Toward Privacy-Assured and Searchable Cloud Data Storage Services

Ming Li, Utah State University
Shucheng Yu, University of Arkansas at Little Rock
Kui Ren, University at Buffalo
Wenjing Lou and Y. Thomas Hou, Virginia Polytechnic Institute and State University

Abstract

Cloud computing is envisioned as the next generation architecture of IT enterprises, providing convenient remote access to massively scalable data storage and application services. While this outsourced storage and computing paradigm can potentially bring great economical savings for data owners and users, its benefits may not be fully realized due to wide concerns of data owners that their private data may be involuntarily exposed or handled by cloud providers. Although end-to-end encryption techniques have been proposed as promising solutions for secure cloud data storage, a primary challenge toward building a full-fledged cloud data service remains: how to effectively support flexible data utilization services such as search over the data in a privacy-preserving manner. In this article, we identify the system requirements and challenges toward achieving privacy-assured searchable outsourced cloud data services, especially, how to design *usable* and *practically efficient* search schemes for encrypted cloud storage. We present a general methodology for this using searchable encryption techniques, which allows encrypted data to be searched by users without leaking information about the data itself and users' queries. In particular, we discuss three desirable functionalities of usable search operations: supporting result ranking, similarity search, and search over structured data. For each of them, we describe approaches to design efficient privacy-assured searchable encryption schemes, which are based on several recent symmetric-key encryption primitives. We analyze their advantages and limitations, and outline the future challenges that need to be solved to make such secure searchable cloud data service a reality.

The cloud has long been envisioned as the next generation IT architecture, which promises to provide massively scalable data storage and application services to society at a reduced cost, primarily attributed to the centralized management of elastic resources [1]. In this emerging computing platform, the cloud provider, application developers, and end users can all reap benefits. One of the most attractive cloud services nowadays is data storage, where end users outsource large volumes of data to cloud servers to enjoy virtually unlimited hardware/software resources and ubiquitous access, without investing a large amount of capital up front for their own data warehousing and maintenance. Indeed, many well-known cloud service providers (CSPs) have started providing lucrative data storage services during the past few years, including Microsoft SkyDrive, Amazon S3, Dropbox, Apple iCloud, and Google Drive.

Despite the tremendous business and technical advantages of the cloud, the security and privacy concern has been one of the major hurdles preventing its widespread adoption. Especially for outsourced data services, the owners' exclusive control over their data is ultimately relinquished to the CSPs. For example, Google's recent privacy policy implies that they essentially own the right to arbitrarily handle the uploaded user data.[1] As a result, from the data owners' point of view, whenever their outsourced data contain sensitive personal information, such as financial and medical records, and social network profiles, it can no longer be considered as private as before. On the other hand, although in reality CSPs usually enforce data security through mechanisms like firewalls and virtualization, these measures do not fully guard against threats of unauthorized data access from insiders, outsiders, or other cloud tenants due to the non-bug-free deployment and low degree of transparency. Infamous data breach incidents occur from time to time, such as the recent Sony PlayStation data breach[2] and DropBox privacy leakage.[3]

A promising approach for owners to take back control of their data is to adopt end-to-end data encryption (i.e., cryptographic

[1] *http://nvonews.com/2012/04/26/google-drive-owns-everything-you-upload-privacy-policy-concerns-remain/.*

[2] *http://www.businessweek.com/news/2011-05-03/sony-data-breach-exposes-users-to-years-of-identity-theft-risk.html and Dropbox privacy leakage*

[3] *http://www.pcmag.com/article2/0,2817,2387343,00.asp.*

0890-8044/13/$25.00 © 2013 IEEE

cloud storage), which has been investigated by numerous researchers recently [8]. However, while data encryption guarantees data confidentiality, it also rules out many routine manipulations over the data necessary in the plaintext domain. One fundamental requirement is to be able to perform search operations that can sort out relevant information from huge amounts of data. For enterprise end users, database search is an everyday operation that underlies their corporate business intelligence. Individual cloud users such as mobile subscribers would also like convenient and intelligent services that help them with daily activity planning, which heavily involves query and answer. Therefore, to build a full-fledged cloud data service, it is highly desirable to enable privacy-assured search over encrypted data, which ideally does not leak any sensitive user information to the cloud, such as business secrets or private personal activities. Without being able to effectively utilize the outsourced data, the cloud will merely be a remote storage with limited values.

