou, Privacy-preserving multi-keyword fuzzy search over encrypted data in the cloud, in: Proceedings of IEEE Conference on Computer Communications, INFOCOM '14, 2014, pp. 2112–2120.
56. C. Liu, L. Zhu, L. Li, Y. Tan, Fuzzy keyword search on encrypted cloud storage data with small index, in: Proceedings of IEEE International Conference on Cloud Computing and Intelligence Systems, 2011, pp. 269–273.
57. J. B. Lovins, Development of a stemming algorithm, Mech. Translat. & Comp. Linguistics 11 (1-2) (1968) 22–31.
58. M. F. Porter, An algorithm for suffix stripping, Program 14 (3) (1980) 130–137.
59. N. Durrani, H. Schmid, A. Fraser, P. Koehn, H. Schütze, The operation sequence model-combining n-gram-based and phrase-based statistical machine translation, Computational Linguistics 41 (2) (2015) 185–214.
60. K. W. Church, P. Hanks, Word association norms, mutual information, and lexicography, Computational linguistics 16 (1) (1990) 22–29.
61. N. Cao, C. Wang, M. Li, K. Ren, W. Lou, Privacy-preserving multi-keyword ranked search over encrypted cloud data, IEEE Transactions on Parallel and Distributed Systems (TPDS) 25 (1) (2014) 222–233.
62. Y. Yang, X. Zheng, V. Chang, C. Tang, Semantic keyword searchable proxy re-encryption for postquantum secure cloud storage, Concurrency and Computation: Practice and Experience 29 (19).
63. S. Yi, Z. Qin, Q. Li, Security and privacy issues of fog computing: A survey, in: International Conference on Wireless Algorithms, Systems, and Applications, Springer, 2015, pp. 685–695.
64. F. Hahn, F. Kerschbaum, Searchable encryption with secure and efficient updates, in: Proceedings of the ACM SIGSAC Conference on Computer and Communications Security, CCS '14, 2014, pp. 310–320.
65. E. Kushilevitz, R. Ostrovsky, Replication is not needed: Single database, computationally-private information retrieval, in: Proceedings of the 38th Annual Symposium on Foundations of Computer Science, 1997, pp. 364–373.
66. N. J. King, V. Raja, Protecting the privacy and security of sensitive customer data in the cloud, Computer Law & Security Review 28 (3) (2012) 308–319.
67. M. Li, S. Yu, Y. Zheng, K. Ren, W. Lou, Scalable and secure sharing of personal health records in cloud computing using attribute-based encryption, IEEE transactions on parallel and distributed systems 24 (1) (2013) 131–143.
68. M. Javanmard, M. A. Salehi, S. Zonouz, Tsc: Trustworthy and scalable cytometry, in: Proceedings of the 7th IEEE International Symposium on Cyberspace Safety and Security, 2015, pp. 1356–1360.
69. J.-J. Yang, J.-Q. Li, Y. Niu, A hybrid solution for privacy preserving medical data sharing in the cloud environment, Future Generation Computer Systems 43 (2015) 74–86.
70. V. Goyal, O. Pandey, A. Sahai, B. Waters, Attribute-based encryption for fine-grained access control of encrypted data, in: Proceedings of the 13th ACM conference on Computer and communications security, Acm, 2006, pp. 89–98.
71. M. Li, S. Yu, N. Cao, W. Lou, Authorized private keyword search over encrypted data in cloud computing, in: Distributed Computing Systems (ICDCS), 2011 31st International Conference on, IEEE, 2011, pp. 383–392.
72. Y. Yang, X. Zheng, W. Guo, X. Liu, V. Chang, Privacy-preserving smart iot-based healthcare big data storage and self-adaptive access control system, Information Sciences.
73. A. I. Anton, J. B. Earp, Q. He, W. Stufflebeam, D. Bolchini, C. Jensen, Financial privacy policies and the need for standardization, IEEE Security & privacy 2 (2) (2004) 36–45.
74. L. Chen, N. Zhang, K.-C. Li, S. He, L. Qiu, Improving file locality in multi-keyword top-k search based on clustering, Soft Computing 22 (9) (2018) 3111–3121. doi:10.1007/s00500-018-3145-6.
URL https://doi.org/10.1007/s00500-018-3145-6

75. R. Fathi, M. A. Salehi, E. L. Leiss, User-friendly and secure architecture for authentication of cloud services, in: Proceedings of the 8th International Conference on Cloud Computing, IEEE Cloud '15, 2015.
76. Y. Guo, X. Yuan, X. Wang, C. Wang, B. Li, X. Jia, Enabling encrypted rich queries in distributed key-value stores, IEEE Transactions on Parallel and Distributed Systems.
77. Y. Guo, F. Liu, Z. Cai, N. Xiao, Z. Zhao, Edge-based efficient search over encrypted data mobile cloud storage, Sensors 18 (4) (2018) 1189.
78. S. Hu, C. Cai, Q. Wang, C. Wang, X. Luo, K. Ren, Searching an encrypted cloud meets blockchain: A decentralized, reliable and fair realization, in: IEEE INFOCOM, 2018.
79. C. Cai, X. Yuan, C. Wang, Towards trustworthy and private keyword search in encrypted decentralized storage, in: 2017 IEEE International Conference on Communications (ICC), 2017, pp. 1–7.
80. S. Basu, A. Bardhan, K. Gupta, P. Saha, M. Pal, M. Bose, K. Basu, S. Chaudhury, P. Sarkar, Cloud computing security challenges & solutions-a survey, in: Computing and Communication Workshop and Conference (CCWC), 2018 IEEE 8th Annual, IEEE, 2018, pp. 347–356.

DOI: 10.1002/cpe