cryptography

MDPI

Article

Practical Architectures for Deployment of Searchable Encryption in a Cloud Environment

Sarah Louise Renwick * and Keith M. Martin

Information Security Group, Royal Holloway, University of London, London TW20 0EX, UK; keith.martin@rhul.ac.uk

* Correspondence: sarahlouise.renwick.2012@live.rhul.ac.uk

Received: 15 September 2017; Accepted: 2 November 2017; Published: 15 November 2017

Abstract: Public cloud service providers provide an infrastructure that gives businesses and individuals access to computing power and storage space on a pay-as-you-go basis. This allows these entities to bypass the usual costs associated with having their own data centre such as: hardware, construction, air conditioning and security costs, for example, making this a cost-effective solution for data storage. If the data being stored is of a sensitive nature, encrypting it prior to outsourcing it to a public cloud is a good method of ensuring the confidentiality of the data. With the data being encrypted, however, searching over it becomes unfeasible. In this paper, we examine different architectures for supporting search over encrypted data and discuss some of the challenges that need to be overcome if these techniques are to be engineered into practical systems.

Keywords: searchable encryption; computing on encrypted data; functional encryption; cloud computing

1. Introduction

Nowadays, almost all data is stored digitally. Public cloud service providers provide an infrastructure that gives businesses and individuals access to computing power and storage space to support this digital data on a flexible pay-as-you-go basis. This allows them to bypass the costs associated with having their own data centres such as hardware, construction, air conditioning and security costs. This makes cloud computing a very cost-effective solution for both bulk data processing and data storage.

According to a government survey on information security and data breaches relating to digital data in 2015 [1], 90 percent of large organisations were attacked by an unauthorised outsider and suffered a data breach that year (up from 81 percent the previous year). Small businesses, which were previously not a major target, were also reporting increased attacks, with the number of these businesses attacked in 2015 rising by 14 percent from the previous year. As well as the increase in number of data breaches occurring, the financial costs to companies of these breaches, for example, due to business disruption, loss of sales and compensation, was also shown to have risen.

This demonstrates a clear need for secure data storage in order to ensure the confidentiality of data. However, encrypting data is just one part of the solution. Encrypting data ensures that, in the event of a compromise, no meaningful information should leak about the data itself if the data is compromised, yet it also reduces the possibility of performing computations on the ciphertexts, such as searching for keywords or specific items within the data.

Suppose that, just as in a typical cloud storage environment, your data is stored in encrypted form on a remote server due to local data storage constraints. To locate a piece of data, we could download the entirety of the encrypted data, decrypt it, and search over the unencrypted data. Alternatively, perhaps we could create an index for the encrypted data that is stored locally and used to navigate the encrypted files. Both of these methods provide adequate solutions in theory, yet, in practice, they

present several problems. Firstly, the size of the encrypted data may not be known a priori, or be known in advance to be very large, both of which deem the process of downloading all of the encrypted data highly inefficient and costly. Furthermore, the reason for storing the data remotely in the first place is due to the unavailability of local storage, so downloading all the data in this case would not be an option. Creating an index would require locally storing a file which may be of a size in the order of the number of encrypted files, which again is not feasible due to local storage restrictions. In addition, the index itself could potentially leak information about the encrypted data, compromising confidentiality.

Searchable encryption (SE) provides a solution to this problem by supporting the outsourcing of encrypted data to a remote server, whilst maintaining the ability to search for specific keywords within the encrypted data.

The literature regarding SE is extensive; however, SE is not widely deployed in practice. This chapter identifies and analyses different scenarios to which SE can be applied in the real world and investigates the suitability of certain types of SE schemes to each scenario. We also explore the reasons as to why SE schemes are not widely implemented and look at the security issues and functionality of protocols that are currently being implemented that achieve some form of search over encrypted data.

In this work, we analyze practical scenarios involving access to encrypted data and look at ways that SE could be used to solve problems within these scenarios. When analyzing the scenarios, we looked at features such as the number of users, the adversarial threat, sensitivity of the data involved and whether static or dynamic data is used, and assessed the suitability of particular SE schemes to each scenario. We used this research to define four basic scenarios to which SE could be applied based on the number of users and the capabilities of each user. Within each of these basic scenarios, we identify varying features of the scenarios that occur in the real world. We then map specific SE schemes into the different instances of the scenario depending on the varying features.

There is a comprehensive survey of provably secure searchable encryption schemes that post dates our initial research that can be found here [2]. They follow a similar categorization of SE schemes; however, this survey takes a theoretical approach and does not consider the practical reasons behind the categorisation of the SE schemes. We do not intend to provide a comprehensive survey of every SE scheme to date, and the intention here is to explore what features of an SE scheme might make it more suitable for use in a particular scenario in order to facilitate the design of protocols that use SE to provide solutions to real-world problems.

The rest of this article is organised as follows: in Section 2, we define the system and adversarial models for SE. Section 3 defines the four scenarios and analyzes the SE schemes in the literature, mapping them into these four scenarios according to various features of the schemes. Section 4 looks at SE schemes that are designed specifically to be deployed in the real world and discusses their various strengths and weaknesses. Section 5 looks at several factors that are preventing the deployment of SE. The article concludes with Section 6.

2. Searchable Encryption

In this section, we describe the generic system and adversarial models for SE.

2.1. System Model

Searchable Encryption allows an entity to search over encrypted data that they have outsourced to a remote server. In its simplest form, this involves locating encrypted data items on the server that contain a specified keyword. A searchable encryption scheme generally involves three entities:

- **Server:** A remote *server* (such as a pay-as-you-go cloud server) on which encrypted data is stored. We assume that the data stored on the server is of a sensitive nature, so it is undesirable to reveal it to any third parties, especially the server. Note that, in some cases, for example medical records, it may be illegal to reveal the records. We thus assume that the data items remain encrypted at all times when on the server.

- **Data Owner:** This entity initiates the scheme and holds the master secret key which is used to generate search queries in the single user case or user secret keys in the multi-user case (in the multi-user setting, the data owner is enrolled as the user and generates a user secret key in order to produce search queries). The *data owner* is always able to submit search queries to the server (referred to as *reading* data), as well as encrypting data to be stored on the server (referred to as *writing* data). The data owner is able to control which additional users (if any) are able to read or write data in the scheme.
- **User(s):** SE schemes contain a varying numbers of *users* that have the capability to either write data, read data, or do both of these tasks, as determined by the data owner.

The data owner associates metadata with each data item to be encrypted, usually consisting of keywords contained within the data item or keywords describing the data item's contents, and then creates an encrypted index using this metadata. The set of data items is encrypted separately using a standard encryption scheme, so the focus of searchable encryption is how the encrypted index is created. In this work, we do not consider the retrieval of the encrypted data items from the server. The encrypted index and encrypted data items are outsourced to the server and, in order to search for a keyword in the encrypted index, the data owner (or a user) produces a search query which the server uses to locate the relevant data items.