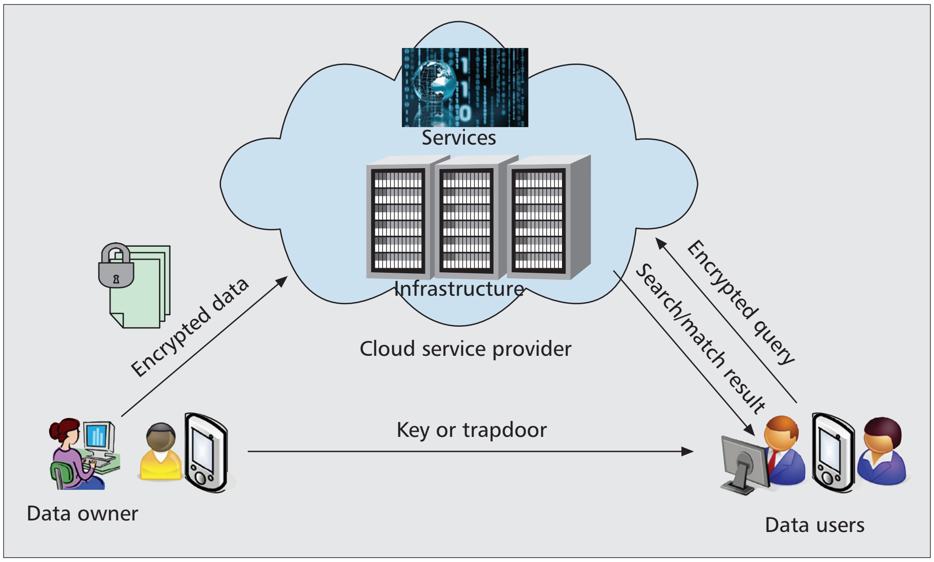

Figure 1. *System architecture for searchable cloud data storage services.*

Over the past decade, searchable encryption (SE) techniques have become a significant research area [6, 11], which are tailored cryptographic solutions addressing privacy-assured search over encrypted data under different system requirements and security models. However, they are still far from feasible to apply in real cloud data services due to insufficient realization of two key properties:

- *Functional usability*: Most existing SE schemes can only deal with Boolean keyword searches, where queries are expressed by Boolean formulas and encrypted documents that satisfy the formula are returned. Such search operations are still quite restrictive, and are unlikely to have wide-scale applications alone.
- *Efficiency*: Schemes with rich search functionalities often require public key cryptography operations, which makes searchable encryption a burden to existing cloud storage in terms of both computation/bandwidth demand and access latency.

In this article, we advocate that it is possible for *functional usability and efficiency* to be simultaneously achieved in order to build privacy-assured searchable cloud data services. We start by identifying an important set of desirable properties including both privacy goals and search usability. Then we present a general methodology for constructing privacy-assured search schemes based on several building blocks, including recently developed efficient symmetric-key encryption primitives (e.g., symmetric searchable encryption [SSE]). For each of the proposed usable search functionalities, we survey recent research advances, and give insights on the advantages and limitations of each approach. Ending with a set of future challenges, this article intends to bring attention to and motivate further research on enabling privacy-assured searchable cloud storage a reality.

Privacy-Assured Searchable Cloud Storage Architecture

Problem Statement

We begin by describing a general cloud data storage service architecture involving three (types of) entities (Fig. 1). The *data owner* (or data contributor) is one or multiple entities who generate and encrypt data, and upload them to the cloud server. The owner can be either an organization or an individual. The *cloud server* belonging to a CSP possesses significant storage and computation resources, and provides them to end users in a pay-per-user manner. There are one or more *data users* in the system, who may need to perform queries over the outsourced data in order to extract useful information. In addition, in order to enable public auditing, a third party auditor can be employed, which is discussed in [13] and is outside the scope of this article. The owner's data are encrypted end-to-end using secret keys created by him/herself, and a searchable index is usually created and encrypted along with the outsourced data. To allow data access and search by users, the data owner usually generates and distributes search tokens (or trapdoors), which are encrypted queries to users, either actively or upon users' requests. When a user wants to gain file access or initiate a query, he/she submits a corresponding token to the server, who then returns a matching set of documents in an encrypted format. In some situations, the data user and data owner can be the same physical entity.

Within the scope of this article, we focus on how to enable privacy-assured search for cloud data services. The above system architecture captures a wide range of searchable cloud data storage applications. In some scenarios, the data owner and user can be the same person; for example, Alice uploads her personal albums to Dropbox and wants to search for a particular photo afterward. Or if we consider a corporate data owner, a company may outsource its business records to a cloud server to enjoy the low-cost storage. At the same time, an employee in the auditing department may need to search the business database for records containing sensitive activities. Alternatively, the data owner may be an individual while the user can be a company. For instance, consider a pervasive healthcare application where each patient uploads her health monitoring data periodically to a third party medical service. The latter operates in the cloud, and will provide health reports to the patient by evaluating the patient's data using some flexible diagnostic criteria.

