Search in Encrypted Data: Theoretical Models and Practical Applications

Qiang Tang

APSIA group, SnT, University of Luxembourg
6, rue Richard Coudenhove-Kalergi, L-1359 Luxembourg
qiang.tang@uni.lu

November 14, 2012

Abstract. Recently, the concept of Search in Encrypted Data (SED) has become a highlight in cryptography. A SED scheme enables a client to have third-party server(s) to perform certain search functionalities on his encrypted data. In this book chapter, we aim at conducting a systematic study on SED schemes. Firstly, we describe three application scenarios and identify the desirable security requirements. Secondly, we provide two orthogonal categorizations and review the related security models for each category of SED schemes. Thirdly, we analyze the practical issues related to SED schemes and identify some future research directions.

1 Introduction

A Search in Encrypted Data (SED) scheme allows third-party server(s) to search on behalf of a client without the need to recover the plaintext data while preventing the server(s) from learning any plaintext information. SED has become a very active research area in cryptography in recent years. Two seminal SED schemes are the one by Song, Wagner, and Perrig [37] and the one by Boneh et al. [8]. The scheme from [37] allows a client to encrypt its database and store the encrypted database at a remote server. Later on, the client can instruct the server to search in the encrypted database and return the relevant data. The scheme from [8] is often referred to as PEKS, namely public key encryption with keyword search. With a PEKS scheme, a client publishes his public key so that any entity can encrypt messages for him. Later on, the client can allow a third-party server to search in the encrypted messages by assigning a token to it. Following [8, 37], a lot of variants have been proposed to extend the concepts in many aspects. For instance, Yang et al. proposed the concept of public key encryption supporting equality test [43]. In contrast to the scheme from [8], the scheme from [43] allows a third-party server to search on the ciphertexts which are encrypted with public keys from multiple different clients. With the wide adoption of cloud computing applications, SED schemes have been regarded by many to be an important technology in securing outsourcing databases while preserving data utility and confidentiality.

In this book chapter, we aim at a systematic study on existing SED schemes and their security implications. In particular, we reflect on the related theoretical security models and try to understand their practical security guarantees.

In the first step, we study three representative application scenarios, which have motivated a variety of theoretical SED schemes. Despite the frequent citations, the security requirements of these scenarios have not been investigated in depth in the literature. This fact means that there may be a gap between the theoretical security guarantees of existing SED schemes and the practical needs of the application scenarios. In the second step, we present two categorizations for SED schemes. One is based on whether a scheme supports full-domain or index-based search. The other is based on the answers to two questions, namely "Who can contribute searchable data in the outsourced database?" and "Who can issue search queries to the third-party server(s)?". Due to the desired storage and search functionalities, outsourcing data storage and search operations to third-party server(s) inevitably results in some privacy loss for the client. The answers to the above two questions define the characteristics of the search functionalities provided by a SED scheme, and consequently determine the *inevitable information leakage* of the scheme. In the third step, based on the results in the first two steps, we provide some practicality analysis against the existing provably secure SED schemes. The analysis shows that many practical security concerns have not been covered by theoretical security models. As a result, we are able to identify some future research directions for SED schemes.

Organization. The rest of this chapter is organized as follows. In Section 2, we describe three application scenarios and identify their security requirements. In Section 3, we categorize the existing SED schemes. In Section 4, we review the security models for SED schemes. In Section 5, we analyze the practical issues facing SED schemes. In Section 6, we conclude the chapter.

2 SED Application Scenarios

In this section, we describe three representative application scenarios for SED schemes and the related security requirements. In addition, we also mention some possible variants.

2.1 Search in Outsourced Personal Database

Suppose Alice is a frequent traveler and needs to access her database during her travel anytime and anywhere in the world. For this, Alice can outsource her personal database to a third-party service provider, such as Google or Dropbox. With this approach, Alice needs to reveal everything (the data and search criteria) to the third-party service provider, which makes the solution undesirable from the privacy perspective. To achieve a privacy-preserving solution, Alice can employ a SED scheme to encrypt her database and outsource the ciphertext. Later on, Alice can issue a search query (containing encrypted search criteria) to the service provider, which can then search in the database and return the encrypted documents which match the search criteria. As to this scenario, we distill the following security requirements.

- Only Alice can generate contents for the oursourced database and decrypt the encrypted contents in the database, and only Alice can issue meaningful search queries.
- No entity, including the server, should learn what Alice has searched for.

This application scenario was firstly mentioned in [37], and has motivated the investigation of SED schemes in the symmetric-setting in Section 3.2.

2.2 Email Routing Service

Among all oursourcing services, email may be one of the most well-known examples, where users' email data is stored and related services are managed by the email service providers. In email services, the service providers normally have access to all emails of their customers in plaintext so that a lot of privacy concerns exist (e.g. sensitive email messages and targeted advertisements). Now, suppose that there is an email service provider, which wants to provide secure email service and allow users to receive encrypted emails. In this situation, a user Alice can employ a SED scheme and have all her emails encrypted under her (public) key. Later on, Alice can ask the service provider to search in her encrypted emails and then selectively retrieve the interesting ones. For instance, during her vacation, Alice can simply retrieve the emails labeled as "urgent" through her smart phone, without being bothered by other emails. As to this scenario, we distill the following security requirements.

- Every entity should be able to generate encrypted emails for Alice, but only Alice can read her emails.
- Only Alice can issue meaningful email retrieval queries. In addition, Alice may also want the service provider to scan her encrypted emails, in particular the attachments, to detect viruses or malwares without learning unnecessary plaintext information [26].
- No entity, including the server, should learn what Alice has searched for.

This application scenario was firstly mentioned in [8], and has motivated the investigation of SED schemes in the asymmetric-setting in Section 3.2.

2.3 Matching in Internet-based PHR Systems

An Internet-based PHR (personal healthcare record) system, such as Microsoft Health Vault[1], helps users store their PHRs and allow the information to be accessed and edited via a web browser or some APIs, and they may also help users find kindred spirits (i.e. build social networks) and share their information. Considering a user, say Alice, her PHR data can come from a lot of sources. For example, she can get prescription results from her doctor, treatments from a hospital, test results from a laboratory, and monitoring results from home-based sensors. Quite often, a lot of Alice's PHR data may be directly sent to Alice's account, while the rest will be input by Alice herself. Fig. 1 shows a general picture of an Internet-based PHR system.

In most existing Internet-based PHR systems, users will be provided privacy controls. However, there are a number of concerns which stop users from sharing their data. One concern is that the system providers, say Microsoft, are always able to fully access the data. Although there will be some privacy agreement,

[1] http://www.healthvault.com/personal/index.aspx

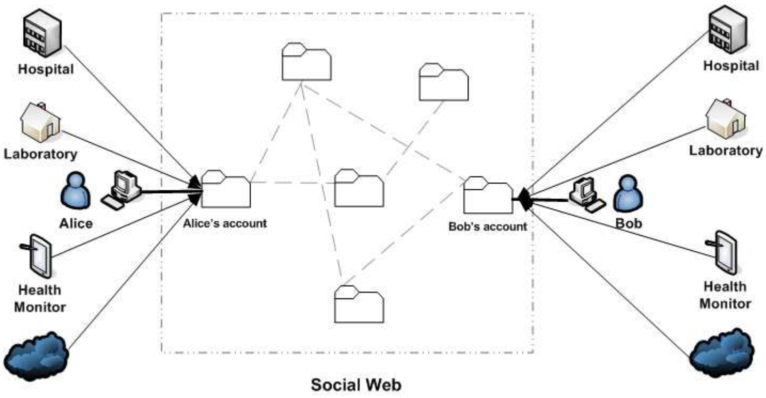

Fig. 1. An Illustration of Outsourced PHRs

but users may still worry about that these providers may abuse their data. The other concern is that, even if the service providers behave honestly, their databases may be compromised, in which case all data may be leaked. Since PHRs are sensitive information to individuals, and an information leakage may cause undesirable consequences, such as being discriminated by the potential employer because of a disease.

