Chapter 2
An Investigation Study of Privacy Preserving in Cloud Computing Environment

Ahmed M. Manasrah, M. A. Shannaq, and M. A. Nasir

Abstract Cloud computing allows users with limited resources to farm out their data to the cloud for computation, bandwidth, storage, and services on a pay-per-use basis. Consequently, researchers worldwide are trying to address issues related to the user's data privacy through proposing various methods such as outsourcing data in an encrypted form. However, encrypting data will conceal the relationships between data. Moreover, due to the voluminous data at the data centers, designing an efficient and reliable online-encrypted text-based searching scheme is challenging. Therefore, this paper surveys the state of the art on the data privacy preserving over the cloud through analyzing and discussing the various privacy-preserving methods that were proposed to sustain the privacy of the user's data. The pros and cons of the surveyed approaches are drawn in comparison with each other. Finally, the results are consolidated and the issues to be addressed in the future are concluded for the advancements in cloud data privacy preserving.

Keywords Cloud computing · Cloud storage · Privacy preserving

1 Introduction

The establishment of cloud has brought tremendous benefits to users and enterprises. The idea behind the establishment of the cloud is to allocate ubiquitous, on-demand access to processing resources and data storage to computers and other devices to store and process their data at a third-party data centers that might be located outside their premises. The allocated on-demand resources can be invoked and revoked with

A. M. Manasrah (✉)
Computer and Information Science Department, Higher Colleges of Technology, Sharjah, UAE

Computer Sciences Department, Yarmouk University, Irbid, Jordan
e-mail: amanasrah@hct.ac.ae; ahmad.a@yu.edu.jo

M. A. Shannaq · M. A. Nasir
Computer Sciences Department, Yarmouk University, Irbid, Jordan

B. B. Gupta et al. (eds.), *Handbook of Computer Networks and Cyber Security*,
https://doi.org/10.1007/978-3-030-22277-2_2

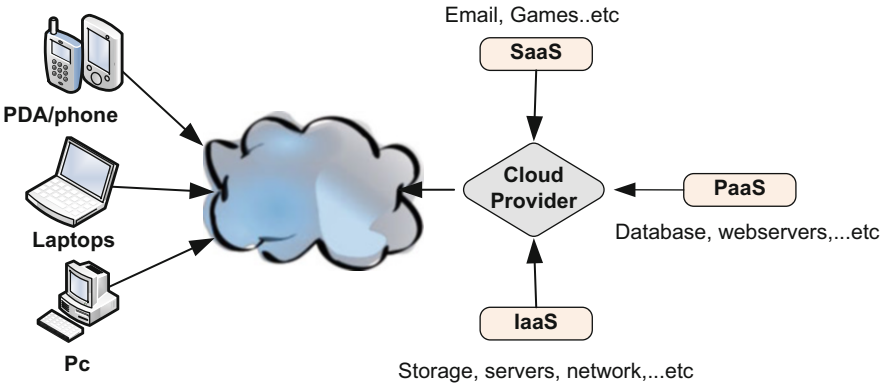

Fig. 2.1 Cloud component, typically Infrastructure as a Service (IaaS), Software as a Service (SaaS) or Platform as a Service (PaaS) [1]

minimal administration efforts. The shared resources aim to provide coherence and economy of scale, such as the utilities over the networks (electricity, gas, water, etc.). Therefore, companies and enterprises can avoid different infrastructure costs and focus more on their business and productivity.

A cloud provider (or cloud service providers or CSPs) offers some cloud computing components (see Fig. 2.1) on a "pay as you go or pay per use" basis. This may lead to high charges if the cloud-pricing model is not well adapted by the administrators.

With the constant growth in demand for cloud computing, the cloud provider might not meet the different organizations legal need while they need to contemplate the benefits of using the cloud against its risks. For instance, the control of the back-end infrastructure is limited only to the CSP. Moreover, CSPs often decide on the usage and management policies, which might abate the cloud user's ability over their deployment. Cloud users are also restricted with certain control and management policies of their applications, data and services, such as allocating certain amount of bandwidth for each customer and are often shared among other cloud users. Cloud computing involves constraints that make the progress in cloud computing services challenging; these constraints are consolidated in Table 2.1.

The reliance of the cloud computing usage by organizations and users has taken a long time since the time cloud computing came into existence. The reason behind this delay in adopting cloud computing is the security concerns because IT security is challenging even under the best of circumstances. Typically, the cloud environments are likely to have strong security measures deployed at their infrastructures. However, companies and organizations are of more concern of security at the CSP.

The CSP might not be able to meet the regulatory requirement of a company or organization. For instance, a law that allows the government to get at the data in secret is a demotivating factor for foreign companies to store their data inside such countries. Other countries may have even more rigorous government-access rules.

Table 2.1 Cloud computing constraints and challenges

Constraint/Challenge	Description
Naming heterogeneity	When the customers and the cloud service providers, using different names to identify attributes
Multi-occupancy	Allows multi-occupants to have an isolated environment for each one in terms of (CPU, memory, and network) in the same physical machine
Virtualization	Allows multi-occupants to execute their applications on the same physical environment, but separately
Forward secrecy	Old security keys cannot be accessed by any group member
Backward secrecy	Future generated keys should not be accessible to previous group members hence, cloud data is only accessible to privileged users
Searchable encryption	Encrypted cloud data should be searchable without decrypting the data neither the query and the returned records satisfy the search query

Typically, in the cloud environment, the data are processed or stored at data centers that are located far away from the organization city or country. Therefore, losing the control of the data is a security risk to most of the world organizations because in this case, someone else is controlling the data (i.e., the CSP). The concern is even amplified with free CSPs especially that SCPs can delete the outsourced data if they believe that the data violating some service terms [2–4]. Even though the demand for cloud computing is increasing, the concerns about users' data privacy are also increasing and formidable. Therefore, another set of issues concerning the advances in the field of privacy preserving for users' identity and their data also exists and acts as a barrier in this regard as shown in Table 2.2. Unfortunately, providing and preserving data privacy in the cloud have not been fully developed yet, and still require extra efforts in order to achieve successful results. Therefore, addressing all these issues could assist in designing novel privacy-preserving searching mechanisms over encrypted cloud data that are secure against intruders or attackers. Such designs could be a mark of success in the preservation of privacy in Cloud Computing.

In this paper, the issues related to cloud data privacy preserving are addressed. Various existing approaches related to data encryption concerning cloud data privacy preserving are discussed. After studying the existing approaches, issues and challenges are pointed out. To the best of our knowledge, this is the first survey that shortlist the issues and challenges of users and data privacy preserving over the cloud along the various possible solutions for the future researches.

2 Privacy-Preserving Methods

Various efforts have been made to address the preservation of data privacy over the cloud. This paper analyzes some of those efforts and provides a brief overview to the most known approaches in the field. This paper therefore classifies the privacy-preserving approaches in cloud computing into five broad categories as illustrated in Fig. 2.2.

