A Survey of Provably Secure Searchable Encryption

CHRISTOPH BÖSCH, PIETER HARTEL, WILLEM JONKER, and ANDREAS PETER,
University of Twente, The Netherlands

We survey the notion of provably secure searchable encryption (SE) by giving a complete and comprehensive overview of the two main SE techniques: searchable symmetric encryption (SSE) and public key encryption with keyword search (PEKS). Since the pioneering work of Song, Wagner, and Perrig (IEEE S&P '00), the field of provably secure SE has expanded to the point where we felt that taking stock would provide benefit to the community.

The survey has been written primarily for the nonspecialist who has a basic information security background. Thus, we sacrifice full details and proofs of individual constructions in favor of an overview of the underlying key techniques. We categorize and compare the different SE schemes in terms of their security, efficiency, and functionality. For the experienced researcher, we point out connections between the many approaches to SE and identify open research problems.

Two major conclusions can be drawn from our work. While the so-called IND-CKA2 security notion becomes prevalent in the literature and efficient (sublinear) SE schemes meeting this notion exist in the symmetric setting, achieving this strong form of security efficiently in the asymmetric setting remains an open problem. We observe that in multirecipient SE schemes, regardless of their efficiency drawbacks, there is a noticeable lack of query expressiveness that hinders deployment in practice.

Categories and Subject Descriptors: A.1 [**General Literature**]: Introductory and Survey; E.3 [**Data**]: Data Encryption; H.3.0 [**Information Storage and Retrieval**]: General

General Terms: Security

Additional Key Words and Phrases: Secure data management, privacy, keyword search on encrypted data, searchable encryption, provably secure

ACM Reference Format:
Christoph Bösch, Pieter Hartel, Willem Jonker, and Andreas Peter. 2014. A survey of provably secure searchable encryption. ACM Comput. Surv. 47, 2, Article 18 (August 2014), 51 pages.
DOI: http://dx.doi.org/10.1145/2636328

1. MOTIVATION AND INTRODUCTION

We start with our motivation for writing this survey and introduce the main concepts and challenges of provably secure searchable encryption.

1.1. Motivation

The wide proliferation of sensitive data in open information and communication infrastructures all around us has fueled research on secure data management and boosted its relevance. For example, legislation around the world stipulates that electronic health records (EHRs) should be encrypted, which immediately raises the question of how to

Andreas Peter is supported by the THeCS project as part of the Dutch national program COMMIT.
Authors' addresses: C. Bösch, P. Hartel, W. Jonker, and A. Peter, Centre for Telematics and Information Technology, University of Twente, P.O.-Box 217, 7500 AE Enschede, The Netherlands.

DOI: http://dx.doi.org/10.1145/2636328

search EHRs efficiently and securely. After a decade of research in the field of provably secure searchable encryption, we felt that the time has come to survey the field by putting the many individual contributions into a comprehensive framework. On the one hand, the framework allows practitioners to select appropriate techniques to address the security requirements of their applications. On the other hand, the framework points out uncharted areas of research since by no means are all application requirements covered by the techniques currently in existence. We hope that researchers will find the inspiration in this survey that is necessary to develop the field further.

1.2. Introduction to Searchable Encryption

Remote and cloud storage is ubiquitous and widely used for services such as backups or outsourcing data to reduce operational costs. However, these remote servers cannot be trusted, because administrators, or hackers with root rights, have full access to the server and consequently to the plaintext data. Or imagine that your trusted storage provider sells its business to a company that you do not trust, and which will have full access to your data. Thus, to store sensitive data in a secure way on an untrusted server, the data has to be encrypted. This reduces security and privacy risks by hiding all information about the plaintext data. Encryption makes it impossible for both insiders and outsiders to access the data without the keys but at the same time removes all search capabilities from the data owner. One trivial solution to re-enable searching functionality is to download the whole database, decrypt it locally, and then search for the desired results in the plaintext data. For most applications, this approach would be impractical. Another method lets the server decrypt the data, runs the query on the server side, and sends only the results back to the user. This allows the server to learn the plaintext data being queried and hence makes encryption less useful. Instead, it is desirable to support the fullest possible search functionality on the server side, without decrypting the data, and thus, with the smallest possible loss of data confidentiality. This is called *searchable encryption* (SE).

General Model. An SE scheme allows a server to search in encrypted data on behalf of a client without learning information about the plaintext data. Some schemes implement this via a ciphertext that allows searching (e.g., Song et al. [2000], as discussed in Section 3.1.1.1), while most other schemes let the client generate a searchable encrypted index. To create a searchable encrypted index I of a database $\mathcal{DB} = (M_1, \ldots, M_n)$ consisting of n messages[1] M_i, some data items $\mathcal{W} = (w_1, \ldots, w_m)$, for example, keywords w_j (which can later be used for queries), are extracted from the document(s) and encrypted (possibly nondecryptable, e.g., via a hash function) under a key K of the client using an algorithm called $\mathsf{BuildIndex}$. M_i may also refer to database records in a relational database (e.g., MySQL). In addition, the document may need to be encrypted with a key K' (often, $K' \neq K$) using an algorithm called Enc. The encrypted index and the encrypted documents can then be stored on a semitrusted (honest-but-curious [Goldreich 2004]) server that can be trusted to adhere to the storage and query protocols, but which tries to learn as much information as possible. As a result, the server stores a database of the client in the following form:

$$I = \mathsf{BuildIndex}_K(\mathcal{DB} = (M_1, \ldots, M_n), \mathcal{W} = (w_1, \ldots, w_m));\ C = \mathsf{Enc}_{K'}(M_1, \ldots, M_n).$$