To solve the privacy problem, users can employ a SED scheme and have all their PHR data encrypted under their own (public) keys. Moreover, the users can authorize third-party server(s) to match their encrypted data without recovering the plaintext information. As to this scenario, we distill the following security requirements.

- A user, say Alice, should be able to allow multiple entities to generate contents for her, and only Alice should be able to decrypt the encrypted contents intended for her.
- Together with another user, Alice can authorize third-party server(s) to match their ciphertexts. The authorization should support different levels of granularity, and third-party server(s) should not be able to learn any information more than the match result, such as "equal" or "not-equal" in exact matching.

This application scenario has motivated the SED schemes for joint databases in the asymmetric-setting [39–41, 43].

3 Categorization of SED Schemes

In this section, we review and categorize the existing SED schemes which follow the works from [8, 37]. Note that some researchers have regarded *functional encryption* as a general form of searchable encryption [10], where a client outsources its encrypted data to third-party server(s) and assigns tokens to them so that they recover certain plaintexts if the tokens satisfy some conditions. Nevertheless, we omit discussions of these generalized primitives.

3.1 Full-domain vs. Index-based

Given a data item, which can be of various forms (e.g. an email, a document, a video clip, etc), search can be done in two ways.

- One is to perform full-domain search, in which a search will sequentially go through every data item in order to test some criteria. For instance, a search can be to test whether some term appears more than a threshold times and another term does not appear in the content. Full-domain search takes linear operations to cover all data items, and is flexible in the sense that the criteria can be anything and be defined on the fly. The downside is its inefficiency when the data items are of big size.
- The other is index-based search or keyword-based search, where every data item is firstly characterized by a list of keywords which are then used to build a search index for the data item. Later, when a search takes place, the criteria will be tested based on the indexes instead of the contents of data items. In the literature, there are two basic ways to construct indexes, namely forward index shown in Fig. 2 and inverted index shown in Fig. 3. With this approach, search can be done much more efficiently since it does not need to go through the contents of all data items. The downside is that search criteria may not be as flexible as in the case of full-domain search. The performance will depend on the selection of keywords and how the index is constructed and maintained.

item identifiers	**keywords**
Item ID1	keyword1, keyword3, keyword5
Item ID2	keyword5,
Item ID3	keyword2, keyword4
Item ID4	keyword2, keyword3
Item ID5	keyword1, keyword2 keyword4
Item ID6	keyword4, keyword5 keyword6
Item ID7	keyword2, keyword3 keyword4, keyword5

Fig. 2. Forward Index

keywords	**item identifiers**
keyword1	Item ID1, Item ID5
keyword2	Item ID3, Item ID4, Item ID5, Item ID7
keyword3	Item ID1, Item ID4, Item ID7
keyword4	Item ID3, Item ID5, Item ID6, Item ID7
keyword5	Item ID1, Item ID2, Item ID6, Item ID7
keyword6	Item ID6

Fig. 3. Inverted Index

The above description assumes that no encryption is done on the data and indexes. Depending on the intended search functionality, SED in the full-domain setting will encrypt the data items in certain block-wise manner, while SED in the index-based setting will encrypt the indexes and keywords. In both settings, search queries will be blinded to prevent information leakage. In next subsection, we categorize the existing SED schemes into these settings.

3.2 Symmetric-Setting vs. Asymmetric-Setting

Based on the answer to the questions "Who can contribute searchable data in the outsourced database?" and "Who can issue search queries to the third-party server(s)?", SED schemes can be divided into two categories, which are referred to as *symmetric-setting* and *asymmetric-setting*. This categorization is orthogonal to that in previous subsection.

In the symmetric-setting, as described in the seminal paper [37], only the client is able to generate the searchable contents, generate valid search queries, and decrypt the encrypted contents. This setting matches the application in Section 2.1. The SED scheme from [37] assumes full-domain search and the encryption is word by word. Along the line, a number of variants exist.

- Despite of its more general purpose, the seminal work of Goldreich and Ostrovsky on Oblivious RAMs [21] gives us techniques for performing full-domain search on encrypted data stored on a server. Compared with other schemes, the advantage of Oblivious RAMs is allowing the client to hide data access patterns (i.e. achieving query result privacy as defined below). The downside of this approach is that it requires polylog rounds of communication and significantly more inefficient than other schemes.
- Following the work in [37], the concept of index-based SED schemes in the symmetric-setting was investigated in [16, 18, 20]. Goh [20] formalized the IND-CKA property (i.e. indistinguishability against chosen keyword attack), namely semantic security against an adaptive chosen keyword attack, and an improved IND2-CKA property which is slightly stronger than IND-CKA. Goh also proposed two schemes based on bloom filters, which can result in some false positive incidents in the query results. Chang and Mitzenmacher [16] formalized a property similar to IND2-CKA, and proposed two SED schemes without bloom filters. Curtmola et al. [18] improved these security definitions by taking into account trapdoor privacy which has not been adequately covered in [16, 20]. The constructions in [16, 20] use forward index, while the construction in [18] uses inverted index.
- Shen, Shi, and Waters [36] formulated the *full security* property for symmetric-key predicate-only encryption ($\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{GenToken}, \mathsf{Query}$), which is a building block for predicate encryption. Though the primitive is not initially designed for the purpose of SED, but it can be regarded as a general formulation of SED with forward index: the client runs Setup to generate his secret key; given a document, the index is computed as an encryption of the keywords by the function $\mathsf{Encrypt}$; by representing the search criteria with a predicate, a trapdoor can be generated by the function $\mathsf{GenToken}$; a search is done by running the function Query to test whether the keywords in an index satisfy the predicate or not. In this regard, it is more general than those formulations in [16, 20].
- The seminal work [37] considers a single-user setting, where the client is the intended receiver of encrypted data items. How to extend index-based

SED to multi-user seeting has been investigated in [1,5,7,18,38]. In the formulation of Curtmola et al. [18], multiple intended receivers can submit search queries. The client (who contributes the searchable data) dynamically manages the receivers' search capability and the scheme construction is based on the concept of broadcast encryption. In the formulation of Bao et al. [5], a new party, namely user manager, is introduced into the system, to manage multiple clients' search capability. Unlike the formulation in [18], all clients can contribute searchable contents and can search. The downside of this formulation is that the user manager is also capable of submitting search queries and decrypting encrypted data items. In practice, all clients are required to fully-trust the user manager, who may be a single point of failure for the whole system. In addition, the clients need to interact with the user manager to generate an index. This may be an efficiency drawback. In the formulations from [1,7], any entity can search in all encrypted data. These SED schemes are termed order preserving encryption, where the ciphertexts preserve the order the plaintexts, so that every entity can perform an equality comparison. One drawback of these schemes is that they do not define standard security notions. In contrast, the scheme from [38] allows similar functionality but with standard security notions.

- Golle, Staddon, and Waters [22] proposed two SED schemes which allow the client to attach multiple keywords to a data item and then perform conjunctive keyword search. In both schemes, an index tells the number of encrypted keywords.

- Most SED schemes assume that a search is either a sequential scan in the encrypted data or a direct matching in the index. Chase and Kamara [17] specifically studied SED for structured data (e.g. matrices, labeled data, graphes, and labeled graphes) and investigated its application in controlled disclosure.