Figure 1 depicts the basic architecture of an SE scheme, which consists of the following five steps:

1. **Data encryption.** The data to be outsourced to the server is encrypted by the data owner (or a user, depending on the scenario) and sent to the server. In general, the encrypted data will consist of two parts:
   (a) *Encrypted data items*: The data items the data owner wishes to outsource to the server are typically encrypted using symmetric key cryptography. Each encrypted data item is identified by a unique identifier (ID).
   (b) *Secure index*: A *secure index* is created which enables the server to use a *search token* to learn the IDs of the relevant encrypted data items and locate them on the server.
2. **User's secret key transfer.** The data owner generates a secret key for a user and transfers it to them. The user is able generate search queries using their secret key.
3. **Search query.** A user (or data owner) generates a search query for a keyword ω and sends it to the server, allowing it to ascertain which encrypted data items satisfy their query.
4. **Search results.** The search results returned to the user can take two forms:
   (a) IDs of the encrypted data items that satisfy the query associated with the search token;
   (b) Encrypted data items that satisfy the query associated with the search token.

   The former response is advantageous over the latter in situations where the user does not require access to the actual data item (during statistical analysis of the encrypted data, for example), since it involves lower communication costs. The latter response involves the server locating the encrypted data items satisfying the query by locating them using the set of IDs, which increases the search time. In this paper, we consider schemes that generate search results consisting of IDs of encrypted data items that satisfy the query associated with the search token only; the retrieval of the actual data items is beyond the scope of this paper.

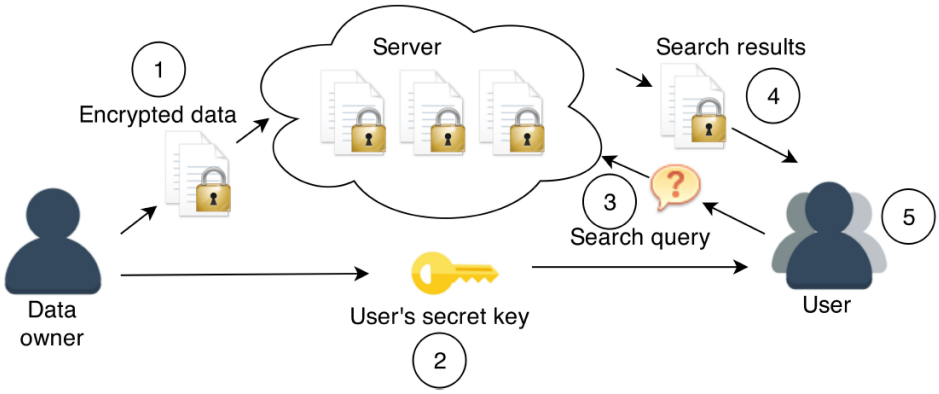

Figure 1. Searchable encryption model.

2.2. Adversarial Model

The server in a typical SE scheme is modelled as an 'honest-but-curious', meaning it will follow the designed protocol but might try to learn information that it is not authorised to know. It is also assumed that the server does not collude with users in order to gain access to encrypted data that is beyond their privileges. Furthermore, any data transferred between entities is visible to an external adversary.

The information that is leaked by an SE encryption scheme can be categorised as follows:

- **Access patterns**: This is the set of encrypted data items (or relevant IDs) that satisfy the search query. Hiding the access pattern requires that a set of search results for one search query be indistinguishable from that of another from the perspective of the server and/or an external adversary.
- **Search patterns**: These are the search queries that are made to the server. Hiding the search pattern requires that the search queries be indistinguishable from one another from the perspective of the server and/or an external adversary.
- **Secure index information**: This is information regarding keywords that are used to index the data items in the secure index. Hiding this information requires that two indexes be indistinguishable from one another from the perspective of the server and/or an external adversary. The typical security requirements of an SE scheme are that no information is leaked to the server or an external adversary from the secure index without a search query. it is assumed that no information is leaked from the encrypted data items. The data items are typically encrypted using a symmetric key encryption scheme, hence it is assumed that a suitable scheme is chosen so as to ensure no leakage from these ciphertexts occurs. Whereas leakage of the access and search patterns to the server and an external adversary are usually acceptable. We note that access pattern privacy can be achieved using private information retrieval (PIR) techniques [3]; however, as this increases the communication complexity of the search results, SE schemes do not usually employ these methods to conceal the access pattern.

3. Application of Searchable Encryption to Provide Solutions in Practical Scenarios

In this section, four basic scenarios for searching encrypted data in the cloud are described along with general methods of achieving SE that are potentially suitable for these environments. This section analyses SE schemes that achieve a single keyword equality search over encrypted data. The four basic scenarios are as follows:

1. **Scenario 1**: Only the data owner reads and writes all the data.
2. **Scenario 2**: Only the data owner can read the data, all users can write data.
3. **Scenario 3**: Only the data owner can write the data, many users can read the data.
4. **Scenario 4**: Many users (including the data owner) can both read and write data.

In Section 3, we describe each of these scenarios in detail, give a practical example of each scenario and present a generic method of SE that can be applied to each scenario, if applicable. In the real world, each scenario will present a number of variables that can affect which SE scheme would be best to use in each case. We consider the following variables:

- Size and type of data (static or dynamic),
- Frequency of queries,
- Number of additional users (if any),
- Sensitivity of data,
- Type of search query (equality, conjunctive etc.),
- Available local storage.

For each of these variables, we identify the most suitable SE scheme according to its construction and features.

3.1. Scenario 1: Only the Data Owner Reads and Writes All the Data

The simplest scenario is where only the data owner is involved in reading and writing data and there are no additional users in the system (Figure 2). The data owner sets up the system and generates the secret key. They create an encrypted index for the data items using the secret key and outsource it to the remote server along with the set of encrypted data items. The data owner generates search queries for the server using their secret key, which the server then uses to compute the search results satisfying the search query. The search results are then transferred to the user.

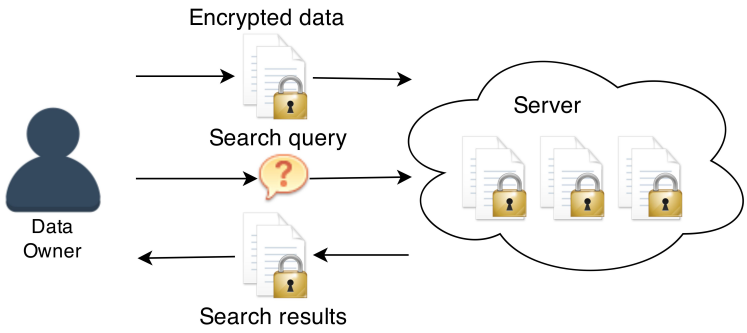

Figure 2. The data owner reads and writes all the data.

In practice, the data owner in this scenario could be the owner of a small business who wishes to outsource their confidential client data to a public cloud server due to an inadequate local storage capacity.

As there is only one entity reading and writing the data in this scenario, public-key encryption (PKE) does not offer advantages over more efficient symmetric-key encryption (SKE), as no secret key distribution is required. A generic method for performing SE in this scenario is symmetric searchable encryption (SSE) (see Definition A1). There are many different SSE schemes defined in the literature that support static data [4–10] and dynamic data [11–17], to provide a few notable examples.