Table 2.2 Privacy-preserving issues and challenges

Issue	Challenge
Insufficient control	The data are stored and processed in the cloud out of the data owner control
Lack of training and expertise	The constant change and complexity of the cloud environments forces the data owners to provide special expertise to manage the different cloud technologies. Therefore, recruiting and training talents are the barriers against implementing cloud strategies
Information disclosure	Since sensitive information and user's data move across the cloud, does the CSPs disclose any information to governments
Unauthorized storage/usage	Backups should not reveal neither it is possible to access and retrieve Sensitive information should not be accessed or revealed from Backups
Uncontrolled data proliferation	The data flow in the cloud should not be predictable neither controllable
Dynamic provision	The dynamic nature of the cloud should always keep the privacy of the data and their owners unclear, even for a legally responsible entity
Data accessibility, transfer and retention [5]	How the data on cloud are being accessed, destructed and by whom?
Location of data	The physical location of the storage servers may have legal implications (such as Jurisdiction issues)
Data security and disclosure of breaches	How the customer's data being protected by the CSPs. Does the CSP alert customers when cloud security is breached?
Addressing transborder data flow restriction [6]	Does the CSP adopt an international regulatory and compliance laws and rules? How the data protection across different regulatory and legal jurisdictions is maintained?

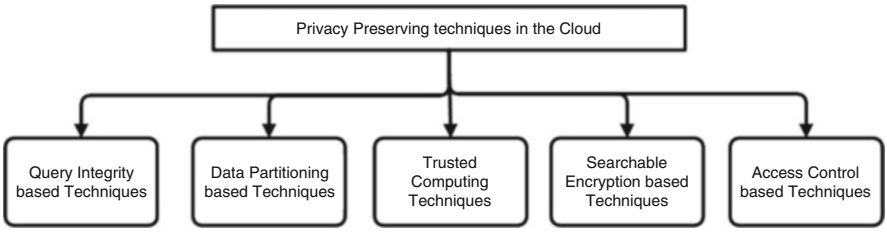

Fig. 2.2 Categories of privacy-preserving techniques in cloud computing

The following subsections examine most of the known cloud-based privacy-preserving methodologies and analyze these methodologies in terms of their pros and cons in comparison with each other.

3 Searchable Encryption-Based Techniques

Generally, IT managers and even individuals are likely to be cautious of delegating the control of their data to outside service providers because information stored at a third party may have weaker privacy protections than information in the possession of the creator of the information. Moreover, the outside provider has the right to change their underlying technology without their customer's consent, which may cause issues related to performance, and latency [4, 7]. Traditionally, data privacy is preserved by cryptographic primitives by the side of unique and secure identities for the queries and their responses jointly with usage/access rights policies. However, searching over the encrypted data is a formidable mission. Moreover, users normally lose control over their encrypted outsourced data in a tradeoff relation to their security and privacy preservation of the outsourced data. However, considering the diverse types of data that can be stored in the cloud and the user's demand for the data safety, preserving the data privacy in the cloud becomes even more challenging [8].

For instance, looking for certain data that are stored in an encrypted form in the cloud, one may need to download all encrypted data, and then decrypts and searches them. However, it is not efficient neither convenient especially with huge encrypted data or a resource constraint devices. Alternatively, the user may require sending his private key to the cloud server to perform the decryption and searching procedures on his behalf. However, sending the private key to the cloud server may cause serious issues with data files integrity and secrecy [9–13]. Therefore, to ensure the privacy of the outsourced data, different searchable encryption-based systems have been proposed. These searchable encryption-based systems entail encrypting the data by the data owner before outsourcing it to the cloud with the ability to search and retrieve relevant data through a keyword search or ranked keyword search techniques. These searchable encryption schemes can be divided into three categories: Symmetric-key based techniques, Fuzzy-searchable based techniques, and Public-key based techniques as portrayed in Fig. 2.3.

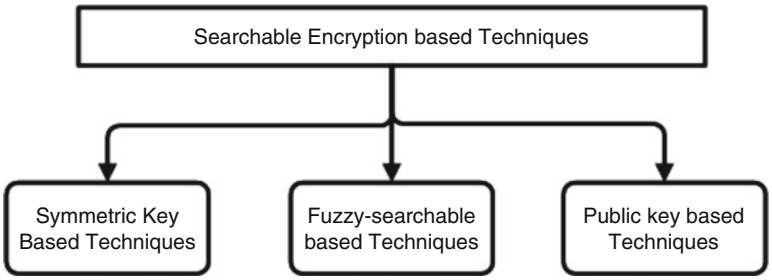

Fig. 2.3 Taxonomy of searchable encryption-based techniques

3.1 Symmetric-Key Based Techniques

The symmetric-key encryption system allows a data owner to outsource his data, encrypted with a symmetric encryption-based techniques (i.e., stream cipher), to untrusted locations over the cloud. The encrypted outsourced data are still searchable for relevant files by means of a trapdoor (i.e., a keyword) that is generated via the data owner private key. The generated trapdoor will be transferred to the server to search for a matched encrypted data with the trapdoor. In this regards, Song, Wagner [14] introduced an encryption and a searching technique over encrypted data with sequential scanning. The authors construct a special two-layered encryption technique that allows searching over cipher-texts without disclosing any sensitive information to the server. The authors proposed to encrypt each word separately assuming that each word has the same length, and then compute the bitwise exclusive or (XOR) with a special sequence of pseudorandom bits inside the plain text. To carry out the search, the data owner must create a private key (k_i) that is corresponded to the locations of the searched word (W_i). The generated private key is then XORed with the cipher-text ($C_i \oplus W_i$) to extract a corresponding structure that is in the form $\langle s, F_{ki}(s) \rangle$ where (s) is some pseudorandom sequence values generated using some stream cipher, and $F_{ki}(s)$ is a pseudorandom function. In this technique, the complexity of encrypting the data and searching for a specific keyword over the encrypted data increases at most linearly with the size of the files collection and the data length. For instance, for a document of length (n) words, the encryption and the searching algorithms require O(n) stream and block cipher operations. However, the proposed technique leaks important information about the documents using any statistical techniques. To handle the variable length words, Goh [15] developed a semantic secure indexes model to prevent leaking any sensitive or statistical information of the outsourced documents against adaptive chosen keyword attacks. The proposed model constructs an index for each document based on pseudo-random functions used as hash functions, and Bloom filters (BF) as a document word index. The word in this model is represented in an index by a codeword for each document which is derived through applying the pseudorandom function once with the word as input and another with a unique document identifier. The non-standard use of the pseudorandom function is to prevent correlation attacks. To search over encrypted documents for the word (y), the user should compute the trapdoor $T_y \leftarrow \text{Trapdoor}\left(K_{\text{priv}}, I_{D_i}\right)$ for the word (y), where (K_{priv}) is the master private key, (D_i) is a unique document identifier, and I_{D_i} is the index for each document (D_i). The trapdoor (T_y) is then send to the server where the encrypted documents and the corresponding BF index $I_{D_i} = (D_i, \text{BF})$ existed. The server tests for a match with the documents through the function SearchIndex $\left(T_y, I_{D_i}\right)$. The BF is represented as an array of bits initially set to 0, and a set of hash functions to mark a set element as 1 of some array positions. To verify if an element belongs to the BF array, the hash values for this element are computed to identify the corresponding array positions. If any of the bits at these positions is 0, then the element is not in the set. This technique provides O (1) search time complexity per document and

can handle variable length words. However, this scheme only supports exact match queries.