- Islam, Kuzu, and Kantarcioglu [28] investigated an attack with respect to data access pattern leakage and presented experimental results. Moreover, they proposed a generic mitigation technique, which introduces a certain degree of false positive rate in the searching. Christoph et al. [12] proposed an interactive SED scheme based on the recent advances in fully homomorphic encryption schemes. The proposed scheme is secure in a security model which is at least as strong as that from [36]. The original scheme supports equality test, but can be easily adapted to support many useful search features, including aggregating search results, supporting conjunctive keyword search queries, advanced keyword search, search with keyword occurrence frequency, and search based on inner product. With all the schemes, the client can protect his access pattern by retrieving the matched documents through some private information retrieval (PIR) protocols. Compared with the solution from [28], PIR will not introduce any false positive results, although it may suffer from complexity penalties.

- Hore et al. [24] investigated the problem of supporting multidimensional range queries on encrypted data, and proposed a solution based on the idea of computing a secure indexing tag of the data by applying bucketization. Kerschbaum and Sorniotti [30] proposed a SED scheme in the symmetric-setting, that supports range queries, relying only on Diffie-Hellman assumptions in the random oracle model. Kuzu et al. [31] and Raykova et al. [33] investigated SED schemes that allow searching based on keyword similarities.

In the asymmetric-setting, as described in the seminal paper [8], every entity is able to generate searchable contents without any explicit authorization from the client. By default, only the client can generate valid search queries and decrypt the encrypted contents. This setting matches the application in Section 2.2. Note that the SED formulation from [8] is often referred to as PEKS, namely public key encryption with keyword search. The concept formulation and scheme construction of PEKS from [8] are limited in many aspects, and a lot of variants have proposed thereafter.

- The PEKS formulation [8] only considers the encryption of single keyword and only supports exact matching in the search. The authors from [11, 25, 27] extended the formulation to support search queries with conjunctive keywords. Their extension are straightforward in the sense that the keywords are somehow still encrypted one by one, and an index tells the number of encrypted keywords. Moreover, the solution from [11] also supports subset and range queries. Katz, Sahai and Waters [29] proposed schemes supporting disjunctions, polynomial equations, and inner products. Note that the schemes from [11, 27, 29] have been under the name of predicate encryption, which can provide the functionality of SED.
- The PEKS formulation [8] only covers the encryption and matching of keywords, but it does not touch on the encryption of affiliated messages. A natural question is whether we can use the same public key for encrypting both the keywords and the messages. The concept of integrating PKE and PEKS has been investigated in [3, 44]. It is worth noting that these formulations do not address the linkability concern discussed in Section 5.
- When applying a PEKS scheme, it is assumed that the trapdoor should always be transmitted from the client to the server through a secure channel [8]. The authors from [4, 34] argued that this additional requirement is not realistic in some application scenarios, so they proposed the concept of PEKS with designated tester (referred to as dPEKS), by letting the encryption of keywords done under both public keys from the client and a specific server. On one hand, dPEKS successfully gets rid of the security channel requirement of PEKS. On the other hand, it has very limited applications due to the following facts: the message sender must obtain both public keys and, for one ciphertext, only one server can search. For both PEKS and dPEKS, the schemes may still be vulnerable to offline keyword recovery attacks against a curious server [14]. In addition, it is emphasized that keyword ciphertext is secure against a malicious client [4], but it is unclear why this makes any difference because the client can always recover the message and somehow determine the encrypted keywords any way.
- With respect to index-based SED schemes in the asymmetric-setting, the keyword set is certainly required to be public so that every entity can generate searchable contents. Byun et al. [14] showed that the server can mount an offline message recovery attack to recover the keywords in the received trapdoors. The attack is straightforward. The server first generates fake tags based on the keywords it chooses, and then analyses the results of the client's search queries. If a trapdoor matches a fake tag (say, from keyword w), then the server can conclude that other matched tags also contain w and those unmatched tags do not contain w. To address this type of attack, Tang and Chen [42] proposed the concept of PEKS with registered keywords. In this new formulation, the message senders need to be authorized by the client before being able to generate meaningful indexes. With the extra authorization process, the client can prevent entities such as the server from mounting

an attack. The downside of this approach is that the setup phase becomes more complex. Another countermeasure is to use the computational puzzle technique, as shown in [39, 41]. With this approach, a brute-force attack can be made computationally expensive, but the downside is that the message sender needs to compute a puzzle in order to construct an index. In [4], Baek, Safiavi-Naini, and Susilo suggested to refresh the keyword set to mitigate attacks from the server. The problem with this approach is that new trapdoors cannot be used to search old contents, and this may cause usability problem in many applications. Moreover, it is still possible for the server to recover the keywords in trapdoors.

- By definition, a PEKS scheme only allows the server to search the client's ciphertexts, and there are scenarios two clients may want a server to perform certain forms of search in their joint database. Yang et al. [43] proposed the concept PKEET (public key encryption supporting equality test). The drawback with their formulation is that any entity can perform the search without any explicit authorization. Tang [39–41] introduced the concept of fine grained PKEET and presented corresponding constructions. Under the new formulations, two clients can authorize a third-party server to perform equality test based on their ciphertexts and some simple form of fuzzy equality testing. The authorization can come with different levels of granularity. This extension matches the internet-based PHR application described in Section 2.3.

- In PEKS and most of its variants (and SED in the symmetric-setting as well), the server always knows the search results, namely the documents matched by every search request. This pattern itself leaks some information about the encrypted documents (and keywords), although no entity except the client can decrypt anything. Boneh et al. [9] investigated the concept of public key storage with keyword search, which is an extension of PEKS that allows PIR queries. Compared with PEKS [8], the extension is substantial because a message sender and the server may need to engage in an interactive protocol for storing the encrypted contents. Moreover, a search operation may also need to be done through an interactive protocol. With all these changes, a stronger security model has been designed for this primitive and the search results can be hidden from the server. The construction given in [9] is constrained in the sense that the number of data items associated to every keyword is required to be bounded by a constant.

- Hwang and Lee [25] extended the PEKS concept to multiple-user setting, where multiple clients can submit search queries for encrypted data which is intended for them all. The intuition behind their formulation is very straightforward, i.e. the index for every data item is generated based on the public keys of all clients.

- For most SED schemes in the asymmetric-setting (and SED in the symmetric-setting as well), a search request will typically require the server to sequentially scan all the encrypted data items or indexes therefore result in linear complexity. Bellare, Boldyreva and O'Neill [6] introduced the concept of efficiently full-domain searchable deterministic encryption, which allows any entity to generate searchable contents and generate trapdoors to search in encrypted contents (generated by all entities) with logarithmic complexity. The downside with this approach is that the ciphertext is deterministic, so that an adversary can test any guessed plaintext by an encryption operation, and this is strictly weaker than the security guarantees provided by any other security model.

– For almost all SED schemes in the asymmetric-setting, including some constructions of the deterministic encryption [6], search is done in the encrypted "index" instead of the ciphertexts of messages. For some application scenarios, the linkability problem described in Section 5 may be a serious issue. Ibraimi et al. [26] proposed the concept of public key encryption with delegated search (referred to as PKEDS) which allows a server to search directly in the ciphertext. The drawback with their formulation is that the security model only considers the one-wayness property against a server and it is even weaker than that in [6]. Fuhr and Paillier [19] investigated the concept of full-domain searchable encryption and presented a solution in the random oracle model. Hofheinz and Weinreb [23] revisited the concept and proposed a solution in the standard model. In both works, the authors only consider information leakage from ciphertexts and allow the server to access the information encoded in the trapdoors.

To give an intuitive view of the categorization results, we summarize the mentioned SED schemes in Fig. 4. We emphasize that the summary only contain representative SED schemes mentioned above, and it is not intended to be a full list of all existing SED schemes.