We consider the cloud server to be semi-trusted. This means that most of the time it behaves properly and does not deviate from the protocol, but it will try to find out as much private information in the stored data as possible. This assumption considers data exposure threats from both insiders and outsiders, and is in line with the current technology trend and business model. For insider threats, there may be curious employees inside the CSP who access user data for their own benefit. Thus, the CSP and data owners are not assumed to be in the same trust domain. In addition, some users may also try to access/utilize the data beyond their privileges, either individually or in collusion with each other.

Depending on the information available to the adversary, a basic threat model (known ciphertext, KC) is to assume that the cloud server only possesses the encrypted data and searchable index. In a stronger known background (KB) model, the cloud server may possess some statistical information about the outsourced dataset (e.g., the distributions of term frequency and document frequency). Under this model, the security guarantee is often termed *as-strong-as-possible*.

System Requirements

Next, we sketch a set of desirable system design requirements from both the functionality and privacy aspects.

Functional Properties — For data search, perhaps the most important property is *usability*, which is the basis for attracting customers. The current Google search is a great example of what is necessary in plaintext domain search. The following is an (incomplete, but typical) list of them:

- *Multi-keyword search*: The search condition should support Boolean expressions consisting of combinations of multiple keywords, including conjunctive normal form (CNF) and disjunctive normal form (DNF).
- *Result ranking*: The ranked search function greatly enhances the relevance of returned search results and reduces communication overhead, which is highly desirable for building usable cloud data services.
- *Error tolerance*: To accommodate various typos, representation inconsistencies, and so forth, search schemes should have a fuzzy nature. This means a search needs to also return relevant results for keywords within a certain edit distance from the input query.
- *Handle structured data*: A large portion of today's online data is represented using rich structures beyond simple text-form, such as social network graphs. Without being able to utilize those structured data, the economic potential of cloud services will not be fully realized.

We note that in the encrypted domain, it is very difficult for the above properties to be simultaneously achieved. We describe how the state-of-the-art schemes achieve some combination of them.

Privacy Assurance — In a searchable cloud storage service, both the owner's outsourced data and users' queries over those data may contain sensitive information and need protection against an adversary. More specifically, the system should meet the following privacy requirements:

- *Data and index confidentiality*: Without the secret key K, no one, including the cloud server, should be able to learn sensitive information from the owner's private data. Similarly, they should not be able to deduce sensitive information underlying the data index, because the index is often closely related to the data itself.
- *Query confidentiality*: Users' most important concern is to hide the search criteria on which they are evaluating the data (e.g., their query keywords). These should not be derivable from the search trapdoor and data/index sent to the cloud server, even when the server possesses some additional background information such as keyword distribution.
- A higher-level requirement is *query unlinkability*, that is, the cloud server shall not learn whether two queries have the same criteria. Note that this intrinsically requires the trapdoor to be non-deterministic.

Efficiency — A privacy-assured data search scheme should have low computation, communication, and storage overheads. For such a scheme to be deployed in a large-scale cloud storage system with economic practicality, we argue that the search process should be completed within both constant communication round and computation time (independent of the database size). In general, the privacy guarantee conflicts with efficiency and functionality goals. For example, it is more private to prevent the cloud server from learning the access pattern (i.e., the sequence of returned data). However, current techniques that protect it, such as oblivious RAM and private information retrieval (PIR), are still far from practical. The former requires logarithmic rounds, while the latter must "touch" the whole dataset outsourced to the server. Thus, we do not discuss them in this article, although those studies are of independent interest.

Toward Designing Privacy-Assured Search Schemes

Related Techniques

First, we briefly discuss and compare several existing techniques, and their relevance to the privacy-assured cloud-based search problem.

- *Secure multiparty computation (SMC)*: In SMC, each party P_i possess some private input x_i, and every party computes some (public) function $f(x_1, \ldots, x_n)$ without revealing x_i to others, except what can be derived from the input and output.
- *Private information retrieval (PIR)*: PIR involves two parties: a client and a server. In asymmetric PIR, the server hosts a public database $\mathcal{D}$, while the client retrieves a record i from $\mathcal{D}$ without revealing i to the server. In symmetric PIR (a.k.a. oblivious transfer), the non-retrieved records should also be withheld from the client, which can be regarded as a special case of SMC.
- *Searchable encryption (SE)*: SE also involves a client and a server, where the latter stores an encrypted database $\widetilde{\mathcal{D}}$, and the former possesses a private query Q that wants to obtain the query result $Q(\mathcal{D})$ without revealing both Q and plaintext $\mathcal{D}$ to the server.

The above also represent different computation models. In the SMC and PIR models, the computation is usually split among the parties, and each party maintains some (private) input. However, in SE, the computation is mainly carried out by the server, and the server does not possess any input. Thus, it can be seen that SE is the technique most relevant to the problem defined in this article.