Similarly, Chang and Mitzenmacher [16] built a dictionary-based keyword index for each document based on pseudo-random functions. The authors aim to mask a dictionary keyword index for each file using pseudo-random bits to be kept at a remote server. On the other hand, the users can easily retrieve certain files using a short seed that enables the server to unmask selective parts of the index. For each file, an index is created as a set of linked lists, each linked list is associated with a list of keywords in the dictionary of the corresponding file. Initially, all values are set to 0, then if the document m_j contains the keyword w_i, its index position $I_j[P_s(i)]$ is set to 1. The users compute a secrete value r_i using a mapping function F where, $r_i = F_r(i)$, $i \in [2^d]$. For each document, a masking index string M_j is created through a document mapping function G, such that $M_j[i] = I_j[i] \oplus G_{ri}(j)$. The documents are then encrypted using an encryption algorithm and the encrypted documents are outsourced to the cloud along with the corresponding index mask string M_j. Two secrets keys (s) and (r) along with the dictionary are kept at the user's device. Since the authors presume that the data owners are using mobile devices with limited bandwidth and storage space, their solution incur minimum overhead in terms of bandwidth and storage. The search time for this approach is O(n/p), where (n) is the size of the documents collection and (p) is the number of cores. However, this scheme supports an exact single keyword match queries.

To improve the efficiency and the security to a higher degree compared to the previous schemes and to support multi-user environments, Curtmola, Garay [17] proposed a searchable broadcast encryption scheme. The proposed searchable symmetric encryption (SSE-1 and SSE-2) is based on an index per document. The user that owns the data can grant/revoke privileges to authorized users to access/query the outsourced data. In this schema, the proposed index has an array that holds a collection of linked list for documents identifier containing a keyword $D(w_i)$ in an encrypted form and a look-up table to trace and decrypt the first elements of each list in the array. The nodes of the linked list L_i are the document identifiers $D(w_i)$ that contain the keyword w_i. The array locations are the nodes of all $L_i$in a scrambled way. The lookup table (T) entries on the other hand are the keywords w_i index in the array and the decryption key of the first element in L_i. Both the array and the lookup table are encrypted and kept at the server along with the encrypted files. However, if a position in the array is known along with first node encryption key, one can trace and decrypt the other nodes of L_i which correspond to the document identifiers $D(w_i)$. In this schema, the server complexity is constant per document with the searched word, and the overall complexity for each query is proportional to the number of documents that have the searched single word. The computation and the storage complexity at the user side is O(1) and the search time for the server is optimal, but the update of the index is inefficient. Similarly, Chase and Kamara [18] considered stronger security definitions to produce schema that is efficient, associative, and adaptively secure in structured data. The authors of this schema proposed an encryption model for structured data like social networks, images, maps, location information, etc. and, at the same time, the proposed structured

data can be privately queried. The focus of this scheme is to build a structured encryption algorithm that is searchable using specific query token if the secrete key is known. The structured data encryption algorithm operates over a labeled data that has a label (L) and a sequence of data items (m) (i.e., connecting a set of keywords to each data item). For each keyword (w), an array is initiated to hold a pointer j from the pseudo-random permutation set $G_K(L(w))$ and the semi-private item v_i. In this schema, the dictionary was implemented based on hash tables which makes this schema yields an optimal search time O(|I|). However, the encrypted index can be very large. Similarly, van Liesdonk et al. [19] proposed a schema to deal with adaptive security based on one index per keyword to support efficient search and updates of the documents stored at a CSP server. Their proposed scheme converts each distinct keyword into a searchable representation of the form $S_W = (f_{kf}(w), m(I_w), R(w))$ that can be tracked by the trapdoor $T_w = \left(f_{k_f}(w), R'(w)\right)$ with the ability to efficiently update the searchable representation whenever needed. $f_{k_f}(w)$ is a pseudorandom function that identifies S_W, $m(I_w)$ is a masking function for the collection of documents IDs that contains the keyword (w), $R(w)$ and $R'(w)$ are the associated unmasking functions. In case $f_{k_f}(w)$ is found, the server sends back the encrypted data items with the matched IDs in I_w to the client. Even though this schema uses only a simple primitive like pseudo-random functions, but it still obliges for two rounds of communication to generate, update the index, and to search for the documents. Finally, the proposed schema may produce a very large encrypted index. Kurosawa and Ohtaki [20] proposed a schema that is slightly stronger than Curtmola et al. [17]. They proposed a verifiable searchable symmetric encryption scheme that is universally composable (i.e., Protocols security is preserved even if arbitrarily composed with other instances of the same or other protocols) [21] and reliable against active adversaries or malicious servers. They address the issue of an active adversary who might forge the encrypted files to make the retrieving of the files incorrect. The proposed schema is translated to a client/server protocol. The protocol has two phases: (1) the store phase which is executed once by the client to compute $(I, C) \leftarrow \text{Enc}(\text{Gen}(1^k), D, W)$, where I is an encrypted index of the keywords W, C is the encrypted documents D, and the $\text{Gen}(1^k)$ is the secrete key. (2) The search phase which is executed many times by the server to compute $(C(w), \text{Tag}) \leftarrow \text{Search}(I, C, \text{Trpdr}(K, w))$, where $C(w)$ is a ciphertext of D, $t(w) \leftarrow \text{Trpdr}(K, w)$ is a trapdoor generated by the client in response to a keyword w query and Tag is $MAC(K, m)$ a tag generation algorithm for a message m encrypted using the key K. If the client receives $\left(\tilde{C}(w), \text{Tag}\right)$ from the server, the client verifies the validity of the received Tags the Tag Accept/Reject $\leftarrow$ Verify $\left(K, \text{Trpdr}(K, w), \tilde{C}(w), \text{Tag}\right)$. The client decrypts the files if the verification functions returns accept. The proposed scheme consists of six polynomial time algorithms and requires a linear searching time, but supports only single-keyword search.