	Symmetric-Setting	**Asymmetric-Setting**
Full-domain	[1, 7, 21, 37, 38]	[6, 19, 23, 26, 39–41, 43]
Index-based	[5, 12, 16–18, 20, 22, 24, 28, 30, 31, 33, 36]	[3, 4, 8, 9, 11, 25, 27, 29, 34, 42, 44]

Fig. 4. Summary of Categorizations

In Fig. 5, we classify the existing SED schemes based on the types of queries they support.

	SED Schemes
Equality test	[1, 3–9, 11, 12, 16–31, 33, 34, 36–44]
Conjunctive and Disjunctive keyword queries	[11, 12, 22, 25, 27, 29]
Range queries	[1, 6, 7, 11, 24, 31]
Inner product queries	[12, 29, 36]

Fig. 5. Classification based on Supported Queries

4 SED Security Models

In this section, we review SED security models and analyze their security guarantees. It is worth noting that these models only focus on the confidentiality of data, namely preventing the server(s) from learning any plaintext information. In practice, integrity may also be an important concern for SED, but it has not attracted much attention so far so that we ignore the issue in our analysis.

In the following discussion, we assume the data items come from the set $\mathcal{M}$ while the keywords come from the dictionary $\mathcal{D}$. For the generality, we use a predicate set $\mathcal{F}$ to represent all possible (non-encrypted) queries. One of the simplest types of predicates is the equality test, denoted as $\mathcal{F} = \{\mathsf{f}_w \mid w \in \mathcal{D}\}$. In this case, $w' \in \mathcal{W}$ satisfies f_w if and only if $w' = w$. For a set X, the notation $x \in_R X$ means that x is chosen from the set X uniformly at random. Probabilistic polynomial-time is abbreviated as P.P.T.

4.1 Index-based SED in Symmetric-Setting

Generally, an index-based SED scheme in the symmetric-setting consists of the following polynomial-time algorithms.

- $\mathsf{KeyGen}(k)$: Run by the client, this algorithm takes a security parameter k as input, and outputs a secret key K. Let the data space be $\mathcal{M}$ and the predicate set be $\mathcal{F}$.
- $\mathsf{BuildIndex}(m, K)$: Run by the client, this algorithm takes a data item $m \in \mathcal{M}$ and the secret key K as input, and outputs an index I_m which encodes a set of keywords $u(m)$ for m. Let $|u(m)|$ be the cardinality of the set $u(m)$.
- $\mathsf{Trapdoor}(\mathsf{f}, K)$: Run by the client, this algorithm takes a predicate $\mathsf{f} \in \mathcal{F}$ and the secret key K as input, and outputs a trapdoor T_{f}.
- $\mathsf{Test}(T_{\mathsf{f}}, I_m)$: Run by the server, this algorithm takes a trapdoor T_{f} and an index I_m as input, and outputs 1 if $u(m)$ satisfies f and 0 otherwise.

This definition is a generalization of those from [5,16,20,22,36] with a forward index structure. The SED definition from [18] adopts an inverted index structure, in which the BuildIndex algorithm takes a collection of data items and outputs a single index for all the items. Despite this structural difference, security definitions for both cases stay similar so that we only describe security models according to the above definition. Note that some of the schemes (e.g. [5]) support special properties, and we do not discuss them in detail here.

For an index-based SED scheme, regardless the setting (either symmetric or asymmetric), information leakage may come from three sources, namely index, trapdoor, and query results. Correspondingly, there are three types of privacy concerns, including index privacy, trapdoor privacy, and query result privacy.

- *Index privacy*, which means that indexes should not leak unnecessary information about the encoded keywords. Ideally, an index should be constructed in such a way that the server only learns whether the predicate in the trapdoor is satisfied or not in an execution of the Test algorithm. Intuitively, an index should not allow the server to recover the encoded keywords. However, how much information an index leaks in an execution of the Test algorithm also depends on how the trapdoor is constructed (or, trapdoor privacy). Informally speaking, a trapdoor containing predicate in plaintext possibly may cause more information leakage than a trapdoor only containing encrypted keywords.
- *Trapdoor privacy*, which means that trapdoors should not leak unnecessary information about the encoded predicates. Ideally, in an execution of the Test algorithm, a trapdoor should only allow the server to tell whether the encoded predicate is satisfied by the index or not. For example, a trapdoor

should not allow the server to recover the encoded predicate. Analogy to the statement in index privacy, how much information a trapdoor leaks in an execution of the Test algorithm also depends on whether the index privacy is achieved or not. Due to this dependency, index privacy and trapdoor privacy should be considered simultaneously.

– *Query result privacy*, which means that a search query should not leak unnecessary information about the searched results to the server. It is not difficult to imagine that, in a specific application scenario, the pattern of how a data item matches the client's search request can already leak some information about the data item. We elaborate on this in Section 5.

With respect to all these privacy concerns, the adversary is an honest-but-curious (or, semi-honest) server, which will faithfully executing the protocol algorithms while they will also try to perform any operations possible to obtain private information. Essentially, the server will not deviate from the protocol specification in executing the Test algorithm to forge invalid results. Any outsider attacker is always less privileged than an honest-but-curious server, so that we will mention it only when necessary.

Among the security models available for SED (e.g. those from [16–18, 20, 36]) in this setting, the security model by Shen et al. [36] is the strongest one in the sense that it covers index privacy and trapdoor privacy simultaneously. However, none of them considers query result privacy. Formally, the definition from [36] is given below.

Definition 1. *An index-based SED scheme in the symmetric-setting is fully secure if no P.P.T. attacker has non-negligible advantage in the attack game defined in Fig. 6. The attacker's advantage is defined to be* $|\Pr[b = b'] - \frac{1}{2}|$

1. Setup. The challenger runs the KeyGen algorithm and obtains the secret key K, data space be $\mathcal{M}$, and the predicate set $\mathcal{F}$. The challenger publishes $\mathcal{M}, \mathcal{F}$ and picks a random bit b.
2. Challenge. The attacker adaptively makes the following types of queries.
   - On the j-th index query, the attacker outputs two data items $m_{j,0}, m_{j,1} \in \mathcal{M}$, where $|u(m_{j,0})| = |u(m_{j,1})|$. The challenger responds with $\mathsf{BuildIndex}(m_{j,b}, K)$.
   - On the i-th trapdoor query, the attacker outputs two predicates $\mathsf{f}_{i,0}, \mathsf{f}_{i,1} \in \mathcal{F}$. The challenger responds with $\mathsf{Trapdoor}(\mathsf{f}_{i,b}, K)$.

   The attacker's queries are subject to the restriction that, for all index queries $(m_{j,0}, m_{j,1})$ and all trapdoor queries $(\mathsf{f}_{i,0}, \mathsf{f}_{i,1})$, the following is true: $u(m_{j,0})$ satisfies $\mathsf{f}_{i,0}$ if and only if $u(m_{j,1})$ satisfies $\mathsf{f}_{i,1}$.
3. Guess. The attacker outputs a guess b'.

Fig. 6. Attack Game

This definition simultaneously models index privacy and trapdoor privacy because the challenge contains both indexes and trapdoors. With respect to index privacy, the definition implies that an attacker can not distinguish two index vectors of the same cardinality, namely

$$(\mathsf{BuildIndex}(m_{1,0}, K), \mathsf{BuildIndex}(m_{2,0}, K), \cdots),$$

$$(\mathsf{BuildIndex}(m_{1,1}, K), \mathsf{BuildIndex}(m_{2,1}, K), \cdots),$$

given that the available trapdoors perform identically between pairwise indexes from the two vectors. With respect to trapdoor privacy, the definition implies an attacker can not distinguish two trapdoor vectors of the same cardinality, namely

$$(\mathsf{Trapdoor}(\mathsf{f}_{1,0}, K), \mathsf{Trapdoor}(\mathsf{f}_{2,0}, K), \cdots),$$
$$(\mathsf{Trapdoor}(\mathsf{f}_{1,1}, K), \mathsf{Trapdoor}(\mathsf{f}_{2,1}, K), \cdots),$$

given that the the pairwise trapdoors from the two vectors perform identically on the available indexes. The restriction stated in the attack game eliminates the attacker's trivial winning situations. To satisfy the above security definition for the inner product predicate, Shen et al. [36] proposed a scheme by relying on bilinear pairings with composite-order groups, where the order is a product of four big prime numbers. So far, there is no implementation for such bilinear pairings, and the complexity is theoretically very high.