- **Size and type of data.** If the dataset is very large, it will be beneficial to choose an SE scheme whose search time does not increase linearly with the size of the data set. SSE schemes that achieve a sublinear search time using a static dataset are [6,7]. The schemes of [6,11] only require $\mathcal{D}(\omega)$ symmetric decryptions per search, where $\mathcal{D}(\omega)$ is the number of data items satisfying the search query. Examples of schemes that support a dynamic data set and sublinear (with respect to the number of data items) search are that of [9,11,13,14]. The scheme of [14] is also parallellizable so it can achieve a search complexity of $\mathcal{O}(\frac{\mathcal{D}(\omega)}{p}\log n)$ where n is the number of encrypted data items, and p is the number of processors. The work of [12] has been designed specifically to support search over 'very large' databases.

- **Frequency of queries.** If a vast amount of search queries will need to be evaluated on the encrypted data, then choosing a scheme with efficient Query and Search algorithms will be beneficial. Fast search times can be produced using deterministic and order preserving encryption [18,19]. Searchable encryption schemes deterministic encryption can produce very fast (logarithmic) search times; however, in order to achieve this, some security is sacrificed. The index consists of deterministically encrypted keywords, hence repetitions in these keywords will be revealed to the server. Deterministic encryption in the symmetric-key setting is more secure against keyword guessing attacks than in the public-key setting. When the ciphertexts are created using a public key, an adversary is able to create arbitrary plaintexts of their choosing. Hence, if an adversary obtains a ciphertext, they can compare it to any of their arbitrarily created ciphertexts in order to determine the associated plaintext. In the symmetric-key setting, an adversary is unable to mount an attack like this as they do not have the secret key to generate any ciphertexts. For more expressive queries, such as range queries, order-preserving encryption can be used to achieve a fast search time. The security sacrifices for the faster search time using order preserving encryption is similar to those when using deterministic encryption. If the data owner just wishes to minimize their computation requirements, it will be beneficial to have a Query algorithm that is not so computationally intensive. The schemes of [4,9,14] only require the evaluation of one pseudo-random function (PRF) per search query, which makes their Query algorithm very efficient.
- **Number of additional users.** Not applicable to this scenario.
- **Sensitivity of data.** As outlined in Definition A4, ORAM can be used to perform searches over encryption without leaking the access pattern, a downfall of most SE schemes. However, these types of schemes can be very computationally expensive so they will not be suitable for very large datasets. Several schemes have been presented that improve the efficiency of the ORAM technique [20–24]. TWORAM, presented in [24], is the most promising ORAM-type searchable encryption scheme that can conceal the search pattern; however, it requires that the data items are stored in oblivious access memory in order to conceal the access pattern. This scheme achieves a sub-linear search time.
- **Type of search query.** The searchable encryption scheme suitable for this scenario with the most expressive queries is that of [25], which supports the following types of queries: conjunctive, disjunctive, polynomial and conjunctive/disjunctive normal formulae. This scheme uses bilinear maps in the construction of the index and the search queries, which are fairly computationally intensive for the user. Recent SSE schemes [12,26] can support arbitrary disjunctive and Boolean queries; however, the scheme of [26] is slightly more efficient in terms of search time, as, in the worst case, its searches are sublinear, whereas, for arbitrary disjunctive and Boolean queries, using [12] these types of searches are performed in linear time.
- **Available local storage.** In this setting, where only the data owner is reading and writing data to the remote server, if the data owner has local storage available, then an index for the encrypted data could be stored locally, un-encrypted. This would achieve the fastest possible search, as the data owner is searching un-encrypted data.

3.2. Scenario 2: Data Owner Can Read Data, All Users Can Write Data

In this scenario, both the data owner and a number of arbitrary other users are able to write data (Figure 3). The data owner, however, is the only party that can read the data.

This scenario is well illustrated by an email routing system. Users send encrypted emails to the data owner. To allow relevant routing of these emails, the server receives all the incoming emails and routes them according to the data owner's preferences. In order to ascertain how to route the encrypted emails, they will be securely indexed with keywords (these could be all words in the subject of the email, for example). The data owner supplies the server with search queries to enable it to detect which emails contain a specific keyword and then route them according to the data owner's instructions without revealing the plaintext email to the server.

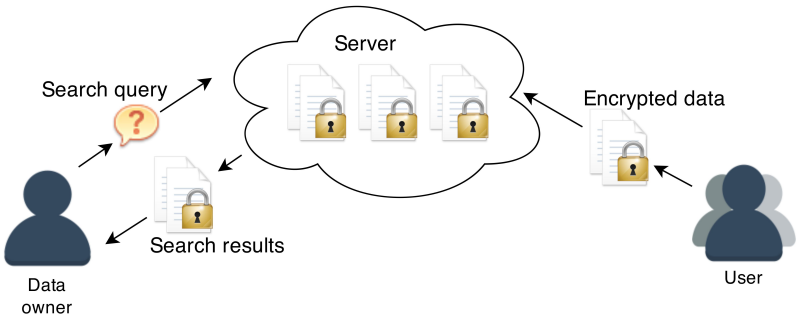

Figure 3. Data owner can read data, all users can write data.

As there is more than one party that can write data, SE schemes for this scenario need to be able to support dynamic data, as encrypted data will most likely be added to the server at different times by different parties. In terms of encryption, public-key searchable encryption (PKSE or PEKS) schemes are well suited here (see Definition A2). Using PKSE allows simple key distribution to the users, as encryption can be done using the data owner's public key. The secret key is used by the data owner to generate search queries. This also ensures that only the data owner is able to produce search queries and decrypt the data items.

- **Size and type of data.** As encrypted data items in this scenario are uploaded by different users, the index is necessarily a forward index, having an entry per data item. Subsequently, the search times are linear in the number of encrypted data items, which might not be suitable for practical systems. In addition, most of the public-key schemes suitable for this scenario use pairings and computationally intensive group operations in the Search and BuildIndex algorithms, which, for large datasets, may not be feasible for a practical system. The scheme of [27] does not use pairings in the BuildIndex or Search, which may make it a more practical choice.
- **Frequency of queries.** Due to the computationally intensive primitives used in the algorithms in public-key searchable encryption schemes, they might not be suitable for use in scenarios where a high frequency of queries are issued to the server. The schemes of [27–29] do not use group operations or pairings in their Query algorithms. They use hash functions [28,29] and polynomial evaluation [27].
- **Number of additional users.** As the data is encrypted using a public key, no secure channel is needed for key distribution; hence, a large number of additional users does not impact the efficiency of the scheme in this sense. However, more users may imply more encrypted data being stored on the server so it could indirectly affect the efficiency of the scheme due to the issues arising from searching over very large datasets (see previous point regarding size and type of data).
- **Sensitivity of data.** Public-key searchable encryption schemes are vulnerable to the offline keyword guessing attack. In this attack, an adversary is able to uncover the keyword(s) that were used to generate the search query. As the search queries are generated using a public key, the adversary is able to generate arbitrary search queries for any keyword(s) of their choosing and evaluate them on the encrypted data. By comparing the search results generated by their target search query and their arbitrarily generated search queries, they are able to break the security of the search query by seeing which search queries produce identical search results. The schemes of [30,31] are not susceptible to this attack; however, Ref. [30] involves a user registering a keyword prior to encrypting data, so there is an extra round of communication between the user and the data owner prior to encryption. The scheme of [31] requires an extra server in order to prevent offline keyword guessing attacks, with the assumption that the two servers do not collude.
- **Type of search query.** The most expressive searchable encryption schemes for this scenario are that of [32,33]. The scheme presented in [32] supports conjunctive, subset and range queries, whereas Ref. [33] supports the following types of queries: conjunctive, disjunctive, polynomial and conjunctive/disjunctive normal formulae.