None of the previous schemes is explicitly dynamic with the ability to add, delete, and update files efficiently. Therefore, Kamara and Papamanthou [22] proposed to extend the inverted index approach proposed in Curtmola and Garay [17] and construct a new sublinear-time schema that is secure against adaptive chosen

keyword attacks. The proposed schema has reduced index sizes with the ability to add/delete files efficiently. Therefore, they added three extra encrypted data structure, namely search array, search table (i.e., dictionary), and a deletion array that can be used by the server to monitor the search array positions in case of an update. They used a homomorphic encryption scheme to encrypt the node's pointers. To modify the pointer without ever having to decrypt the node, they used a private-key encryption scheme which consists of XORing the message with two pseudo-random functions. Finally, they added a free list that can be used by the server to determine the free locations to add new files. The proposed dynamic index-based schemes are a tuple of nine polynomial-time algorithms. The client generates a secret key $K \leftarrow Gen(1^k)$ to be used for the files (D) encryption to produce an encrypted index I and a sequence of ciphertexts C $(I, C) \leftarrow (K, D)$. In order for the client to search for a keyword, the client builds a search token $\tau_s \leftarrow SrchToken(K, w)$. The client can also request to add or delete a file (f) through generating add $(\tau_a, C_f) \leftarrow AddToken(K, f)$ or delete $\tau_a \leftarrow DelToken(K, f)$ tokens. The clients also can issue a search request $I_w \leftarrow Search(I, C, \tau_s)$ with the encrypted index I, a sequence of ciphertexts C and a search token τ_s to retrieve a sequence of files identifiers $I_w \subset C$. In this schema, the searching time for the server is linear (by using a hash table) which is optimal, but this approach is very complex and difficult to implement.

Moreover, the search procedure cannot be parallelized on the server because they represent a T-set as a linked list. As a result, Kamara and Papamanthou [23] improved the efficiency further through proposing a new dynamic and highly parallelizable sub-linear searchable symmetric encryption scheme based on the multi-core architectures. In this schema, they used a new tree-based multi-map data structure which they call a keyword red-black tree (KRB). The KRB tree is a dynamic data structure that is similar to an inverted index but can be used to answer multi-map queries efficiently. The KRB allows both keyword-based search and file-based search operations. This schema is useful for handling updates efficiently. The parallel search is executed similar to the binary trees, where the first processor searches for a specific keyword at the root of the tree. The tree will be divided into two sub-trees, the first processor continues with one sub-tree while another processor is assigned to the other sub-tree. The set of keywords are kept in a keyword hash table as a tuple (key, value) with a key of exponential size and the value is an encryption of a Boolean value. This approach yields very efficient schemes in less than the optimal sequential search time, and allows efficient updates, but this scheme is designed only for single keyword Boolean search, that means whether or not the keyword exists. A complete comparison of all the schemes can be found in Table 2.3 and Table 2.4.

Although these searchable symmetric encryption techniques allow a user to search securely over encrypted data through keywords, the main disadvantage with these techniques is that they support only exact keyword searches. Consequently, this reduces the system efficiency because the search complexity will be the number of distinct keywords in the document collection. Another approach to solve such problems are the Fuzzy-Searchable Encryption based systems.

Table 2.3 Comparison of several symmetric-key encryption schemes

Scheme	Dynamism	Search time	Index size
Song et al. [14]	Static	O(n/p)	N/A
Goh [15]	Dynamic	O(n/p)	O(n)
Chang and Mitzenmacher [16]	Static	O(n)	O(mn)
Curtmola et al. [17]	Static	O(r)	O(m + n)
van Liesdonk et al. [19]	Dynamic	O(r)	O(mn)
Chase and Kamara [18]	Static	O(r)	O(mn)
Kurosawa and Ohtaki [20]	Static	O(n)	O(mn)
Kamara et al. [22]	Dynamic	O(r)	O(m + n)
Kamara and Papamanthou [23]	Dynamic	O((r/p) log n)	O(mn)

Where n is the size of the document collection, r the number of documents containing keyword **w**, m the size of the keywords space, and p the number of cores

Table 2.4 Comparison of several symmetric-key encryption schemes

Scheme	Description	Main drawbacks
Song et al. [14]	A technique for searching in encrypted data with sequential scanning by using a special two-layered encryption construct that allows searching the cipher-texts	It leaks important information about the documents using statistical techniques, and only works with words of the same length
Goh [15]	An efficient secure index construction based on pseudo-random functions and Bloom filters	Supports only exact match queries
Chang and Mitzenmacher [16]	A dictionary-based keyword index for each document based on pseudo-random functions	Supports only exact match queries
Curtmola et al. [17]	A solution for the multi-user problem based on broadcast encryption	Updates to the index are inefficient
Kamara et al. [22]	An efficient, associative, and adaptively secure schema based on creating a model for structured data	The encrypted index can be very large
van Liesdonk et al. [19]	Two schemes based on one index per keyword to support efficient search and updates of the database	The encrypted index can be very large
Kurosawa and Ohtaki [20]	A verifiable searchable symmetric encryption scheme that is universally composable	Supports only single-keyword search
Kamara et al. [22]	A new schema that achieves the properties based on the inverted index approach [17]	Complex and difficult to implement
Kamara and Papamanthou [23]	A new dynamic and sub-linear searchable symmetric encryption scheme that is highly parallelizable based on the multi-core architectures	Single keyword Boolean search

3.2 *Fuzzy-Searchable Encryption*

Fuzzy keyword search returns the matching files to the users' searching inputs that even matched exactly to a set of predefined keywords or the closest possible matching files based on keyword similarity semantics, because fuzzy keyword search can tolerate minor typos and formatting inconsistencies [24]. In this regards, Adjedj et al. [25] described a way to solve the issue of preserving privacy in a biometric identification system using a fuzzy search scheme. They used symmetric searchable encryption (SSE) which allows a client to encrypt the data in such a way that these data can still be searched to achieve reasonable computational costs for each identification request. In this schema, they combined SSE and locality-sensitive hashing (LSH). The main purpose of using LSH is to make outputs the same result for near points and a different result for distant points by using a matching algorithm which computes a similarity score between the two points. By using SSE architecture, the secret keys are stored on the client side but not on the database side (i.e., server side stores the encrypted data without secret keys). This will ensure the privacy of the stored data, but it is unsuitable for many applications, such as when data are frequently updated or streaming.