In the above definition, the value of $\mathsf{Test}(T_\mathsf{f}, I_m)$ directly tells whether $u(m)$ satisfies the predicate f or not. As a result, it does not model *query result privacy*. The security model in [18] provides similar security guarantee, although it is intended only for SED schemes that support an equality test predicate. The construction in [18] employs an inverted index structure and uses padding to hide the number of data items associated with every keyword, and employs pseudorandom functions to provide the security. The security model from [12] covers all three privacy properties, but it is specifically designed for interactive SED schemes.

4.2 Full-domain SED in Symmetric-Setting

Generally, a full-domain SED scheme in the symmetric-setting consists of the following polynomial-time algorithms.

- $\mathsf{KeyGen}(k)$: Run by the client, this algorithm takes a security parameter k as input, and outputs a secret key K. Let the data space be $\mathcal{M}$ and the predicate set be $\mathcal{F}$.
- $\mathsf{Encrypt}(m, K)$: Run by the client, this algorithm takes a data item $m \in \mathcal{M}$ and the secret key K as input, and outputs a ciphertext C_m.
- $\mathsf{Decrypt}(C_m, K)$: Run by the client, this algorithm takes a ciphertext C_m and the secret key K as input, and outputs a plaintext m or an error message.
- $\mathsf{Trapdoor}(\mathsf{f}, K)$: Run by the client, this algorithm takes a predicate $\mathsf{f} \in \mathcal{F}$ and the secret key K as input, and outputs a trapdoor T_f.
- $\mathsf{Test}(C_m, T_\mathsf{f})$: Run by the server, this algorithm takes a ciphertext C_m and a trapdoor T_f as input, and outputs 1 if m satisfies f and 0 otherwise.

This definition is a generalization of those from [1, 7, 21, 37, 38]. The definitions from [1, 7, 38] do not have Trapdoor and Test algorithms, because the order of ciphertexts is identical to the order of encrypted data items in [1, 7] and there is a public function any entity can run to compute the order of the plaintexts in any ciphertexts in [38].

For a full-domain SED scheme, regardless the setting (either symmetric or asymmetric), information leakage may come from three sources, namely ciphertext, trapdoor, and query results. Correspondingly, there are three types of privacy concerns, including ciphertext privacy, trapdoor privacy, and query result privacy. Ciphertext privacy means that indexes should not leak unnecessary information about the encoded data items, the other two types of privacy properties are similar to those for an index-based SED scheme described in Section 4.1.

- The aim of [37] is to design a SED scheme with some level of ciphertext privacy and trapdoor privacy, although the security analysis is not done through a well-defined security model. It is clearly stated that query result privacy is not an intended property. The proposed scheme in [37] relies on pseudorandom functions and symmetric-key encryption schemes.
- For the security model from [1, 7, 38], trapdoor privacy is not a concern since there is no trapdoor, and query result privacy is not the aim. As to ciphertext privacy, the security models guarantees that only the ordering of the data items is leaked from the ciphertexts. The proposed scheme in [1] uses some bucketization technique, the proposed scheme in [7] uses block ciphers, and the proposed schemes in [38] use bilinear pairings.
- The security model from [21] can cover all privacy concerns, although it is not explicitly designed for SED.

4.3 Index-based SED in Asymmetric-Setting

Generally, an index-based SED scheme in the asymmetric-setting consists of the following polynomial-time algorithms.

- $\mathsf{KeyGen}(k)$: Run by the client, this algorithm takes a security parameter k as input and generates a public/private key pair (PK_r, SK_r). Let the data space be $\mathcal{M}$ and the predicate set be $\mathcal{F}$.
- $\mathsf{Tag}(m, PK_r)$: Run by a sender, this algorithm takes a data item $m \in \mathcal{M}$ and the public key PK_r as input and outputs an index I_m which encodes a set of keywords $u(m)$ for m. Let $|u(m)|$ be the cardinality of the set $u(m)$.
- $\mathsf{Trapdoor}(\mathsf{f}, SK_r)$: Run by the client, this algorithm takes a predicate $\mathsf{f} \in \mathcal{F}$ and the private key SK_r as input, and outputs a trapdoor T_f.
- $\mathsf{Test}(I_m, T_\mathsf{f}, PK_r)$: Run by the server, this algorithm takes an index I_m, a trapdoor T_f, and the client's public key PK_r as input, and outputs 1 if $u(m)$ satisfies f and 0 otherwise.

This definition is a generalization of those from [3, 4, 8, 9, 11, 25, 27, 29, 34, 42, 44]. Nevertheless, some of the schemes (e.g. [4, 34, 42, 44]) support special properties, and we do not discuss them in detail here.

As stated in Section 4.1, a SED scheme in this setting has the same privacy concerns, namely index privacy, trapdoor privacy, and query result privacy. Similarly, an honest-but-curious server is regarded as the attacker in the security models. Below, we rephrase the definitions from [11, 27, 29] to this generalized SED definition.

Definition 2. *An index-based SED scheme in the asymmetric-setting is secure if no P.P.T. attacker has non-negligible advantage in the attack game defined in Fig. 7.*

1. Setup. The challenger runs the KeyGen algorithm and obtains a key pair (PK_r, SK_r), the data space be $\mathcal{M}$, and the predicate set $\mathcal{F}$. The challenger publishes $\mathcal{M}, \mathcal{F}$ and PK_r.
2. Phase 1. The attacker adaptively makes trapdoor queries. For a trapdoor query with the predicate f, the challenger responds with $\mathsf{Trapdoor}(\mathsf{f}, SK_r)$. At some point, the attacker outputs two data items m_0 and m_1, where $|u(m_0)| = |u(m_1)|$. In this phase, the attacker's queries are subject to the following restriction: for any trapdoor query with the input f, $u(m_0)$ satisfies f if and only if $u(m_1)$ satisfies f.
3. Challenge. The challenger chooses $b \in_R \{0, 1\}$ and responds with the challenge $I_{m_b} = \mathsf{Tag}(m_b, PK_r)$.
4. Phase 2. The attacker can issue the same type of queries as in Phase 1, subject to the same restriction.
5. Guess. The attacker outputs a guess b'.

Fig. 7. Attack Game

In the above attack game, if we let $u(m)$ contain only one keyword and the predicate be equality test one described at the beginning of this section, then it is identical to the security definition for PEKS in [8]. As to the construction of SED schemes in this category, most existing schemes make use of bilinear pairings, and use the bilinear property to perform search operations.

This definition only models index privacy, while without explicitly touching upon trapdoor privacy. Given a trapdoor, it is unclear how much information the attacker can learn about the encoded keywords in a secure SED scheme under this definition. For example, in the PEKS construction [8] the trapdoor is a deterministic function of the keyword, while in [19, 23] the server can recover the keyword in any trapdoor. Among all security models, the model from [34] considers trapdoor privacy against an outsider attacker but not an honest-but-curious server. The work [9] assumes a very special setting for SED where all algorithms except for KeyGen can be interactive protocols, and the security model covers all three privacy properties. Due to its special setting, the security model does not fit other SED schemes.

4.4 Full-domain SED in Asymmetric-Setting

Generally, a full-domain SED scheme in the asymmetric-setting consists of the following polynomial-time algorithms.