Methodology and Building Blocks

Next, we introduce the methodology and briefly describe the main building blocks toward designing usable and efficient privacy-assured search schemes.

Methodology — We describe a top-down approach in Fig. 2. Given a search functionality in the plaintext domain, one can decompose it into a certain data index structure and primitive data operations using relevant information retrieval (IR) principles. Then we can try to find a proper encryption scheme to encrypt the data while simultaneously allowing data operations. The second step is nontrivial. Although efficiency could be improved due to the adopted index structure, care needs to be taken to prevent leakage of private information to the cloud server, especially when the server possesses background information on the query statistics. As a result, sometimes the encryption primitive itself may need to be adapted to meet privacy requirements. To illustrate this approach, in Table 1 we compare several existing schemes representing different design spaces in terms of search functions, security, IR method, and so on. We elaborate how they fit into this methodology in what follows.

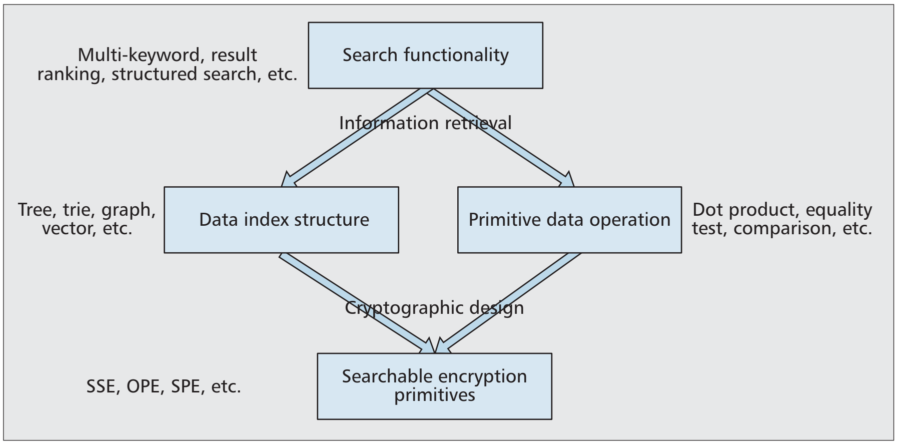
Multi-keyword, result ranking, structured search, etc.
Search functionality
Information retrieval
Tree, trie, graph, vector, etc.
Data index structure
Primitive data operation
Dot product, equality test, comparison, etc.
Cryptographic design
SSE, OPE, SPE, etc.
Searchable encryption primitives

Figure 2. *Top-down methodology for designing privacy-assured search schemes.*

Symmetric Searchable Encryption — Curtmola *et al.* proposed SSE, which is a deterministic symmetric key encryption scheme with security guarantees under rigorous definitions [6]. The SSE scheme is based on the inverted index and uses pseudorandom functions/permutations, which makes the search quite efficient. Roughly speaking, the index consists of blinded keywords $f_k(w_i)$ and lists of FIDs containing w_i, where $f()$ is a pseudorandom function and k is the secret key. The search trapdoor is also in the same form so that the server can perform matching. However, it only supports single-keyword exact query.

Scalar-Product-Preserving Encryption —An SPE scheme [15] preserves the dot product between two d-dimensional vectors (e.g., a query vector $\vec{q}$ and a database record $\vec{p_i}$). Roughly speaking, the secret key is composed of one $(d + 1)$-bit vector $\vec{S}$ and two $(d + 1) \times (d + 1)$ invertible matrices $\{M_1, M_2\}$. Every data vector $\vec{p_i}$ and query vector $\vec{q}$ are extended, $\vec{p_i}$ is randomly scaled, and both of them are split into two random vectors $\{\vec{p_i}', \vec{p_i}''\}$ and $\{\vec{q}', \vec{q}''\}$, respectively. They are encrypted as $\{M_1^T\vec{p_i}', \mathrm{M}_2^T\vec{p_i}''\}$ and $\{\mathrm{M}^{1}_{-1}\vec{q}', \mathrm{M}^{2}_{-1}\vec{q}''\}$, respectively. The server can then recover $r\vec{q} \cdot \vec{p_i}$ from ciphertexts, without knowing the original vectors. For each dimension, the method of splitting is controlled by the corresponding bit in key vector $\vec{S}$, which is proven to provide sufficient security against known-ciphertext attacks [15].