In an attempt to tolerate minor typos and formatting inconsistencies, Li et al. [24] realized that depending on a spell checker mechanism does not address the problem (i.e., mistyped words or two valid words typed interchangeably) due to the extra communication cost with the users to identify the correct words. Therefore, they proposed the first solution for effective fuzzy keyword search over encrypted cloud data. They constructed a wildcard-based fuzzy set $S_{w_i,d} = \left\{S'_{w_i,0}, S'_{w_i,1}, \ldots, S'_{w_i,d}\right\}$ with edit distance d for each keyword $w_i \in W$ before building the index. The $S'_{w_i,\tau}$ denotes the set of words w'_i with τ wildcards representing the edit operations on $w_i \in W$. This technique can deal with minor typo errors when users type in query keywords through using the edit distance to quantify keyword similarity through semantic keyword with edit distance $d = 1$ from w_i. That is, all the words that are satisfying the similarity criteria $ed\left(w_i, w'_i\right) \leq d$ are listed. The index $\left\{\left(\left\{T_{w'_i}\right\} w'_i \in S_{w_i,d}, Enc\left(sk, FID_{w_i} \middle\| w_i\right)\right)\right\} w_i \in W$ with the set of encrypted files IDs (FID_{w_i}) that contain the keyword w_i is built and a trapdoor set $\left\{T_{w'_i}\right\}$ is computed for each word$w' \in S_{w_i,d}$. The index and the encrypted files are then outsourced to the cloud server for storage. The secret key sk is shared between the data owner and authorized users. To search for a keyword w with a private key k, the authorized user computes the trapdoor set $\left\{T_{w'_i}\right\} w' \in S_{w,k}$ and send to the server. The server then compares the request with the index table and returns all possible encrypted file identifiers $\left\{Enc\left(sk, FID_{w_i} \middle\| w_i\right)\right\}$. The size of the index $S_{w_i,d}$ with a keyword length of l and edit distance of d is $O(l^d)$. This schema is secure and privacy preserving, but it is only applicable to strings under edit distance, and fuzzy sets may become too big with longer words, which necessitates issuing large trapdoors sets. Therefore, Kuzu et al. [26] described an efficient similarity search over the

encrypted data based on the locality sensitive hashing (LSH) which is the nearest neighbor algorithm for index creation and the bloom filter (BF) for translation of strings, to provide a more generic solution and to utilize the distinct similarity search contexts. Similar features are put into one bucket with high probability due to the property of LSH while not similar features are kept into different buckets. This schema embeds the query string into the BF and represented as a set of n-grams. Each n-gram is then subject to a hash function and the corresponding bit locations are set to 1. They use a publicly available typo-generator which produces a variety of spelling errors to check if the keywords contain typographical errors, and to measure the Jaccard distance between the encodings of the original and perturbed versions, to determine distance thresholds for their Fuzzy Search scheme. In this schema, one round is needed for a limited number of data items with large set of features, and two rounds are needed if the number of data items is huge, but it introduce a certain degree of false positive rate in the searching results.

However, a semi-honest-but-curious cloud server might save its computation or download bandwidth through executing only a fraction of the search operation honestly and return a fraction of the search results honestly as well. Therefore, a verifiable scheme is needed to ensure that the user can verify the correctness and the completeness of the search results. In this regards, Wang et al. [27] proposed a new efficient and verifiable fuzzy keyword search (VFKS) scheme over the encrypted data in cloud computing to return the closest possible results based on similarity semantics. They use a wildcard-based fuzzy keyword set and the BF to enable a fuzzy keyword search over encrypted data and maintain keyword privacy and the verifiability of the search result. Their approach consists of the algorithms (Keygen, Buildindex, trapdoor, search, Verify). In which the Keygen algorithm $(\mathrm{sk}, \mathrm{sk}') \leftarrow (\mathrm{Keygen}(1^{\mathrm{k}}))$ executed by the data owner with a security parameter k to produce the secrete key (sk) to generate the index and the document encryption key (sk') used to decrypt the document. The Buildindex algorithm $G_W \leftarrow \mathrm{Buildindex}(\mathrm{sk}, W)$ executed by the data owner to create the index $\mathrm{G_W}$, i.e., a symbol-based tree using the secrete key (sk) and the distinct keyword set of the documents collection D.

The symbol-based index tree $\mathrm{G_W}$ and the encrypted documents are outsourced to the cloud server. The user can generate a trapdoor set $\{\mathrm{T}_{\omega'}\}\,\omega' \in \mathrm{S}_{\omega,\mathrm{d}} \leftarrow \mathrm{trapdoor}\left(\mathrm{sk}, \mathrm{S}_{\omega,\mathrm{d}}\right)$ for all wildcard-based fuzzy keywords $\mathrm{S}_{\omega_i,\mathrm{d}} = \left\{\mathrm{S}'_{\omega_i,0}, \mathrm{S}'_{\omega_i,1}, \ldots, \mathrm{S}'_{\omega_i,\mathrm{d}}\right\}$ of the keyword ω' with edit distance $\mathrm{ed}(\omega, \omega') < \mathrm{d}$. The server executes the search algorithm $(\mathrm{flag}, \mathrm{ID}_\omega, \mathrm{proof}) \leftarrow \mathrm{Search}\,(G_W, \{T_{\omega'}\})$ upon receiving the user trapdoor set $\{\mathrm{T}_{\omega'}\}$ to search for the document with keyword ω and return the document identifier ID_ω, true and a proof if document existed otherwise false, and a proof. The user executes $(\mathrm{true/false}) \leftarrow \mathrm{Verify}(T_\omega, (\mathrm{flag}, \mathrm{ID}_\omega, \mathrm{proof}))$ to verify whether the server is honest or not over the search result $(\mathrm{flag}, \mathrm{ID}_\omega, \mathrm{proof})$ and outputs true if the server honestly search, otherwise false is returned. They utilized the well-known multi-way tree to store the fuzzy keyword set over a predefined symbol set, which might grow in size if the keyword length is huge. This schema is secure and privacy preserving, while supporting efficient verifiability

Table 2.5 Comparison of several Fuzzy-searchable encryption schemes

Scheme	Description	Main drawbacks
Adjedj et al. [25]	A way of solving the issue of preserving privacy in a biometric identification system based on a fuzzy search scheme	Unsuitable for many applications when data are frequently updated or streaming
Li et al. [24]	The first solution for effective fuzzy keyword search over encrypted cloud data based on a fuzzy set for the keywords before building the index	Has a long word which necessitates performing large trapdoors
Kuzu et al. [26]	An efficient similarity search over the encrypted data based on LSH and BF	Introduces a certain degree of false positive rate in the searching
Lu [28]	A privacy-preserving search logarithm over the encrypted data to support a range of queries based on Logarithmic Search on Encrypted Data	The indexing information makes it as vulnerable as order-preserving encryption
Wang et al. [27]	A new efficient and verifiable fuzzy keyword search based on the method of wildcard-based fuzzy keyword set and the BF	The same key is used to encrypt and decrypt the data

of the searching result. However, this schema focuses on key word search but does not consider a phrase search. Moreover, the index generation is handled by the data owner, which means that the owner might abandon the exact keyword index constructed before and generate a specialized fuzzy-keyword index for fuzzy search, hence wasting much more computation and storage resources (Table 2.5).