- $\mathsf{KeyGen}(k)$: Run by the client, this algorithm takes a security parameter k as input, and and generates a public/private key pair (PK_r, SK_r). Let the message space be $\mathcal{M}$ and the predicate set be $\mathcal{F}$.
- $\mathsf{Encrypt}(m, PK_r)$: Run by the client, this algorithm takes a data item $m \in \mathcal{M}$ and the public key PK_r as input, and outputs a ciphertext C_m.
- $\mathsf{Decrypt}(C_m, SK_r)$: Run by the client, this algorithm takes a ciphertext C_m and the private key SK_r as input, and outputs a plaintext m or an error message.
- $\mathsf{Trapdoor}(\mathsf{f}, SK_r)$: Run by the client, this algorithm takes a predicate $\mathsf{f} \in \mathcal{F}$ and the private key SK_r as input, and outputs a trapdoor T_{f}.
- $\mathsf{Test}(C_m, T_{\mathsf{f}}, PK_r)$: Run by the server, this algorithm takes a ciphertext C_m, a trapdoor T_{f}, and the client's public key PK_r as input, and outputs 1 if m satisfies f and 0 otherwise.

This definition is a generalization of those from [6, 26, 39–41, 43], but the specific definitions differ in certain aspects and provide different security guarantees.

- In the definition from [6], the Trapdoor algorithm takes only the predicate and the public key as input, therefore any entity can generate a trapdoor. Under this definition, a SED scheme can only achieve an enhanced version of one-wayness. The enhancement means, in addition to prevent an attacker from recovering the plaintexts, ciphertexts also hide other characteristic information of the plaintexts. The proposed schemes in [6] can be realized based on a public-key encryption scheme and a hash function in the encrypt-then-hash manner.
- In the definition from [26], the predicate set is the equality test one. There are two trapdoor functions: one is similar to the one in the above definition for generating item-dependent trapdoors, and the other is for generating master trapdoors. Correspondingly, there are two Test functions: one is used to test item-dependent trapdoor to test whether the item in the trapdoor is identical to that in a ciphertext, and the other is a server to test whether a ciphertext contain any chosen plaintext data item. Note that both Test functions need to take a master trapdoor as input. As to security, one-wayness property is defined in the presence of an honest-but curious server, while ciphertext privacy and trapdoor privacy are defined against an outside attacker. The proposed scheme in [26] makes use of bilinear pairings in the asymmetric setting.
- In the definition from [43], there is no Trapdoor function required and the Test function takes two ciphertexts as input. In the definitions given in [39–41], the Trapdoor function is replaced with an authorization function Aut function which allows two clients to generate an authorization token. Correspondingly, the Test function takes two ciphertexts and a token as input. The security models from [39, 41, 40] give better security control to the clients, in contrast to that given in [43] where clients have no security control at all. All these schemes make use of bilinear pairings.

5 Practicality Analysis

In the plaintext case, both full-domain search and index-based search can be easily implemented and they are usually supported at the same time. However, according to the discussions in Section 3, existing SED schemes are much more constrained in this perspective so that they may not satisfy the needs of the practical application scenarios, such as those described in Section 2. As to security, the analysis in Section 4 has shown that most SED schemes do not achieve the strongest security we may hope for from a theoretical perspective. Moreover, we will show below that, even in the existing strongest security models, a SED scheme may still put a client's privacy at risk in reality.

5.1 Analysis of Full-domain SED Schemes

For full-domain SED schemes, the first practical concern is the encryption block size, due to the fact that encryption techniques, either symmetric or asymmetric, only takes a binary string of certain length as input. This means that a data item is potentially needed to be split into blocks before encryption is done in a block-wise manner. For instance, in the SED scheme from [37], every word is treated as a block and only exactly matching is supported. In order to support search in a fuzzy manner, the data item may need to be encrypted letter-wise instead of word-wise. Referring to the analysis [37], the downgrade of encryption granularity may result in additional security issues, such as more severe statistical analysis. In practice, the encryption block size will not only be limited by the encryption algorithms, it also depends on the search criteria to be supported. Moreover, considering the personal database outsourcing scenario in Section 2.1 and the Internet-based PHR scenario in 2.3, there can be many different types of data items such as photos, videos, and audios. How to support a variety of search criteria on different types of data items without further deteriorating security and efficiency is an interesting research question.

Another concern is the computational and storage efficiency, in particular for SED schemes in the asymmetric-setting. For all the schemes in [26, 39–41, 43], suppose that encryption is required word-wise for a data item with x words, then the encryption will take $O(x)$ public key operations (e.g. exponentiation and pairing) and the storage may become significantly larger than the plaintext case. The more severe efficiency bottleneck lies in the fact that every search request in a single encrypted data item will also take $O(x)$ operations, which may be overwhelming for a data item of moderate size. In contrast, if the SED scheme in [6] is used, a search only requires logarithmic number of integer comparisons. But the downside is that the scheme achieves only a somewhat one-wayness property.

Yet another concern is that, for schemes from [39–41, 43] and one scheme from [6] in the asymmetric setting, the encryption of every data block contains two parts: one part is a standard encryption used for data recovery, the other part is an encryption of a checksum of the data block used for searching. This structural construction may result in a linkability problem, in which a malicious sender may put a bogus checksum for a data block. We will discuss this problem in more detail for index-based search in the asymmetric-setting in next subsection.

5.2 Analysis of Index-based SED Schemes

In contrast to full-domain SED schemes, index-based SED schemes face other challenges although they suffer less from the above concerns. The index for a data item is constructed based on the related keywords. For many applications, such as the personal database outsourcing scenario in Section 2.1 and the email routing scenario in Section 2.2, it is likely that the keyword space is of limited size and the distribution of keywords is non-uniform. Moreover, the keyword set might be public knowledge. These factors may result in some security issues.

- By observing the client's search results, the server may infer the embedded keywords in search queries and the keyword information in the indexes. For instance, if a lot of search queries match the same index, then the server can infer that the index contains some keywords which have high distribution probabilities in the keyword set. To prevent the information leakage, the property of query result privacy is required.

- SED schemes in the asymmetric-setting are generally vulnerable to the offline message recovery attack and existing solutions such as those from [4, 39, 41, 42] suffer from efficiency and usability problems, as we have mentioned in Section 3. Regardless the aforementioned solutions, how to practically solve this issue is still an open problem.

Another concern is information leakage from indexes. Take the forward index shown in Fig. 2 of Section 3.1 as an example, suppose a SED scheme straightforwardly encrypts both the data items and their associated keywords. The server and any other entity can notice how many keywords are associated with each data item, and they may infer that the more keywords a data item has the more important it is to the client. The schemes from [11, 25] suffer from this problem. The scheme from [16] mitigates this problem by using a predefined and fixed dictionary of size x, and every index contains x bit showing whether each keyword is in the related data item or not. In the scheme from [20], it is required to estimate the upper bound of the possible keywords from all data items, then random values will be added in every index to hide keyword occurrence differences. For the inverted index shown in Fig. 3 of Section 3.1, if a SED scheme encrypts straightforwardly the keywords and their associated data items. Then, the server and any other entity know how many data items are associated to each keyword, and they may infer that the popularity of encrypted keywords. The issue becomes more obvious when the keywords are from a set of limited size and are not uniformly distributed. To mitigate the problem, a SED scheme (e.g. that from [18]) may try to add dummy values to the pad the indexes in the case of forward index or to pad item identifiers in the case of inverted index. Unfortunately, this mitigation measure may cause issues when the indexes need to be updated (e.g. new data items or new keywords to be added).