- **Available local storage.** As the encrypted data is sent straight to the server by the users, in order to index the data locally, the data owner will need to download the encrypted data in order to index it. This will need to be repeated every time an encrypted data owner is added by a user. Due to the large communication costs associated with this method, we do not consider this a feasible option.

3.3. Scenario 3: Data Owner Can Write Data, Many Users Can Read Data

In this scenario, all the encrypted data stored on the server is written by the data owner, who wishes to provide read capabilities to other users, whilst not allowing them the ability to write data (Figure 4). As only the data owner is writing data, SKE is a good choice for constructing SE schemes for this scenario and the data could be either static or dynamic.

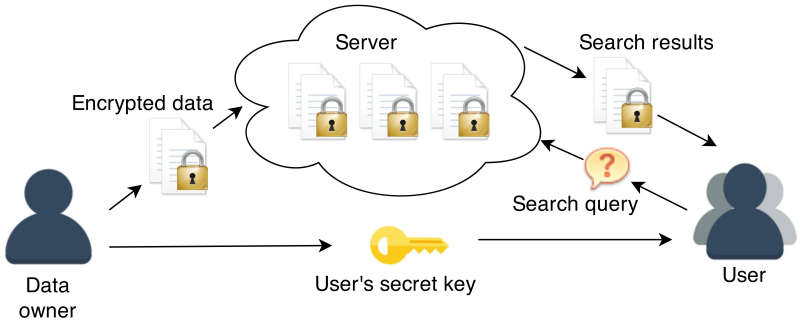

Figure 4. Data owner can write data, many users can read data.

In practice, a corporation may want to outsource some encrypted data to a remote cloud server. In order to allow employees to read the data, an SE scheme that supports multiple readers is required. This group of users could also be dynamic, meaning that users may get their read capabilities revoked, when an employee is fired for example, or new users may be enrolled into the system when the corporation hires new employees.

Although this scenario seems to reflect many real-world instances of searching on encrypted data, the literature is not very vast.

- **Size and type of data.** Using broadcast encryption (BE) (see Definition A3), as a method for read-only user addition and revocation was put forward in [7]. Using this method, a single user SSE scheme can be converted to a multi-user one, inheriting the properties of the underlying single-user scheme. Choosing a suitable single-user SSE scheme (see Section 3.1) and applying a broadcast encryption scheme to it would provide a good solution here.
- **Frequency of queries.** The solution described in the point above also applies here.
- **Number of additional users.** If there is a dynamic user group searching the encrypted data, then the user addition and revocation mechanisms will need to be efficient. The scheme of [34] stores a unique key per user on the server, which is used together with a search query to search the encrypted data. To revoke a user, the user's key is deleted from the server, which prevents the server from being able to execute the user's search queries. The revocation in this scheme does not effect the other authorised users.
- **Sensitivity of data.** There are no ORAM solutions specifically for this scenario. One could use a broadcast encryption scheme along with the single user ORAM in order to allow multi-users to query the encrypted data.
- **Type of search query.** Using broadcast encryption extends a single-user SSE scheme to multi-user one, hence the single- user schemes detailed in Section 3.1 are also applicable here when combined with a broadcast encryption scheme to handle user revocation and addition.

- **Available local storage.** If users have available local storage, then each one could store an index locally, if trusted by the data owner. This would work most efficiently when using a static dataset, as any updates to the data would require the data owner to update all the user indexes. If the additional users do not have equal access rights, then it would be possible using this method to have a different index for each user, so the user can only learn information regarding data items they are authorized to search.

3.4. Scenario 4: Many users can read and write data

In this scenario, many users including the data owner have read and write capabilities (Figure 5).

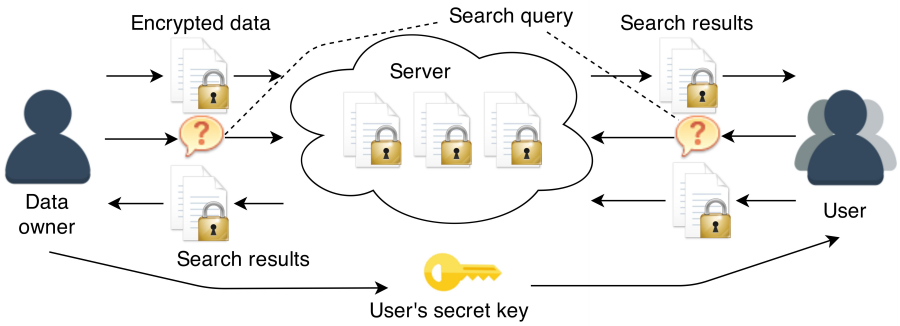

Figure 5. Many users can read and write data.

An example of this scenario in the real world is the management of encrypted electronic health records (EHRs). If you take just one hospital, there are numerous doctors, nurses and other staff (which represent the users and data owners) that write information to the EHRs. Due to the sensitive nature of the data, access will be restricted to certain users only. For example, a general practitioner may be authorised to read the records of their patients only, whereas a heart specialist may be authorised to read all records relating to heart conditions. All medical staff may need to write data to the records.

As there is more than one entity writing data, as in Section 3.2, we need to support dynamic data. Both SKE and PKE are possible in this scenario but produce a subtle difference in the SE schemes functionality. In a scheme that uses SKE, a revoked user will no longer be able to read or write data, whereas, when using PKE, a revoked user will no longer have read capabilities; however, they will still be able to write data to the server as this is done using the public key. We shall look at PKE and SKE based methods of SE in this scenario.

- **Size and type of data.** For large datasets, deterministic encryption [18] could be used along with a proxy server that uses proxy re-encryption to transform user's queries to the correct form. Each user could encrypt data using their own public key. In order to search other user's encrypted data, the proxy server can convert search queries to the correct form depending on which user encrypted the data. This could involve the proxy server computing many different search queries in order for a user to search many different user's data, so this method might not scale well to a large number of users.
- **Frequency of queries.** A scheme by Bao et al. [3] allows multiple readers as well as writers using SKE. It employs a trusted authority to generate keys and enroll/revoke users in the system. The method used to add and revoke users is similar to that using BE; however, generating a search query and writing a data item to the server requires a valid value that varies from user to user. The server stores information regarding these values and authorized users in a dynamic list. Using SKE makes the Query and Search algorithms more efficient, hence this method might be suitable for a system with a high volume of queries.
- **Number of additional users.** Multi-receiver public key encryption with keyword search (multi-receiver PEKS) [28] extends the PEKS scheme to a setting that allows multiple readers

as well as writers. Each user in the scheme has their own public key. For a large number of additional users, this scheme is suitable as user keys can be transferred to the users publicly.

- **Sensitivity of data.** If the data is very sensitive, the ORAM techniques may be applicable. A multi-user ORAM scheme is presented in [35], which uses a chain of proxy servers to protect the data access pattern of each user. Another multi-user ORAM scheme is presented in [20]; however, this involves an ORAM construction per user and only supports a user reading data from their own ORAM construction, whereas they can write data to other user's ORAM constructions.
- **Type of search query.** Wang et al. [36] present a conjunctive scheme suitable for this scenario. No other schemes supporting more expressive search queries are detailed in the literature.
- **Available local storage.** As the encrypted data is sent straight to the server by the users, in order to index the data locally, the data owner will need to download the encrypted data in order to index it. This will need to be repeated every time an encrypted data owner is added by a user. Due to the large communication costs associated with this method, we do not consider this a feasible option.