All these previous techniques are based on symmetric key encryption, in which the same key is used to encrypt and decrypt the data. To enable an authorized user to access the encrypted data, the data owner must share this key. By sharing this key, unauthorized users can also use this key to access the encrypted data.

3.3 Public-Key Encryption

A searchable symmetric key-based encryption schema are valid for users owning the data and wish to upload it to a third-party and untrusted server (i.e., cloud server). On the other hand, there are cases when the outsourced data (medical data, stock quotes, emails, etc.) are public and uploaded by different owners and the user is not aware of it, at the same time, the user wishes to retrieve certain files without revealing to the server which file he wants. The public-key encryption with keyword search is the solution for such cases. The public-key encryption uses two different keys, private and a public key. The private key is given by the data owner to the users and the public key is given to the server in this context as illustrated in Fig. 2.4.

The first searchable encryption scheme using a public key system was proposed in [29]. This scheme can be extended to handle range, subset, and conjunctive

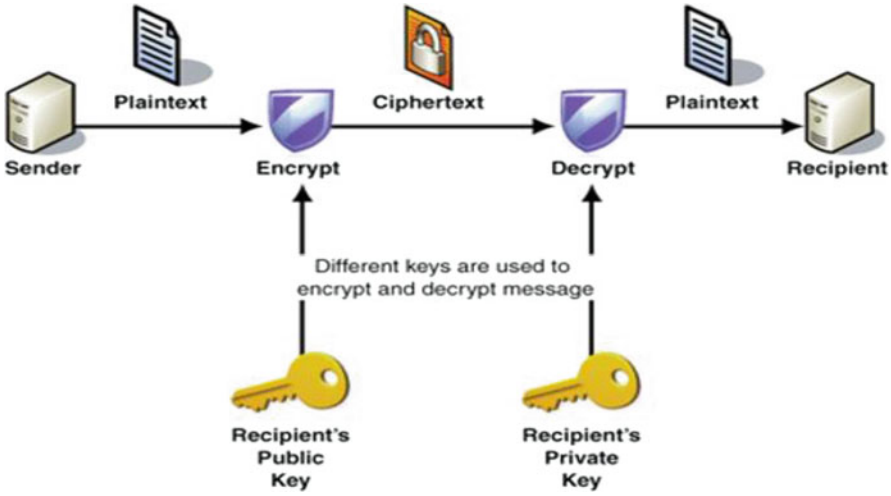

Fig. 2.4 Public-key encryption architecture

queries. It also hides the attributes for messages that match a query. They use identity-based encryption (IBE), in which the keyword acts as the identity. The proposed searchable public-key encryption consists of four polynomial time randomized algorithms (KeyGen, PEKS, Trapdoor, Test). The data owner generates his public/private key pair using the algorithm $(A_{pub}, A_{priv}) \leftarrow \text{KeyGen}(s)$ over a security parameter s. In order to search for any keyword W, the user generates a trapdoor $T_W \leftarrow \text{Trapdoor}(A_{priv}, W)$ using their private key A_{priv} for certain keywords W. The server determines whether a document contains one of the keywords W specified by the users (yes $\| W = W' |$ no $\| W \neq W'$) $\leftarrow \text{Test}(A_{pub}, S, T_W)$ through the received Trapdoor T_W, the given public key A_{pub} and a searchable encryption $S = \text{PEKS}(A_{pub}, W')$. The proposed scheme has two constructions for 12public-key searchable encryption: (1) An efficient construction based on a variant of the Diffie–Hellman (BDH) assumption by building a non-interactive searchable encryption scheme from a bilinear map. They have proved that this scheme is semantically secure against a chosen keyword attack in the random oracle model based on the difficulty of the bilinear Diffie–Hellman problem. (2) A limited construction using any trapdoor permutation, which is less efficient because this construction assumes that, general trapdoor permutations assuming that the total number of keywords the user wishes to search for is bounded by some polynomial function in the security parameter. They can reduce the size of the public file by allowing the user to re-use individual public keys for different keywords. In this schema, the searching time is linear, but Public key solutions are usually computationally expensive. Furthermore, the keyword privacy cannot be protected in the public key setting, since the server could encrypt any keyword with a public key and then use the received trapdoor to evaluate the ciphertext. Finally, the proposed constructions are applicable to searching on a small number of keywords rather than an entire file.

Bellare et al. [30] proposed a deterministic searchable public-key encryption scheme. The main idea in this technique is to associate a tag with a plaintext, which can be computed by the client to form a particular query F(pk, x_1) and by the server from a ciphertext that encrypts it G(pk, c). They can then use this tag (i.e. the output of the polynomial time algorithms F, G) to create a tree-based index that can be used for searching. Since searchable tags are deterministic, the server can organize them in a sorted system and match the minimum logarithmic time. The proposed scheme consists of three polynomial time algorithms AE = (K, E, D). This schema is t-efficiently searchable encryption where t(.) < 1 $\forall$ $x_i \in$ Ptsp(k), Ptsp(k) is the plaintext space and the probability F(pk, x_1) = G(pk, c) = 1 over (pk, sk) $\leftarrow$ K(1^k)andc $\leftarrow$ E(pk, x_1). This technique is a combination of any public-key encryption scheme and any deterministic hash function and so this scheme is secure, but they have left without solution the problem of finding standard model schemes. The issue of this proposed approach is that it only provides privacy to text drawn from a space of large min-entropy.

A range of queries over multiple attributes in the public key settings have been studied in the herein cited study [31]. They proposed an encryption scheme called Multi-dimensional Range Query over Encrypted Data (MRQED) that allows a network gateway to encrypt summaries of network flows before submitting them to the cloud. The proposed scheme was proven with the network audit logs. An authority can release a public key to an auditor to decrypt flows within certain ranges only. The proposed scheme operates over a tuple of flow features (t, a, p) representing the flow timestamp range t $\in$ [t_1, t_2], the flow source address range a $\in$ [a_1, a_2] and the destination flow port number range p $\in$ [p_1, p_2]. Their proposed range queries imply (t $\geq$ t_1) $\wedge$ (a = a_1) $\wedge$ ($p_1 \leq$ p $\leq$ p_2) where all flows (t, a, p) within the defined range can be decrypted with the provided decryption key without revealing the other flow attribute values nor issuing huge number of keys. The proposed schema consists of four polynomial-time algorithms (Setup(k, L_Δ), Encrypt(PK, X, Msg), DeriveKey(PK, DK, B), QeurtyDecrypt (PK, SK, C)) in which the setup algorithm (PK, SK) $\leftarrow$ Setup(k, L_Δ) over a security parameter k and a point in lattice L_Δ (represents a tuple as a point in L_Δ) produces a public key PK and a private key SK. The gateway encrypts the pair (Msg., X) that consists of an arbitrary string representing the entire flow summary and a point X in a multi-dimensional space representing the attributes using the public key PK to produce the ciphertext C $\leftarrow$ Encrypt(PK, X, Msg). The authority derives a decryption key DK $\leftarrow$ DeriveKey(PK, SK, B) for a hyper-rectangle B in L_Δ (i.e., test whether a point X falls inside it) using the public and the private key pair (PK, SK). Finally, an auditor can decrypt (plaintext/null) $\leftarrow$ QeurtyDecrypt(PK, DK, C) relevant flows using the provided key pair (PK, DK) over the retrieved ciphertext C. However, in this schema, each flow is represented as a hyper-rectangle B in L_Δ. This requires issuing one pair of keys for each flow, having a huge number of flows would require a huge number of key pair pools.