Order-Preserving Symmetric Encryption — In OPSE [2], the numerical ordering of plaintext is preserved after encryption. Boldyreva *et al.* [2] provide the first cryptographic construction of OPSE that is provably secure under the security framework of pseudorandom function or pseudorandom permutation. It can be regarded as a function $g(\cdot)$ from a domain $\mathcal{D} = \{1, \ldots, M\}$ to a range $\mathcal{R} = \{1, \ldots, N\}$.

Achieving Secure Ranked Search over Encrypted Data

An especially important functionality in plaintext IR is to support ranking mechanisms over search results according to user-specified relevance criteria. Usually, this is achieved by building an inverted index (index structure) and adopting a ranking function to compute the rank of each file relevant to a given search request (primitive data operations are keyword matching and sorting). For example, the following function can be used:

$$Score(t, F_d) = \frac{1}{|F_d|} \cdot (1 + \ln f_{d,t}), \quad (1)$$

where t is the search keyword (term), $f_{d,t}$ denotes the term frequency (TF) of keyword t in file F_d, and $|F_d|$ is the number of indexed keywords in F_d.

Toward achieving secure ranked search in the encrypted domain, Zerr *et al.* [16] observed that although the posting list elements (document IDs that contain each keyword in an inverted index) are encrypted, the term frequency distribution for each posting list can still help adversaries re-identify the keyword. Thus, they proposed to transform the relevance scores such that their distribution is uniform for each keyword. They show that this scheme satisfies a definition called r-confidentiality in a statistical sense. However, it requires much preprocessing and does not easily handle score dynamics, while the security level is weak.

To improve both the efficiency and privacy, Wang *et al.* proposed a ranked symmetric searchable encryption (RSSE) scheme [12] that enables result ranking for single keyword query. To ensure privacy, a straightforward yet ideally secure RSSE scheme can be derived based on the existing SSE solution in [6], but requires two rounds of interactions between the user and the cloud server, which incurs high communication overhead. Thus, they adopt OPSE [2] to obtain practical performance, where the numerical ordering of the plaintexts is preserved after encryption. Specifically, during the search operation the relevance order (OPSE encrypted relevance scores) of each document is revealed to the server. In this way, efficient relevance score ranking can be done as in the plaintext domain. However, because the original OPSE is a deterministic encryption scheme, this still leaks much information. If the server has some background information on the dataset, such as the distribution of relevance scores for each plaintext keyword, it could reverse-engineer the keyword.

To break this determinacy, the authors propose one-to-many order-preserving mapping (OPM), which maps the same relevance score to different encrypted values. They incorporate the unique file IDs together with the plaintext as the random seed in the final ciphertext chosen process in OPSE. Thus, the same plaintext will no longer be deterministically assigned to the same ciphertext, but instead to a random value within the randomly assigned bucket in a range $\mathcal{R}$. Furthermore, they use different keys to encrypt the relevance score for different posting lists (documents containing each keyword) to make the OPM more indistinguishable.

The RSSE scheme achieves *data and index privacy*, because the relevance scores in the searchable index are encrypted using OPSE with OPM. The highly flattened one-to-many mapping and the fully randomized score-to-bucket assignment in OPSE makes it difficult for an adversary to predict the original plaintext score distribution by observing the ciphertext. In addition, this scheme hides the search keyword from the adversary. But since the trapdoor is deterministic, it does not provide query unlinkability. For efficiency, the encrypted index generation and search operations can both be finished within seconds for 1000 documents.

The above method cannot directly handle multiple-keyword ranked search, because the order of OPSE's ciphertext will

Schemes	Search Functions	Security	IR method	Encryption Primitive	Index Structure	Primitive Data Operation
[16]	b	*r*-confidentiality	Top-*k*	Any	Inverted index	Sorting
[12]	b	i,ii, KB	Relevance score ranking	OPSE	Inverted index	Equality test, sorting
[3]	a, b	i,ii,iii, KB	Coordinate matching	SPE	Binary vectors	Dot product
[9]	c	i,ii	Fuzzy matching	SSE	Inverted index	Equality test
[14]	c	i,ii	Similarity, edit distance based	SSE	Inverted index, trie	Equality test
[5]	d, neighbor query	i,ii	NA	SSE	Graph	Equality test
[4]	d, subgraph query	i,ii,KB	Filtering and verification	SPE	Feature vector	Dot product

Table 1. *Privacy-assured search schemes.*

not be preserved for the sum of multiple relevance scores. To support secure multi-keyword ranked search over encrypted data (MRSE), Cao *et al.* [3] proposed to adopt another similarity measure from the IR community, *coordinate matching*, which captures the relevance of documents to a query through the number of query keywords appearing in a document. Each document index and the query are described as a binary vector (index structure), respectively, such that the similarity is measured by the dot product of the two vectors (primitive data operation). In order to protect the index privacy and search privacy, one shall encrypt the index and query vectors, and compute the similarity score over ciphertexts.