Yet another concern is the linkability issue between the data items and their indexes. In more detail, the potential problem is that a malicious party can try to generate contents and indexes in a malicious manner in order to gain some benefits. Here, the malicious party refers to any entity except for the receiver and the server, which will often be regarded as semi-trusted in practice. We illustrate an attack by taking the email routing scenario in Section 2.2 and a PEKS scheme [8] as example. In this scenario, any sender can send encrypted messages to the client and attached encrypted keywords to every message, and later on search can be done based on the encrypted keywords and message retrieval follows. In this situation, some message senders can always encrypt the keyword "urgent" and attach them to the their spams so that the client always retrieve the spams with a higher priority. All existing index-based SED schemes in the asymmetric-setting, e.g. [3, 4, 8, 9, 11, 25, 34, 42, 44], may suffer from this problem. Note that this problem falls beyond all existing security models. A straightforward countermeasure is to combine an index-based scheme with a full-domain scheme so that the server can filter the encrypted data items to identify malicious contents. However, it is an open problem to design a more elegant solution. In contrast, in the symmetric-setting, this may not be a problem because the client will generate the searchable contents by himself so that no third-party entities will be allowed to contribute anything.

In addition, for index-based SED schemes, how to update indexes has not been discussed much even though they are very practical issues in deploying SED schemes. For these schemes using a forward index structure, updating may be quite straightforward. But it may be more complicated for those using inverted index structure, since straightforwardly adding new index information to old indexes will leak information to the server. The authors from [16, 18] proposed methods to add new indexes, which will be generated based on new secret keys. As a result, old trapdoors cannot be used to search new indexes and it will result in different groups of indexes which can be only searched by certain trapdoors (generated based on different secret keys). This approach may cause problems in key management and search flexibility in reality.

5.3 More General Issues

Referring to the application scenarios described in Section 2, we have mentioned that the client may want a SED scheme which supports queries with flexible search criteria. For instance, exact matching, fuzzy matching, and conjunctive keywords should be supported at the same time. To our knowledge, no existing index-based SED scheme can provide this functionality. Generally speaking, almost all SED schemes only support certain types of search criteria. In order to support a variety of search criteria, the only way is to encrypt the data items with several SED schemes simultaneously, each of them is used for some search criteria. Besides the obvious increase in storage, this approach may result in additional information leakage because the server will learn some search pattern information (i.e. which search criteria has been invoked).

For most SED schemes other than those from [4, 26, 34], if an outsider attacker can eavesdrop on the transmission of encrypted contents and trapdoors, then it is as privileged as the server so that it may infer a lot of private information about the client. In [4, 34], the searchable contents are protected between message senders and the server. Clearly, this approach is very inflexible and not practical, because the client should trust all message senders who may not have the motivation to treat the issue seriously. In [26], the trapdoor transmission from the client to the server is encrypted so that an outsider attacker cannot obtain a useful trapdoor. However, the attacker may still eavesdrop on other communications such as the transmission of the search results, and it can still learn a lot about the index and trapdoor as shown in [42].

Except in [9, 21], query result privacy has not been touched upon in SED schemes, although it is a practical concern for many underlying application scenarios such as the personal database outsourcing scenario in Section 2.1. For a more concrete example, suppose that Alice stores both her work-related documents and personal documents on a remote server protected by a SED scheme. Moreover, she only queries her work-related documents in her office, and queries personal documents at home. One day, if the server notices at 10:00 pm that Alice is querying the same document as that she queried at 11:00 am, then the server can guess that Alice is working over the time in her office (it is likely that nobody is in her house). Due to the fact that the server is allowed to know the search results in existing SED schemes, there is no straightforward enhancement to provide query result privacy.

Besides confidentiality, data integrity (or, more specifically operation correctness) is another serious concern in practice, such as in all the scenarios described in Section 2. Due to a lot of reasons, a server may try to provide an incomplete or wrong result to the client. For example, the server may try to search 10 percent of the database in order to save computing resources. Another situation is that if some of the client's data has been lost, the server may try to provide some wrong result to the client to hide the truth. For data outsourcing in general, the concept of *proof of retrievability* or *proof of storage* has been investigated (e.g. [2, 13, 35]). But, specifically for SED, there has not been rigorous treatment although it is informally investigated in [15]. The investigation of data integrity in SED setting is an open research area.

6 Conclusion

In this chapter, we have categorized the existing SED schemes, reviewed their security models, and provided some practicality analysis. The brief analysis has showed that there are a lot of potential security issues facing SED schemes which are provably secure in their respective security models. How to resolve these identified issues may be interesting theoretical work in the future. Besides, there are other interesting future research directions. One is that a lot of experimental research work is needed to advance the research in the area of SED, despite very few efforts so far (e.g. [32]). Without real-world implementations, it is difficult to evaluate and compare the performances of existing SED schemes. The other is that, all algorithms are non-interactive in most of the definitions of SED, and it is interesting to investigate the case of allowing some algorithms to be interactive. On one hand, non-interactive algorithms are preferable than interactive ones because they do not require additional interactions between the client and the server. On the other hand, the requirement of non-interactive algorithms results in intensive usage of complicated structures (e.g. pairing operations) to have provable secure SED schemes. In addition, as shown in [9], allowing interactive algorithms can enable a SED scheme to achieve stronger privacy guarantees. Along this direction, it is also interesting to investigate the trade-off between the computational efficiency and communication efficiency.

References

1. R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Order preserving encryption for numeric data. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 563–574. ACM, 2004.
2. G. Ateniese, S. Kamara, and J. Katz. Proofs of storage from homomorphic identification protocols. In M. Matsui, editor, *Advances in Cryptology — ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 319–333. Springer, 2009.
3. J. Baek, R. Safavi-Naini, and W. Susilo. On the integration of public key data encryption and public key encryption with keyword search. In S. K. Katsikas et al., editor, *Proceedings of the 9th international conference on Information Security*, volume 4176 of *LNCS*, pages 217–232. Springer, 2006.
4. J. Baek, R. Safavi-Naini, and W. Susilo. Public key encryption with keyword search revisited. In O. Gervasi, editor, *Proceedings of the international conference on Computational Science and Its Applications, Part I*, volume 5072 of *LNCS*, pages 1249–1259. Springer, 2008.
5. F. Bao, R. H. Deng, X. Ding, and Y. Yang. Private query on encrypted data in multi-user settings. In L. Chen, Y. Mu, and W. Susilo, editors, *Proceedings of the 4th international conference on Information security practice and experience*, volume 4991 of *LNCS*, pages 71–85. Springer, 2008.
6. M. Bellare, A. Boldyreva, and A. O'Neill. Deterministic and efficiently searchable encryption. In A. Menezes, editor, *Advances in cryptology — CRYPTO 2007*, volume 4622 of *LNCS*, pages 535–552. Springer, 2007.
7. A. Boldyreva, N. Chenette, Y. Lee, and A. O'Neill. Order-preserving symmetric encryption. In A. Joux, editor, *Advances in Cryptology — EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 224–241. Springer, 2009.