4. Searchable Encryption in the Real World

In this section, we give a brief analysis of some products that claim to support searching outsourced encrypted data. We try to detail the type of encryption used and the setup and search leakages, although, as some of these products are proprietary, full details of the encryption method may not be available.

- **CipherLocker**: A product named CipherLocker which has been created by a company called Private Machines Inc. claims to be 'the first platform that lets you securely search through files that are stored encrypted on the server, without having to download the files' [37]. The model used is different to that of standard SE (Section 2.1), as the index is not kept on the server. The data items themselves are encrypted using a semantically secure authenticated encryption scheme before they are transferred to the server for storage. The searches are performed locally by the user, hence there is no related leakage from the searches to the server. CipherLocker provides a secure way to locate and retrieve encrypted data items from a server, the model is very different to that of SE that is discussed in this paper. We can say that if every SE scheme stored an index and performed the search locally, then all schemes would achieve this high level security. In the literature, a standard assumption is made that a data owner wishes to outsource sensitive data to a remote server due to local storage restraints, hence CipherLocker may not be a suitable solution to searching on encrypted data for everyone.
- **CryptDB**: CryptDB [38] was developed by Popa et al. and is one of the most well-known and documented solutions to searching on encrypted data. The authors are very transparent about the types of encryption used in CryptDB along with the associated leakages, unlike other products discussed in this section. This scheme works on data in SQL databases and allows the evaluation of SQL queries over the encrypted data. Using CryptDB, it is possible to encrypt different parts of the database using different types of encryption, depending on what types of queries will be evaluated on that part of the database. This scheme is a good example of the relationship between functionality and security that is often considered when designing an SE scheme.
  - **Random (RND)**: This is the strongest form of encryption used in CryptDB and produces ciphertexts that are indistinguishable under a chosen plaintext attack (IND-CPA secure). It does not allow any computation on parts of the database encrypted with this form of encryption.
  - **Deterministic (DET)**: Deterministic encryption [18] leaks what values are equal in the encrypted data i.e., equal plaintexts produce equal ciphertexts. Using this type of encryption, one can perform equality checks on the encrypted data, which enables the user to locate encrypted that matches a target value.

- **Order preserving encryption (OPE)**: OPE [39] leaks the order relations between ciphertexts. If we have that a plaintext x is less than that of a plaintext y, then, if using OPE, the resulting ciphertexts c_x, c_y will retain this order relation i.e., c_x is less than c_y. Using OPE to encrypt the data, one can perform order queries and determine the maximum and minimum values within the encrypted data.
- **Homomorphic encryption (HOM)**: Homomorphic encryption [40] produces ciphertexts that are IND-CPA secure and allow add queries to be performed over the encrypted data. The leakage is limited to size of the ciphertexts. The security for this form of encryption is similar to that of RND.
- **Word Search (SEARCH)**: The scheme that is used to perform text search is that of [10]. The text is split into individual keywords and the repetitions are removed and the words permuted. Each keyword is encrypted using the scheme as defined in [10]. The ciphertexts are then padded to ensure they are all the same size. The leakage is limited to which data matches the search query. If there are RND encryptions of the same data, then an adversary may be able to detect the number of distinct keywords in the data by comparing the size of the SEARCH ciphertexts with the RND ciphertext.

CryptDB performs much more than keyword search, as we have detailed. It also provides flexibility for the user to assess what kind of security they need on their data and whether they would like to sacrifice leaking some information to the server in exchange for being able to evaluate more expressive queries. It requires a proxy between the user and the server in order to translate the SQL queries to be compatible with the encrypted data, as well as decrypt the query results before transferring them to the user. CryptDB has been used by several companies including Google (Mountain View, CA, USA), SAP (Newtown Square, PA, USA), Lincoln Laboratory (Lexington, MA, USA), Skyhigh (Campbell, CA, USA) and Microsoft (Redmond, WA, USA) [41]. Recently, the security claims of CryptDB have been called into question. Work by Naveed et al. [42] present attacks on DET and OPE used in CryptDB where a plaintext can be fully recovered when using these forms of encryption, given only the encrypted column and publicly available auxiliary information.

- **CipherCloud**: Ciphercloud market a type of encryption called *Searchable Strong Encryption*, which is not the same as the notion of 'SSE' discussed in this paper. It involves a local index being stored by the user in order to support searching, similarly to Cipherlocker. They also use tokenization to support search on encrypted data. They claim that in some countries that have strict data residency laws, such as Singapore and Luxembourg, companies are not allowed to encrypt their data, hence tokenization is used instead of encryption for clients in these countries to comply with regulations whilst still somewhat protecting their data.

Although any form of encryption on the data is more secure than nothing at all, it is obvious that the claims of some companies that supply products that support searchable encryption are slightly misleading for prospective clients.

5. Deployment Challenges

Despite the existence of seemingly practical searchable encryption schemes, there is still scarce existence of their deployment Some companies claim that they can achieve searchable encryption, but, with further investigation into their systems, they are usually built using a technology called *tokenization*. Tokenization works by replacing a plaintext string (a keyword for example) by a completely random string (*token*) before uploading it to the server and storing a map of these replacements locally. Although the tokens are chosen randomly, identical plaintext strings are mapped to the same token, meaning equality between plaintexts will be leaked. This will provide a fast search (same as searching a plaintext) but requires a lot of data (the mapping) to be stored locally, meaning, if the entirety of the data is tokenized, then there will be just as much data locally stored (excluding

repeated keywords) as there is on the server. This method of data confidentiality seems to be more suited to scenarios where only parts of the outsourced data needs to be protected. For example, in a database of customer details, only the very sensitive portions of the data such as social security numbers might need protecting. The level of security provided by tokenization is roughly equivalent to that of DE, yet it also has the requirement of local storage space, as the user needs to store a dictionary for all the tokenized values. SSE, although only achieving a fairly course grained search, is still able to achieve sublinear search times and a high level of security, so why are companies and consumers not adopting these methods to achieve data confidentiality in the cloud environment? We discuss several reasons that we believe form barriers to the adoption of the cryptographic techniques for searching encrypted data detailed in this work.

Usability. Searching over data in the clear is an everyday task for most people. Users are accustomed to the efficient and highly functional search facilities found with search engines such as Google. As a result, they expect the same levels of usability from an encrypted search. Unfortunately, the SE schemes in the literature do not obtain this combination of functionality with efficiency; they cannot compete with the search tools that users are accustomed to in this respect. This ties in with the point regarding user education, if users understood the security benefits of using SE compared with other methods, such as tokenization, then it might make SE a more appealing choice.

Lack of prototypes. SE is a relatively new technology, and there exists few prototypes or demonstrations to illustrate the successful adoption of SE in the real world. This makes the implementation of SE into new products or systems over established methods highly risky for the user. Alongside this is the issue of the lack of standardised algorithms and protocols relating to SE, which make it harder to implement without specialised knowledge in the area of SE. There are a wide variety of SE schemes available that are well suited to different scenarios; however, without any evidence demonstrating the suitability of a scheme to a particular scenario, there is no motivation for the user to choose this new technology over established methods. This ties into the next point regarding user education.