To this end, they propose a secure inner-product computation mechanism that adapts the SPE scheme originally used for secure k-nearest neighbor (kNN) query in [15]. Basically, the search operation should compute the dot product between a query vector $\vec{q}$ and each data (index) vector $\vec{p_i}$. However, a straightforward application of SPE is not secure as it linearly preserves the dot product, by which the server can statistically analyze similarity scores for two queries differing in one keyword to learn that keyword (called *scale analysis attack*), especially in the KB model. Therefore, to build a secure MRSE that preserves search privacy, they obfuscate the document frequency to diminish the chances for re-identification of keywords. In particular, they propose to add randomness to both the data vector and query vector in order to blind the exact similarity score from the server. The randomness is added on the fly, by extending both vectors with dummy random keywords.

In the MRSE scheme, the data and index privacy are achieved since the encryption algorithm is secure in the KC model. In addition, under the KB model *query confidentiality* is achieved as well as trapdoor unlinkability. It introduces nearly constant search overhead with the increase of keywords; in contrast, in other multiple-keyword search schemes this is linear.

Dealing with Fuzziness and Similarity

In the IR community, similarity can be defined using the edit distance, which represents the least number of modification operations (including deletion, insertion, and substitution) needed to change one word into another. For example, the edit distance between "britney" and "briny" is 2. When the edit distance equals 1, it is usually called *fuzzy search*. Thus, in a similarity search problem, given a keyword w and an edit distance d, a search execution should return a set of files $\{\mathrm{FID}_{w_i}\}$, where $\mathbf{ed}(w, w_i) \leq d$.

A straightforward method is to build a *similarity keyword set* that incorporates not only the exact keywords but also those that differ by up to edit distance d. Also, the search would require a trapdoor for each similar word to the query keyword. However, this incurs high storage and computation overhead at the server side due to the large number of possible similar keywords. Thus, in [9], Li *et al.* proposed using two data representation techniques to enhance the efficiency for fuzzy search. One is to compress the fuzzy keyword set S_{w_i} of each keyword w_i using wildcard letter ★; the other is to use Bloom Filters to represent the index corresponding to a fuzzy keyword set. The first technique reduces the server storage and search cost linear to the product of keyword length with the number of keywords ($O(l|W|)$), whereas the second one further reduces it to $O(|W|)$.

In [14], Wang *et al.* extended the above scheme to handle the general case when $d > 1$. The wildcard suppression technique brings more storage savings when d is larger. However, since the search complexity is still linear to the total number of keywords, they propose to index and encode the keywords using a trie data structure so that the search cost is reduced to $O(1)$.

To protect privacy, the above two schemes both utilize SSE as the basic encryption primitive. Their schemes are proven secure following the rigorous security definitions of SSE. Both of them achieve data and index privacy and query confidentiality, while being very efficient. The security intuition is that each keyword in the similarity set and the query input is encrypted using a pseudorandom function into a random bit string such that the server cannot distinguish keywords from each other. However, query unlinkability is not achieved because of the deterministic nature of SSE.

Handling Structured Data

Large portions of online data are not stored in a simple text form; rather they are have rich data structures. Especially, graphs has been increasingly used to model complicated data, such as social network graphs, medical workflows, relational databases, chemical compounds, and personal images. As more and more sensitive structured data are outsourced, users need to effectively search them even when they are encrypted.

Here we try to give a formal definition of structured data. A structured data record can be viewed as a combination of data structure δ (in case of a graph: $G = (V, E)$ where V is the node set and E stands for the relation set), and a sequence of data items $\mathcal{D} = \{m_1, \ldots, m_n\}$ associated with the nodes.

There can be one or multiple such data records. A query on each δ returns a set of pointers I to a subset of data items in $\mathcal{D}$. A structured searchable encryption scheme transforms the original data into: an encrypted structure index γ, and encrypted data items $\mathcal{C} = \{c_1, \ldots, c_n\}$. Using a query trapdoor τ, the pointers to $\{c_i\}_{i \in I}$ can be recovered from γ and τ. Note that the structured data intrinsically contain relations among data items. In contrast, non-structured data are only a collection of data items; for example, file databases and DNA matching.

Under the above notion, recently Chase and Kamara proposed structured encryption [5] to handle private access to parts of a large matrix or graph in encrypted form. Their scheme generalizes the SSE in [6], where the encrypted index γ is generated from δ by encrypting the structure information (i.e., using pseudo-random functions and permutations). Formal security notions are defined in their work. However, only simple functionalities such as data value access and neighbor queries are supported.