Liu et al. [32] proposed an Efficient Privacy Preserving Keyword Search Scheme (EPPKS) in cloud computing, which reduces a client's computational overhead

by allowing the cloud service provider to participate partially in the decipherment process while protecting the data and the queries privacy. The proposed schema does not require a private key transmission; to make it suitable for the cloud environment. This schema consists of the following seven randomized polynomial time algorithms EPPKS = (Keygen, EMBEnc, KWEnc, TCompute, Test, Decrypt, Recovery). The user and the service provider execute the Keygen function to produce public/private key pair. For the user U, he executes $U : (U_{pub}, U_{priv}) \leftarrow \text{Keygen}(k_1)$ over a sufficiently large security parameter k_1 to produce his key pair (U_{pub}, U_{priv}). Similarly, the service provider S executes $S : (S_{pub}, S_{priv}) \leftarrow \text{Keygen}(k_2)$ over a sufficiently large security parameter k_2 to produce his public/private key pair (S_{pub}, S_{priv}). The user encrypts the data using his public key and the service provider private key to produce the message m ciphertext $C_m \leftarrow \text{EMBEnc}(U_{pub}, S_{priv}, m)$. The keywords are also encrypted before outsourcing the data to the service provider using the user public key $C_{w_i} \leftarrow \text{KWEnc}\left(U_{pub}, W_i\right)$. In order to retrieve a file with keywords W_j, the user executes $T_{W_j} \leftarrow \text{TCompute}\left(U_{priv}, W_j\right)$ and sends it to the CSP. The CSP on the other hand executes $\left(W_i \stackrel{?}{=} W_j\right) \leftarrow \text{KWTest}\left(U_{pub}, C_{W_i}, T_{W_j}\right)$ to determine whether a given file has the keyword W_j. An intermediate result C_ρ will be calculated by the CSP before returning the matching file to the user as a result of executing $C_\rho \leftarrow \text{PDecrypt}(S_{priv}, U_{pub}, C_m)$. Upon receiving the files, the user executes $m \leftarrow \text{Re}\, coery(U_{priv}, C_m, C_\rho)$. This schema supports multiple keyword searching on the encrypted data and it is semantically secure, because the service provider could search in the encrypted files efficiently without leaking any information, but there is a big challenge if the user requires the service provider to provide the computational service.

All these schemes achieve good security and privacy but they require high computations and memory of the end-devices during the encryption and decryption process. Moreover, these schemes provide unsearchable encryption, but do not fit well for less powerful client devices, which have only limited bandwidth, CPU, and memory as discussed in [33]. Table 2.6 consolidates the various public key-based privacy-preserving approaches advantage and their shortcomings.

Among the different available solutions that aim to design operations compatible with data encryptions while preserving the privacy of the data outsourced to the cloud, Searchable Encryption (SE) schemes seem to allow a curious party to carry out searches on encrypted cloud data without having to decrypt it, hence maintaining its privacy. Table 2.7 summarizes the advantages and the disadvantages of the common searchable encryption schemes in cloud computing.

4 Conclusion and Future Work

While data encryption seems to be the right countermeasure to prevent privacy violations, classical encryption mechanisms fall short of meeting the privacy requirements in the cloud setting. Typical cloud storage systems also provide basic operations on stored data such as statistical data analysis, logging and searching

Table 2.6 Comparison of several public-key encryption schemes

Scheme	Description	Main drawbacks
Boneh et al. [29]	The first scheme to use a public key system based on identity-based encryption	Keyword privacy is not protected in the public key setting
Bellare et al. [30]	A deterministic searchable public-key encryption scheme based on associating a tag with a plaintext	Cannot find a standard model schemes
Katz et al. [34]	The first notion of predicate encryption based on IBS, hidden vector encryption (HVE) and attribute-based encryption	This scheme is only proven to be selectively secure and no delegation functionality is provided
Attrapadung and Libert [35]	A protocol based on functional encryption and public key schemes using Inner Product Encryption	Cannot be proven fully secure under some natural assumptions
Liu et al. [33]	A SPKS scheme for cloud storage services based on enabling cloud service providers to participate in the decryption process partially	It may disclose information to CSP to participate in the decryption process

Table 2.7 Comparison of searchable encryption schemes in cloud computing

	Advantages	Disadvantages
Symmetric-key encryption	• The private keys are used in symmetric-key encryption and are resistant to external attacks • Simple to generate keys • Symmetric-key encryption algorithms require low computing power to be created	• The same key is used to encrypt and decrypt the data and by sharing this key, unauthorized users can access the encrypted data • The private key must be exchanged in a secure manner • Every participant must have an identical private key • Symmetric-key encryption supports only exact keyword search
Fuzzy-searchable encryption	Enhances system usability by returning the matching files or the closest possible matching files based on keyword similarity semantics	The same key is used to encrypt and decrypt the data. Through sharing this key, unauthorized users can access the encrypted data
Public-key encryption	The unique private and public keys are provided for each user, which will allow them to perform secure exchanges of information	• Generating the keys is expensive • Public-key encryption algorithms require more computational cost than Symmetric-key encryption

and these operations would not be feasible if the data were encrypted using classical encryption algorithms. Among various solutions aiming at designing operations that would be compatible with data encryption, Searchable Encryption (SE) schemes allow a potentially curious party to perform searches on encrypted data without having to decrypt it. SE seems a suitable approach to solve the data privacy problem

in the cloud setting. A further challenge is raised by SE in the multi-user setting, whereby each user may have access to a set of encrypted data segments stored by a number of different users. Multi-user searchable encryption schemes allow a user to search through several data segments based on some search rights granted by the owners of those segments. Privacy requirements in this setting are manifold, and not only the confidentiality of the data segments but also the privacy of the queries should be ensured against intruders and potentially malicious CSP. Recently, few research efforts came up with multi-user keyword search schemes meeting these privacy requirements, either through some key sharing among users or based on a Trusted Third Party (TTP).