User education. Users may not understand how SE can be used to solve their problem or the security benefits it provides over existing methods. Furthermore, users may not understand the security issues associated with using an alternative method such as tokenization. Educating users on these issues will help them make a more informed choice and prevent the implementation of a method that is not fit for their purpose.

Legal issues. The Investigatory Powers Bill [43], which, if passed, would give new powers to law enforcement allowing them to issue subpoenas to cloud providers storing encrypted data requiring them to break the encryption and release the plaintext data. In the case of SE where the encrypted data stored on the company's server is encrypted and outsourced by a third party and not by the cloud provider itself, they would be unable to provide the plaintext data in this case. This could prevent the deployment of SE as the cloud providers would be unable to store third party encrypted data, as, if requested by law enforcement, they would be unable to provide the plaintext data.

Compliance. It is obvious that SE would be of great benefit to users or institutions that handle a lot of sensitive data, such as banks. However, new technologies such as SE are effectively prevented from being deployed, as they do not comply with these institution's regulations, which usually require the use of legacy systems. Institutions also have policies in place regarding how data is to be handled and how their systems are used. Failure to comply with these regulations by an employee would result in disciplinary or legal action against the employee. It is believed that this provides enough of a deterrent to a potential inside adversary and provides a means for keeping sensitive data confidential without the use of encryption. However, data breaches are becoming more frequent and severe [1], which motivates the need for provably secure methods of data protection as opposed to ones that rely on trust and user compliance. There are laws in place dictating that access to particular types of sensitive data, such as medical records, should be restricted. The laws do not dictate how to enforce such a restriction however. These laws should be updated to specify how the data should be protected, in order to eliminate the reliance on compliance for data security.

Lack of communication between academia and industry. In researching for this work, we encountered firsthand experience of this issue. There is a distinct lack of dialogue between the academics that conduct research in the area of SE and the practitioners that would like to implement the technology. Without a conversation between these two parties, academic researchers do not know which features of SE are most desirable for a practitioner and hence may not focus on these features in their research. In order to facilitate the development of SE schemes that meet the needs of practitioners, there needs to be a more open dialogue between academia and industry. Some companies (Sky High, for example) have started to address this issue by appointing a cryptographic advisory board consisting of five leading academic researchers. This gives the company access to information regarding cutting edge technologies and specialised advice, and, on the other hand, gives the academic researchers problems and areas of research to focus their work on.

6. Conclusions

In this work, we defined four scenarios in the real world in which searchable encryption can be applied. We discuss various features of SE schemes and map suitable SE schemes into the four scenarios. In addition, we discuss some issues that are preventing the adoption of SE in the real world. We hope that this work will help practitioners design products that implement SE over less secure methods such as tokenization, by highlighting suitable SE solutions and discussing the trade-offs between security and efficiency within these scenarios.

Acknowledgments: Sarah Louise Renwick acknowledges support from Thales UK and the Engineering and Physical Sciences Research Council (EPSRC) under a Cooperative Award in Science and Technology (CASE).

Author Contributions: The work in this paper formed part of the Ph.D. research project of Sarah Louise Renwick, working under the supervision of Keith M. Martin. The ideas in this paper were jointly developed. Sarah Louise Renwick is the primary author of the paper.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

SE	Searchable encryption
SSE	Symmetric Searchable Encryption
SKE	Symmetric-Key Encryption
PKE	Public-Key Encryption
BE	Broadcast Encryption

Appendix A. Definitions

Definition A1 (Symmetric Searchable Encryption (SSE))**.** *A searchable symmetric encryption scheme typically consists of the following four algorithms (we assume the data items themselves are encrypted separately using a secure encryption scheme):*

- $SK \xleftarrow{\$} \mathsf{KeyGen}(1^\kappa)$*: A probabilistic algorithm run by the data owner that takes as input the security parameter* $1^\kappa \in \mathbb{N}$ *and outputs a secret key,* $SK \in \mathcal{K}$*.*
- $\mathcal{I}_\mathcal{D} \xleftarrow{\$} \mathsf{BuildIndex}(\mathcal{D}, SK)$*: This algorithm can be probabilistic or deterministic as required and it is run by the data owner. It takes as input the set of metadata* $\mathcal{D}$ *associated with each data item along with the secret key SK and outputs a secure index* $\mathcal{I}_\mathcal{D}$*.*
- $T_\omega \leftarrow \mathsf{Query}(\omega, SK)$*: A deterministic algorithm run by the data owner to produce a search query. It takes as input a keyword* $\omega \in \Delta$ *along with SK and outputs a search query* T_ω*.*
- $R_\omega \leftarrow \mathsf{Search}(I_D, T_\omega)$*: A deterministic algorithm run by the server to produce the search results. It takes as input the secure index* $\mathcal{I}_\mathcal{D}$ *and a search query* T_ω *and outputs the search results* R_ω*.*

Definition A2 (Public Key Searchable Encryption (PKSE)). *A public key searchable encryption scheme typically consists of the following four algorithms (we assume the data items themselves are encrypted separately using a secure encryption scheme):*

- $(PK, SK) \xleftarrow{\$} \mathsf{KeyGen}(1^\kappa)$*: A probabilistic algorithm run by the data owner that takes as input the security parameter* $1^\kappa \in \mathbb{N}$ *and outputs a public and secret key pair,* $(PK, SK) \in \mathcal{K}$.
- $\mathcal{I}_\mathcal{D} \xleftarrow{\$} \mathsf{BuildIndex}(\mathcal{D}, PK)$*: This algorithm can be probabilistic or deterministic as required and is run by either the data owner or an additional user. It takes as input the set of metadata* $\mathcal{D}$ *associated with each data item along with the secret key* SK *and outputs a secure index* $\mathcal{I}_\mathcal{D}$.
- $T_\omega \leftarrow \mathsf{Query}(\omega, SK)$*: A probabilistic algorithm run by the data owner to produce a search query. It takes as input a keyword* $\omega \in \Delta$ *that the data owner or user wishes to search for along with* SK *and outputs a search query* T_ω.
- $R_\omega \leftarrow \mathsf{Search}(PK, I_D, T_\omega)$*: A deterministic algorithm run by the server to produce the search results. It takes as input the secure index* $\mathcal{I}_\mathcal{D}$ *and a search query* T_ω *and outputs the search results* R_ω.