Later, Cao *et al.* [4] proposed a privacy-preserving graph containment query scheme (PPGQ), where whole graphs that contain the query graph are returned. In the plaintext domain, graph containment means checking subgraph isomorphism. As direct checking is NP-complete, the principle of *filtering and verification* is usually used, where a feature-based index is prebuilt for each graph. When data graphs are stored in encrypted form in the cloud, the filtering method based on plaintext index is no longer available. Thus, to support privacy-assured graph search over encrypted data, they convert both the data and query graphs into binary feature vectors (data structure), and use their dot product as the filtering condition (primitive operation). Each bit within a data/query graph's vector represents whether the corresponding feature is subgraph-isomorphic to that graph or not; only when the dot product of the two vectors equals the number of query features will the matching data graph be returned.

The PPGQ scheme is based on the SPE scheme in [5]. As a straightforward application of it would violate the query privacy, the authors propose another randomization technique. Essentially, only the dot products between a query graph and the matching graphs are preserved, while all of the others are randomized. The scheme achieves both data and index privacy, and query confidentiality under the KB model. Performance evaluation shows the trapdoor generation and search functions are very efficient.

Trade-offs and Impacts of Design Choices

In designing privacy-assured search schemes, it is generally perceived that efficiency vs. privacy is an intrinsic trade-off; that is, to achieve higher efficiency, the privacy level will be sacrificed. However, when access pattern privacy is acceptable to leak, higher efficiency may be gained without decreasing privacy, which depends on all the factors including the IR method, index structure, and encryption primitive. For instance, in [14] the use of a trie instead of a list to encode the fuzzy keywords dramatically enhances the efficiency, while the scheme still remains provably secure due to the security of SSE. On the other hand, in [3, 4], the trade-off is mainly between accuracy and privacy; that is, the privacy level decreases if higher result ranking accuracy is desired.

Future Challenges

There are many interesting research issues worth further investigation. The works mentioned above have a common characteristic: they relax the privacy guarantees (i.e., "as strong as possible") to achieve higher efficiency performance. While there are formal privacy definitions for searchable encryption that reveal the access pattern [6], for as-strong-as-possible schemes, how to formally analyze the privacy level given various known background information remains an interesting and important open problem. Addressing it may require tools from information theory and statistics. Differential privacy [7] is a useful formal privacy measure, but it only applies to statistical queries. It is necessary to develop similar privacy metrics for ranked search.

For multi-keyword ranked search, it is desirable to enable advanced relevance criteria such as the ones commonly used in IR. The problem is how to hide the sum of multiple keywords' relevance scores from the server, which may possess statistical distribution information to re-identify the search keywords. A possible approach would be to use a similar procedure as in MRSE to build randomized document indexes and query vectors. For search on structured data, the PPGQ scheme does not handle graphs with labeled nodes, which, however, is quite common in practice as graph nodes usually have concrete and different meanings. On the other hand, because of noises that are usually contained in graph databases, exact graph containment queries may often return very few results. Thus, enabling similarity searches over encrypted graphs is another important functionality for outsourced graph-structured data. It is worthwhile to explore a variety of similarity measures used in the plaintext graphical-IR domain.

In addition, for wider applicability in different scenarios, can we make public-key-based searchable encryption more practical and secure? We studied privacy enhancements of public-key-based multidimensional queries in [10], while its efficiency is still well behind symmetric-key-based solutions. Finally, it is also interesting to ask if more complex data utilization functions can be efficiently evaluated on encrypted data; for example, running database join/merge queries, or graph algorithms on structured data.

Concluding Remarks

In this article, we identify the problem and challenges of enabling privacy-assured searchable cloud data storage services. Recent research advances in this field are surveyed, which suggest that achieving functionally rich, usable, and efficient search on encrypted data is possible without sacrificing privacy guarantee too much. The steady evolution of this field will need to bring expertise from the cryptography, database, and information retrieval communities.

Acknowledgments

The authors would like to thank the reviewers for their helpful comments. Li's work was partly supported by the U.S. National Science Foundation (NSF) under grant CNS-1218085; Ren's research was supported in part by NSF under grants CNS-1054317 and CNS-1262275; Lou and Hou's work was supported by NSF under grants CNS-1155988 and CNS-1217889.

References

[1] M. Armbrust *et al.*, "Above the Clouds: A Berkeley View of Cloud Computing," Feb 2009.

[2] A. Boldyreva *et al.*, "Order-Preserving Symmetric Encryption," *Proc. Eurocrypt '09*, LNCS, vol. 5479, Springer, 2009.

[3] N. Cao *et al.*, "Privacy-Preserving Multi-Keyword Ranked Search Over Encrypted Cloud Data," *IEEE INFOCOM*, 2011, pp. 829–37.