8. D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano. Public Key Encryption with Keyword Search. In C. Cachin and J. Camenisch, editors, *Advances in Cryptology — EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 506–522. Springer, 2004.
9. D. Boneh, E. Kushilevitz, R. Ostrovsky, and W. E. Skeith III. Public key encryption that allows PIR queries. In A. Menezes, editor, *Advances in cryptology — CRYPTO 2007*, volume 4622 of *LNCS*, pages 50–67. Springer, 2007.
10. D. Boneh, A. Sahai, and B. Waters. Functional encryption: definitions and challenges. In Y. Ishai, editor, *Proceedings of the 8th conference on Theory of cryptography*, volume 6597 of *LNCS*, pages 253–273. Springer, 2011.
11. D. Boneh and B. Waters. Conjunctive, subset, and range queries on encrypted data. In S. P. Vadhan, editor, *Proceedings of the 4th conference on Theory of cryptography*, volume 4392 of *LNCS*, pages 535–554. Springer, 2007.
12. C. Bosch, Q. Tang, P. Hartel, and W. Jonker. Selective document retrieval from encrypted databse. In D. Gollmann and F. C. Freiling, editors, *Information Security Conference - 15th Information Security Conference (ISC 2012)*, volume 7483 of *LNCS*, pages 224–241. Springer, 2012.
13. K. D. Bowers, A. Juels, and A. Oprea. Proofs of retrievability: theory and implementation. In *Proceedings of the 2009 ACM workshop on Cloud computing security*, pages 43–54, 2009.
14. J. W. Byun, H. S. Rhee, H.Park, and D. H. Lee. Off-Line Keyword Guessing Attacks on Recent Keyword Search Schemes over Encrypted Data. In W. Jonker and M. Petkovic, editors, *Secure Data Management, Third VLDB Workshop*, volume 4165 of *LNCS*, pages 75–83. Springer, 2006.
15. Q. Chai and G. Gong. Verifiable symmetric searchable encryption for semi-honest-but-curious cloud services. Technical Report CACR 2011-22, University of Waterloo, 2011.
16. Y. Chang and M. Mitzenmacher. Privacy preserving keyword searches on remote encrypted data. In J. Ioannidis, A. D. Keromytis, and M. Yung, editors, *Proceedings of the Third international conference on Applied Cryptography and Network Security*, volume 3531 of *LNCS*, pages 442–455. Springer, 2005.
17. M. Chase and S. Kamara. Structured encryption and controlled disclosure. In M. Abe, editor, *Advances in Cryptology — ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 577–594. Springer, 2010.
18. R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky. Searchable symmetric encryption: improved definitions and efficient constructions. In *Proceedings of the 13th ACM conference on Computer and Communications Security*, pages 79–88. ACM, 2006.
19. T. Fuhr and P. Paillier. Decryptable searchable encryption. In W. Susilo, J. K. Liu, and Y. Mu, editors, *Provable Security, First International Conference, ProvSec 2007*, volume 4784 of *LNCS*, pages 228–236. Springer, 2007.
20. E. J. Goh. Secure indexes. Technical Report 216, IACR, 2003.
21. O. Goldreich and R. Ostrovsky. Software protection and simulation on oblivious RAMs. *Journal of ACM*, 43(3):431–473, 1996.
22. P. Golle, J. Staddon, and B. Waters. Secure conjunctive keyword search over encrypted data. In M. Jakobsson, M. Yung, and J. Zhou, editors, *Applied Cryptography and Network Security, Second International Conference*, volume 3089 of *LNCS*, pages 31–45. Springer, 2004.
23. D. Hofheinz and E. Weinreb. Searchable encryption with decryption in the standard model. http://eprint.iacr.org/2008/423, 2008.
24. B. Hore, S. Mehrotra, M. Canim, and M. Kantarcioglu. Secure multidimensional range queries over outsourced data. *The VLDB Journal*, 21(3):333–358, 2012.
25. Y. H. Hwang and P. J. Lee. Public key encryption with conjunctive keyword search and its extension to a multi-user system. In T. Takagi, editor, *Pairing-Based Cryptography, First International Conference*, volume 4575 of *LNCS*, pages 2–22. Springer, 2007.
26. L. Ibraimi, S. Nikova, P. H. Hartel, and W. Jonker. Public-key encryption with delegated search. In J. Lopez and G. Tsudik, editors, *Applied Cryptography and Network Security - 9th International Conference*, volume 6715 of *LNCS*, pages 532–549, 2011.
27. V. Iovino and G. Persiano. Hidden-vector encryption with groups of prime order. In S. D. Galbraith and K. G. Paterson, editors, *Pairing-Based Cryptography - Pairing 2008*, volume 5209 of *LNCS*, pages 75–88. Springer, 2008.

28. M. Islam, M. Kuzu, and M. Kantarcioglu. Access pattern disclosure on searchable encryption: Ramification, attack and mitigation. In *Proceedings of the Network and Distributed System Security Symposium, NDSS 2012*, page To appear. Internet Society, 2012.
29. J. Katz, A. Sahai, and B. Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In N. P. Smart, editor, *Advances in cryptology — EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 146–162. Springer, 2008.
30. F. Kerschbaum and A. Sorniotti. Searchable encryption for outsourced data analytics. In J. Camenisch and C. Lambrinoudakis, editors, *Public Key Infrastructures, Services and Applications - 7th European Workshop*, volume 6711 of *LNCS*, pages 61–76. Springer, 2011.
31. M. Kuzu, M. Islam, S. Mohammad, and M. Kantarcioglu. Efficient similarity search over encrypted data. In *Proceedings of the 2012 IEEE 28th International Conference on Data Engineering*, pages 1156–1167. IEEE Computer Society, 2012.
32. V. Pappas, M. Raykova, B. Vo, S. M. Bellovin, and T. Malkin. Private search in the real world. In *Proceedings of the 27th Annual Computer Security Applications Conference*, pages 83–92. ACM, 2011.
33. M. Raykova, A. Cui, B. Vo, B. Liu, T. Malkin, S. M. Bellovin, and S. J. Stolfo. Usable, secure, private search. *IEEE Security & Privacy*, 10:53–60, 2012.
34. H. S. Rhee, J. H. Park, W. Susilo, and D. H. Lee. Trapdoor security in a searchable public-key encryption scheme with a designated tester. *J. Syst. Softw.*, 83(5):763–771, 2010.
35. H. Shacham and B. Waters. Compact proofs of retrievability. In J. Pieprzyk, editor, *Advances in Cryptology — ASIACRYPT 2008*, volume 5350 of *LNCS*, pages 90–107. Springer, 2008.
36. E. Shen, E. Shi, and B. Waters. Predicate privacy in encryption systems. In O. Reingold, editor, *Proceedings of the 6th Theory of Cryptography Conference on Theory of Cryptography*, volume 5444 of *LNCS*, pages 457–473. Springer, 2009.
37. D. X. Song, D. Wagner, and A. Perrig. Practical Techniques for Searches on Encrypted Data. In *IEEE Symposium on Security and Privacy*, pages 44–55. IEEE Computer Society, 2000.
38. Q. Tang. Privacy preserving mapping schemes supporting comparison. In *Proceedings of the 2010 ACM workshop on Cloud computing security workshop*, pages 53–58, 2010.
39. Q. Tang. Towards public key encryption scheme supporting equality test with fine-grained authorization. In F. Martinelli and B. Preneel, editors, *Proceedings of the 16th Australasian Conference on Information Security and Privacy*, volume 6812 of *LNCS*, pages 389–406. Springer, 2011.
40. Q. Tang. Public key encryption schemes supporting equality test with authorization of different granularity. *International Journal of Applied Cryptography*, 2(4):304–321, 2012.
41. Q. Tang. Public key encryption supporting plaintext equality test and user-specified authorization. *Security and Communication Networks*, page To appear, 2012.
42. Q. Tang and L. Chen. Public-key encryption with registered keyword search. In *Proceeding of Public Key Infrastructure, 5th European PKI Workshop: Theory and Practice*, volume 6391 of *LNCS*, pages 163–178. Springer, 2009.
43. G. Yang, C. Tan, Q. Huang, and D. S. Wong. Probabilistic public key encryption with equality test. In J. Pieprzyk, editor, *Topics in Cryptology — CT-RSA 2010*, volume 5985 of *LNCS*, pages 119–131. Springer, 2010.
44. R. Zhang and H. Imai. Generic combination of public key encryption with keyword search and public key encryption. In J. Katz and M. Yung, editors, *Proceedings of the 6th international conference on Cryptology and network security*, volume 4521 of *LNCS*, pages 159–174. Springer, 2007.