These studies provide limited keyword search functionality for cloud storage services. Thus, service providers must implement a complete secure search scheme to promote their services. This study proposes a scheme for performing ranked multikeyword searches with fault tolerance in cloud storage systems. The proposed scheme uses similar keyword sets to perform a similarity search, and a secure k-nearest neighbor (kNN) scheme to perform a ranked multikeyword search. Moreover, the proposed scheme is fault tolerant to account for cloud users inputting an incorrect keyword, and still involves performing a file search. When the files are located, they are assigned an associated correlation value.

References

1. Manasrah, A. M., Smadi, T., & ALmomani, A. (2016). A variable service broker routing policy for data center selection in cloud analyst. *Journal of King Saud University-Computer and Information Sciences, 29*(3), 365–377.
2. Zhang, H., et al. (2015). Towards privacy preserving publishing of set-valued data on hybrid cloud. *Cloud Computing, IEEE Transactions on, 99*, 1–1.
3. Wagle, D. M. (2014). Comparative study of privacy preservation and access control of cloud data. *International Journal of Engineering Research & Technology (IJERT), 3*(11), 165–174.
4. Nabeel, M., & Bertino, E. (2014). Privacy preserving delegated access control in public clouds. *Knowledge and Data Engineering, IEEE Transactions on, 26*(9), 2268–2280.
5. AlSudiari, M. A., & Vasista, T. (2012). Cloud computing and privacy regulations: An exploratory study on issues and implications. *Advanced Computing, 3*(2), 159.
6. Seddon, J. J., & Currie, W. L. (2013). Cloud computing and trans-border health data: Unpacking US and EU healthcare regulation and compliance. *Health Policy and Technology, 2*(4), 229–241.
7. Dong, X., et al. (2014). Achieving an effective, scalable and privacy-preserving data sharing service in cloud computing. *Computers & Security, 42*, 151–164.
8. Joseph, N. M., Daniel, E., & Vasanthi, N. (2013). Survey on privacy-preserving methods for storage in cloud computing. In: *Amrita International Conference of Women in Computing*.
9. Jogade, S., Sharma, R., & Kadam, R. (2014). Partitioning data and domain integrity checking for storage-improving cloud storage security using data partitioning technique. *International Journal of Emerging Research in Management & Technology, 3*(3), 133–137.
10. Chen, F., & Liu, A. X. (2014). Privacy and integrity preserving multi-dimensional range queries for cloud computing. In *Networking Conference, 2014 IFIP*. IEEE.
11. Ku, W.-S., et al. (2013). A query integrity assurance scheme for accessing outsourced spatial databases. *GeoInformatica, 17*(1), 97–124.

12. Hu, L., et al. (2013). Spatial query integrity with Voronoi neighbors. *Knowledge and Data Engineering, IEEE Transactions on, 25*(4), 863–876.
13. Naruchitparames, J., & Güneş, M. H. (2011). Enhancing data privacy and integrity in the cloud. In *High Performance Computing and Simulation (HPCS), 2011 International Conference on*. IEEE.
14. Song, D. X., Wagner, D., & Perrig, A. (2000). Practical techniques for searches on encrypted data. In *Security and Privacy, 2000. S&P 2000. Proceedings 2000 IEEE Symposium on*. IEEE.
15. Goh, E.-J. (2003). *Secure indexes for efficient searching on encrypted compressed data* (Technical report 2003/216, Cryptology ePrint archive, 2003). http://eprint.iacr.org/2003/216
16. Chang, Y.-C., & Mitzenmacher, M. (2005). Privacy preserving keyword searches on remote encrypted data. In *Applied cryptography and network security*. Berlin: Springer.
17. Curtmola, R., et al. (2006) Searchable symmetric encryption: Improved definitions and efficient constructions. In *Proceedings of the 13th ACM Conference on Computer and Communications Security*. ACM.
18. Chase, M., & Kamara, S. (2010). Structured encryption and controlled disclosure. In *Advances in cryptology-ASIACRYPT 2010* (pp. 577–594). Berlin: Springer.
19. van Liesdonk, P., et al. (2010). Computationally efficient searchable symmetric encryption. In *Secure data management* (pp. 87–100). Berlin: Springer.
20. Kurosawa, K., & Ohtaki, Y. (2012). UC-secure searchable symmetric encryption. In *Financial cryptography and data security* (pp. 285–298). Berlin: Springer.
21. Canetti, R. (2001). Universally composable security: A new paradigm for cryptographic protocols. In *Foundations of Computer Science, 2001. Proceedings. 42nd IEEE Symposium on*. IEEE.
22. Kamara, S., Papamanthou, C., & Roeder, T. (2012). Dynamic searchable symmetric encryption. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*. ACM.
23. Kamara, S., & Papamanthou, C. (2013). Parallel and dynamic searchable symmetric encryption. In *Financial cryptography and data security* (pp. 258–274). Berlin: Springer.
24. Li, J., et al. (2010). Fuzzy keyword search over encrypted data in cloud computing. In *INFOCOM, 2010 Proceedings IEEE*. IEEE.
25. Adjedj, M., et al. (2009). Biometric identification over encrypted data made feasible. In *Information systems security* (pp. 86–100). Berlin: Springer.
26. Kuzu, M., Islam, M. S., & Kantarcioglu, M. (2012). Efficient similarity search over encrypted data. In *Data Engineering (ICDE), 2012 IEEE 28th International Conference on*. IEEE.
27. Wang, J., et al. (2012). A new efficient verifiable fuzzy keyword search scheme. *JoWUA, 3*(4), 61–71.
28. Lu, Y. (2012). Privacy-preserving logarithmic-time search on encrypted data in cloud. In *NDSS*.
29. Boneh, D., et al. (2004). Public key encryption with keyword search. In *Advances in cryptology-Eurocrypt 2004*. Berlin: Springer.
30. Bellare, M., Boldyreva, A., & O'Neill, A. (2007). Deterministic and efficiently searchable encryption. In *Advances in cryptology-CRYPTO 2007* (pp. 535–552). Berlin: Springer.
31. Shi, E., et al. (2007). Multi-dimensional range query over encrypted data. In *SP'07, IEEE Symposium on*. IEEE.
32. Liu, Q., Wang, G., & Wu, J. (2009). An efficient privacy preserving keyword search scheme in cloud computing. In *Computational Science and Engineering, 2009. CSE'09. International Conference on*. IEEE.
33. Liu, Q., Wang, G., & Wu, J. (2012). Secure and privacy preserving keyword searching for cloud storage services. *Journal of Network and Computer Applications, 35*(3), 927–933.
34. Katz, J., Sahai, A., & Waters, B. (2008). Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *Advances in cryptology–EUROCRYPT 2008* (pp. 146–162). Berlin: Springer.
35. Attrapadung, N., & Libert, B. (2010). Functional encryption for inner product: Achieving constant-size ciphertexts with adaptive security or support for negation. In *Public key cryptography–PKC 2010* (pp. 384–402). Berlin: Springer.