[4] N. Cao *et al.*, "Privacy-Preserving Query Over Encrypted Graph-Structured Data in Cloud Computing," *31st Int'l. Conf. Distributed Computing Systems*, 2011, pp. 393–402.

[5] M. Chase and S. Kamara, "Structured Encryption and Controlled Disclosure," *Advances in Cryptology-ASIACRYPT 2010*, 2010, pp. 577–94.

[6] R. Curtmola *et al.*, "Searchable Symmetric Encryption: Improved Definitions and Efficient Constructions," *Proc. ACM CCS '06*, 2006.

[7] C. Dwork, "Differential Privacy," *Automata, Languages and Programming*, 2006, pp. 1–12.

[8] S. Kamara and K. Lauter, "Cryptographic Cloud Storage," *Proc. 14th Int'l. Conf. Financial Cryptograpy and Data Security*, Berlin, Heidelberg, 2010, pp. 136–49.

[9] J. Li *et al.*, "Fuzzy Keyword Search Over Encrypted Data in Cloud Computing," *Proc. IEEE INFOCOM '10 Mini-Conf.*, San Diego, CA, Mar. 2010.

[10] M. Li *et al.*, "Authorized Private Keyword Search over Encrypted Data in Cloud Computing," *31st Int'l. Conf. Distributed Computing Systems*, 2011, pp. 383–92.

[11] D. Song, D. Wagner, and A. Perrig, "Practical Techniques for Searches on Encrypted Data," *Proc. IEEE S & P 2000*, 2000.

[12] C. Wang *et al.*, "Secure Ranked Keyword Search Over Encrypted Cloud Data," *Proc. ICDCS '10*, 2010.

[13] C. Wang *et al.*, "Toward Publicly Auditable Secure Cloud Data Storage Services," *IEEE Network*, vol. 24, no. 4, July–Aug. 2010, pp. 19–24.

[14] C. Wang *et al.*, "Achieving Usable and Privacy-Assured Similarity Search over Outsourced Cloud Data," *Proc. IEEE INFOCOM '12*, Orlando, FL, Mar. 2012.

[15] W. K. Wong *et al.*, "Secure KNN Computation on Encrypted Databases," *Proc. SIGMOD*, 2009.

[16] S. Zerr *et al.*, "Zerber+r: Top-k Retrieval from a Confidential Index," *Proc. EDBT '09*, 2009.

Biographies

MING LI [S'08, M'11] (ming.li@usu.edu) received his Ph.D. in electrical and computer engineering from Worcester Polytechnic Institute. He joined the Computer Science Department at Utah State University as an assistant professor in 2011. His research interests are in the general areas of cyber security and privacy, with current emphases on data security and privacy in cloud computing, security in wireless networks, and cyber-physical systems. He is a member of ACM.

SHUCHENG YU [S'07, M'10] (sxyu1@ualr.edu) received his Ph.D. in electrical and computer engineering from Worcester Polytechnic Institute. He joined the Computer Science Department at the University of Arkansas at Little Rock as an assistant professor in 2010. His research interests are in the general areas of network security and applied cryptography. His current research interests include secure data services in cloud computing, attribute-based cryptography, and security and privacy protection in cyber physical systems. He is a member of ACM.

KUI REN [SM'11] (kren@ece.iit.edu) is an associate professor in the Computer Science and Engineering Department of the State University of New York, Buffalo. He received his PhD from Worcester Polytechnic Institute. His research interests include cloud and big data security, wireless security, and smart grid security. He received an NSF CAREER Award in 2011 and the Best Paper Award of IEEE ICNP '11. He is an Associate Editor of *IEEE Transactions on Smart Grid*, *IEEE Wireless Communications*, *IEEE Communications Surveys & Tutorials*, and the *Journal of Communications & Networks*.

WENJING LOU [S'01, M'03, SM'08] (wjlou@vt.edu) has been an associate professor at Virginia Tech since 2011. She was on the faculty of Worcester Polytechnic Institute from 2003 to 2011. She received her Ph.D. in electrical and computer engineering from the University of Florida in 2003. Her current research interest is cyber security, with emphases on wireless network security and data security and privacy in cloud computing. She was a recipient of the U.S. NSF CAREER award in 2008.

Y. THOMAS HOU [S'91, M'98, SM'04] (thou@vt.edu) is a professor in the Bradley Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg. His research interests are cross-layer design and optimization for cognitive radio wireless networks, cooperative communications, MIMO-based ad hoc networks, and new interference management schemes for wireless networks. He is also interested in wireless security. He is currently serving as an Area Editor of *IEEE Transactions on Wireless Communications*, and an Editor of *IEEE Transactions on Mobile Computing*, *IEEE Journal on Selected Areas in Communications* Cognitive Radio Series, and *IEEE Wireless Communications*.