Definition A3 (Broadcast Encryption (BE)). *A Broadcast encryption (*BE*) consists of the following four algorithms:*

- $(PK, SK) \xleftarrow{\$} \mathsf{Keygen}(1^\kappa, h)$*: A probabilistic algorithm that takes a security parameter* $\kappa \in \mathbb{N}$ *(and possibly a value* h*, which determines the maximum number of users that can be revoked from the system), and outputs a public and secret key pair* $(PK, SK) \in \mathcal{K}$.
- $SK_u \xleftarrow{\$} \mathsf{Add}(SK, u)$*: A probabilistic algorithm that takes a user identity* $u \in \mathcal{U}$ *and the secret key* SK*, and outputs a secret key* $SK_u \in \mathcal{K}$.
- $c \xleftarrow{\$} \mathsf{Encrypt}(m, \mathcal{G}, PK)$*: A probabilistic algorithm that takes a plaintext* m*, a set of users* $\mathcal{G} \subseteq \mathcal{U}$ *that are authorized to decrypt the resulting ciphertext and the public key* PK*. It outputs a ciphertext* c.
- $(m \text{ or } \perp) \leftarrow \mathsf{Decrypt}(SK_u, c)$*: A deterministic algorithm run by a user that takes a ciphertext* c *and the user's secret key* SK_u*. The algorithm outputs either a plaintext* m *that was encrypted in* c *if the user is authorised to decrypt* c *(i.e., if* $u \in \mathcal{G}$*), or the failure symbol* $\perp$ *otherwise.*

Definition A4. *Oblivious RAM (ORAM), introduced by Goldreich and Ostrovsky [44], allows a user to retrieve data items from a server without revealing to the server which data items were retrieved (access pattern). In order to mask the access pattern, each access to the data is designed to be indistinguishable to an adversary. Supposing the user stores* n *data items on the server, in order to achieve the masking of the access patterns the user is required to store an extra* $\sqrt{n}$ *dummy data items and a shelter storing another* $\sqrt{n}$ *data items. First off, a random permutation* π *is applied to the data items at position* $(1, ..., n + \sqrt{n})$*, i.e., everything but the shelter. In order to access the* i*th data item, the shelter is scanned to check whether to check whether the* i*th data item is in the shelter (this will not be the case on the first access; however, after each access, the contents of the access are stored in the shelter). The items in the shelter are accessed in a predetermined order, regardless of whether the desired data item has been located. If* i *is not found in the shelter, then it is retrieved by calculating* $\pi(i)$*, which is the location of the desired data item. If* i *was located in the shelter, then the next dummy data item in the permuted memory is accessed at* $\pi(n + j)$*, where* j *is the current step in the access). In either case, the retrieved data item, the dummy one or the actual one is written to the shelter, by scanning all of the items in the shelter again. The contents of the shelter are returned to the user. This ORAM solution is called the quadratic solution. Many other methods for achieving ORAM have been put forward in order to enhance its efficiency [20–24]. Using ORAM to achieve searchable encryption completely conceals the access pattern, making it the most secure method for searchable encryption. However, due to the high computational costs incurred by the multiple accesses per search, it does not scale well to encrypted search over a large database.*

References

1. PWC. *2015 Information Security Breaches Survey*; Technical Report; PWC: London, UK, 2015.
2. Bösch, C.; Hartel, P.H.; Jonker, W.; Peter, A. A Survey of Provably Secure Searchable Encryption. *ACM Comput. Surv.* **2014**, *47*, 18.
3. Bao, F.; Deng, R.H.; Ding, X.; Yang, Y. Private Query on Encrypted Data in Multi-user Settings. In Proceedings of the 4th International Conference on Information Security Practice and Experience, Sydney, Australia, 21–23 April 2008; Volume 4991, pp. 71–85.
4. Goh, E.J. Secure Indexes. In *IACR Cryptology ePrint Archive*; Report 2003/216; International Association for Crytologic Research, 2003. Available online: https://eprint.iacr.org/2003/216 (assessed on 16 March 2004).
5. Chang, Y.; Mitzenmacher, M. Privacy Preserving Keyword Searches on Remote Encrypted Data. In Proceeedings of the Third International Conference on Applied Cryptography and Network Security (ACNSS), New York, NY, USA, 7–10 June 2005; Volume 3531, pp. 442–455.
6. Chase, M.; Kamara, S. Structured Encryption and Controlled Disclosure. In Proceedings of the Advances in Cryptology—ASIACRYPT 2010—Theory and Application of Cryptology and Information Security, Singapore, 5–9 December 2010; Volume 6477, pp. 577–594.
7. Curtmola, R.; Garay, J.A.; Kamara, S.; Ostrovsky, R. Searchable Symmetric Encryption: Improved Definitions and Efficient Constructions. In Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS), Alexandria, VA, USA, 30 October–3 November, 2006; ACM: New York, NY, USA; 2006; pp. 79–88.
8. Kurosawa, K.; Ohtaki, Y. UC-Secure Searchable Symmetric Encryption. In Proceedings of the 16th International Conference on Financial Cryptography and Data Security, Kralendijk, Bonaire, 27 February–2 March 2012; Volume 7397, pp. 285–298.
9. Liesdonk, P.V.; Sedghi, S.; Doumen, J.; Hartel, P.H.; Jonker, W. Computationally Efficient Searchable Symmetric Encryption. In Proceedings of the 7th VLDB Workshop—Secure Data Management (SDM), Singapore, 17 September 2010; Volume 6358, pp. 87–100.
10. Song, D.X.; Wagner, D.; Perrig, A. Practical Techniques for Searches on Encrypted Data. In Proceedings of the 2000 IEEE Symposium on Security and Privacy, Berkeley, CA, USA, 14–17 May 2000; pp. 44–55.
11. Kamara, S.; Papamonthou, C.; Roeder, T. Dynamic searchable symmetric encryption. In Proceedings of the ACM Conference on Computer and Communications Security, CCS'12, Raleigh, NC, USA, 16–18 October 2012; pp. 965–976.
12. Cash, D.; Jaeger, J.; Jarecki, S.; Jutla, C.; Krawczyk, H.; Rosu, M.; Steiner, M. Dynamic Searchable Encryption in Very-Large Databases: Data Structures and Implementation. In Proceedings of the 21st Annual Network and Distributed System Security Symposium (NDSS), San Diego, CA, USA, 23–26 February 2014.
13. Cash, D.; Jarecki, S.; Jutla, C.; Krawczyk, H.; Rosu, M.; Steiner, M. Highly-Scalable Searchable Symmetric Encryption with Support for Boolean Queries. In Proceedings of the 33rd Annual Cryptology Conference on Advances in Cryptology—CRYPTO 2013, Santa Barbara, CA, USA, 18–22 August 2013; Volume 8042, pp. 353–373.
14. Kamara, S.; Papamonthou, C. Parallel and Dynamic Searchable Symmetric Encryption. In Proceedings of the 17th International Conference on Financial Cryptography and Data Security, Okinawa, Japan, 1–5 April 2013; Volume 7859, pp. 258–274.
15. Naveed, M.; Prabhakaran, M.; Gunter, C. Dynamic Searchable Encryption via Blind Storage. In Proceedings of the 2014 IEEE Symposium on Security and Privacy (SP), San Jose, CA, USA, 18–21 May 2014; pp. 639–654.
16. Stefanov, E.; Papamonthou, C.; Shi, E. Practical Dynamic Searchable Encryption with Small Leakage. In Proceedings of the 21st Annual Network and Distributed System Security Symposium (NDSS), San Diego, CA, USA, 23–26 February 2014.
17. Pappas, V.; Krell, F.; Vo, B.; Kolesnikov, V.; Malkin, T.; Choi, S.G.; George, W.; Keromytis, A.; Bellovin, S. Blind Seer: A Scalable Private DBMS. In Proceedings of the 2014 IEEE Symposium on Security and Privacy (SP), San Jose, CA, USA, 18–21 May 2014; pp. 359–374.
18. Bellare, M.; Boldyreva, A.; O'Neill, A. Deterministic and Efficiently Searchable Encryption. In Proceedings of the Advances in Cryptology—CRYPTO 2007, Santa Barbara, CA, USA, 19–23 August 2007; Volume 4622, pp. 535–552.

19. Boldyreva, A.; Chenette, N.; Lee, Y.; O'Neill, A. Order-Preserving Symmetric Encryption. In Proceedings of the Advances in Cryptology—EUROCRYPT 2009, Cologne, Germany, 26–30 April 2009; Volume 5479, pp. 224–241.
20. Blass, E.; Mayberry, T.; Noubir, G. Multi-User Oblivious RAM Secure Against Malicious Servers. In *IACR Cryptology ePrint Archive*; Report 2015/121; International Association for Cryptologic Research, 2015. Available online https://eprint.iacr.org/2015/121/20150528:122820 (accessed online 28 May 2015).
21. Chan, H.T.; Shi, E.; Wang, X. Circuit ORAM: On Tightness of the Goldreich-Ostrovsky Lower Bound. In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, 12–16 October 2015; pp. 850–861.
22. Devadas, S.; Fletcher, C.W.; Ren, L.; Shi, E.; Stefanov, E.; van Dijk, M.; Yu, X. Path ORAM: An Extremely Simple Oblivious RAM Protocol. In Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, 4–8 November 2013; pp. 299–310.
23. Devadas, S.; van Dijk, M.; Fletcher, C.W.; Ren, L.; Shi, E.; Wichs, D. Onion ORAM: A Constant Bandwidth Blowup Oblivious Ram. In Proceedings of the 13th International Conference on Theory of Cryptography—TCC 2016-A, Tel Aviv, Israel, 10–13 January 2016; Volume 9563, pp. 145–174.
24. Garg, S.; Mohassel, P.; Papamonthou, C. TWORAM: Efficient Oblivious RAM in Two Rounds with Applications to Searchable Encryption. In Proceedings of the 36th Annual International Cryptology Conference on Advances in Cryptology—CRYPTO, Santa Barbara, CA, USA, 14–18 August 2016; Volume 9816, pp. 563–592.
25. Shen, E.; Shi, E.; Waters, B. Predicate privacy in encryption systems. In Proceedings of the Theory of Cryptography, San Francisco, CA, USA, 15–17 March 2009; Volume 5444, pp. 457–473.
26. Kamara, S.; Moataz, T. Boolean Searchable Symmetric Encryption with Worst-Case Sub-Linear Complexity. In *IACR Cryptology ePrint Archive*; Report 2017/126; Springer: Cham, Switzerland, 2017.
27. Khader, D. Public Key Encryption with Keyword Search Based on K-Resilient IBE. In *IACR Cryptology ePrint Archive*; Report 2006/358; Springer: Berlin, Germany, 2006.
28. Baek, J.; Safavi-Naini, R.; Susilo, W. Public Key Encryption with Keyword Search Revisited. In Proceedings of the Computational Science and Its Applications—ICCSA 2008, Perugia, Italy, 30 June–3 July 2008; Volume 5072, pp. 1249–1259.
29. Crescenzo, G.D.; Saraswat, V. Public Key Encryption with Searchable Keywords Based on Jacobi Symbols. In Proceedings of the 8th International Conference on Cryptology, Progress in Cryptology—INDOCRYPT 2007, Chennai, India, 9–13 Decembe 2007; Volume 4859, pp. 282–296.
30. Chen, L.; Tang, Q. Public-Key Encryption with Registered Keyword Search. In Proceedings of the 6th European Workshop, EuroPKI 2009—Public Key Infrastructures, Services and Applications, Pisa, Italy, 10–11 September 2009; Volume 6391, pp. 163–178.
31. Chen, R.; Guo, F.; Wang, X.; Yang, G. A New General Framework for Secure Public Key Encryption with Keyword Search. In Proceedings of the 20th Australasian Conference, ACISP 2015—Information Security and Privacy, Brisbane, QLD, Australia, 29 June–1 July 2015; Volume 9144, pp. 59–76.
32. Boneh, D.; Waters, B. Conjunctive, Subset, and Range Queries on Encrypted Data. In Proceedings of the Theory of Cryptography, 4th Theory of Cryptography Conference, Amsterdam, The Netherlands, 21–24 February 2007; Volume 4392, pp. 535–554.
33. Katz, J.; Sahai, A.; Waters, B. Predicate encryption supporting disjunctions, polynomial equations, and inner products. *J. Cryptol.* **2008**, *4965*, 146–162.
34. Yang, Y. Towards Multi-user Private Keyword Search for Cloud Computing. In Proceedings of the IEEE International Conference on Cloud Computing (CLOUD), Washington, DC, USA, 4–9 July 2011; pp. 758–759.
35. Zhang, J.; Zhang, W.; Qiao, D. Iowa State University Digital Repository. In *A Multi-User Oblivious RAM for Outsourced Data*; Computer Science Technical Reports, 262; Iowa State University: Iowa, IA, USA, 2014.
36. Pieprzyk, J.; Wang, H.; Wang, P. Keyword Field-Free Conjunctive Keyword Searches on Encrypted Data and Extension for Dynamic Groups. In Proceedings of the 7th International Conference on Cryptology and Network Security, Hong-Kong, China, 2–4 December 2008; Volume 5339, pp. 178–195.
37. CipherLocker™. 2017. Available online: https://cipherlocker.com/ (accessed on 14 November 2017).
38. Popa, R.A.; Redfield, C.; Zeldovich, N. CryptDB: Protecting Confidentiality with Encrypted Query Processing. In Proceedings of the 23rd ACM Symposium on Operating Systems Principles (SOSP), Cascais, Portugal, 23–26 October 2011; ACM: New York, NY, USA, 2011; pp. 85–100.

39. Boldyreva, A.; Chenette, N.; O'Neill, A. Order-Preserving Encryption Revisited: Improved Security Analysis and Alternative Solutions. In *IACR Cryptology ePrint Archive*; Report 2012/625; International Association of Cryptologic Research, 2012. Available online: https://eprint.iacr.org/2012/625 (accessed on 4 November 2012).
40. Paillier, P. Public-key cryptosystems based on composite degree residuosity classes. In Proceedings of the International Conference on the Theory and Application of Cryptographic, Techniques Advances in Cryptology—EUROCRYPT '99, Prague, Czech Republic, 2–6 May 1999; Volume 1592, pp. 223–238.
41. Popa, R.A.; Redfield, C.; Tu, S.; Balakrishman, H.; Kaashoek, F.; Madden, S.; Zeldovich, N.; Burrows, A. CryptDB. 2011. Available online: https://css.csail.mit.edu/cryptdb (accessed 19 October 2017).
42. Naveed, M.; Kamara, S.; Wright, C.V. Inference Attacks on Property-Preserving Encrypted Databases. In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, 12–16 October 2015; ACM: New York, NY, USA, 2015; pp. 644–655.
43. House of Commons Investigatory Powers Bill. 2016. Available online: http://www.publications.parliament.uk/pa/bills/cbill/2015-2016/0172/160172.pdf (accessed 6 May 2016).
44. Goldreich, O.; Ostrovsky, R. Software protection and simulation on oblivious rams. *J. Assoc. Comput. Mach.* **1996**, *43*, 431–473.