To search, the client generates a so-called trapdoor $T = \mathsf{Trapdoor}_K(f)$, where f is a predicate on w_j. With T, the server can search the index using an algorithm called Search and see whether the encrypted keywords satisfy the predicate f and return the corresponding (encrypted) documents (see Figure 1). For example, f could determine whether a specific keyword w is contained in the index [Goh 2003], and a

[1]By messages we mean plaintext data like files, documents, or records in a relational database.

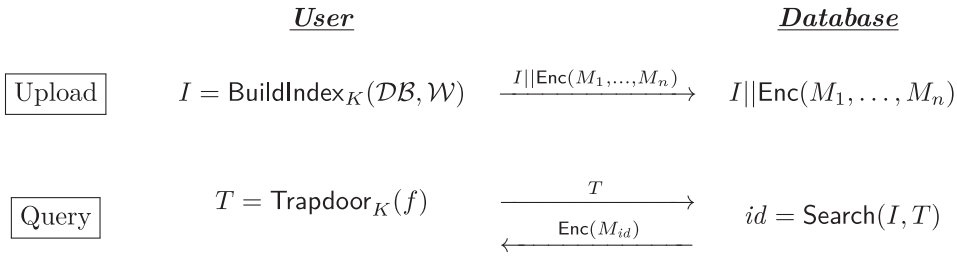


Fig. 1. General model of an index-based searchable encryption scheme.

more sophisticated f could determine whether the inner product of keywords in the index and a target keyword set is 0 [Shen et al. 2009].

Figure 1 gives a general model of an index-based scheme. Small deviations are possible (e.g., some schemes do not require the entire keyword list $\mathcal{W}$ for building the index).

Single user versus multiuser. SE schemes are built on the client/server model, where the server stores encrypted data on behalf of one or more clients (i.e., the writers). To request content from the server, one or more clients (i.e., readers) are able to generate trapdoors for the server, which then searches on behalf of the client. This results in the following four SE architectures:

—single writer/single reader (S/S)
—multiwriter/single reader (M/S)
—single writer/multireader (S/M)
—multiwriter/multireader (M/M)

Depending on the architecture, the SE scheme is suitable for either data outsourcing (S/S) or data sharing (M/S, S/M, M/M).

Symmetric versus asymmetric primitives. Symmetric key primitives allow a single user to read and write data (S/S). The first S/S scheme, proposed by Song et al. [2000], uses symmetric key cryptography and allows only the secret key holder to create searchable ciphertexts and trapdoors. In a public key encryption (PKE) scheme, the private key decrypts all messages encrypted under the corresponding public key. Thus, PKE allows multiuser writing, but only the private key holder can perform searches. This requires an M/S architecture. The first M/S scheme is due to Boneh et al. [2004b], who proposed a public key encryption with keyword search (PEKS) scheme. Meanwhile, PEKS is also used as a name for the class of M/S schemes.

The need for key distribution. Some SE schemes extend the $*$/S setting to allow multiuser reading ($*$/M). This extension introduces the need for distributing the secret key to allow multiple users to search in the encrypted data. Some SE schemes use key sharing; other schemes use key distribution, proxy re-encryption, or other techniques to solve the problem.

User revocation. An important requirement that comes with the multireader schemes is user revocation. Curtmola et al. [2006] extend their single-user scheme with broadcast encryption [Fiat and Naor 1994] (BE) to a multiuser scheme (S/M). Since only one key is shared among all users, each revocation requires a new key to be distributed to the remaining users, which causes a high revocation overhead. In other schemes, each user might have its own key, which makes user revocation easier and more efficient.

Research challenges/tradeoffs. There are three main research directions in SE: improve (1) the efficiency, (2) the security, and (3) the query expressiveness. Efficiency is measured by the computational and communication complexity of the scheme. To

define the security of a scheme formally, a variety of different security models have been proposed. Since security is never free, there is always a tradeoff between security on the one hand and efficiency and query expressiveness on the other. Searchable encryption schemes that use a security model with a more powerful adversary are likely to have a higher complexity.

The query expressiveness of the scheme defines what kind of search queries are supported. In current approaches, it is often the case that more expressive queries result in either less efficiency or less security. Thus, the tradeoffs of SE schemes are threefold: (1) security versus efficiency, (2) security versus query expressiveness, and (3) efficiency versus query expressiveness.

1.3. Scope of the Article

The main techniques for provably secure searchable encryption are searchable symmetric encryption (SSE) and public key encryption with keyword search (PEKS). However, techniques such as predicate encryption (PE), inner product encryption (IPE), anonymous identity-based encryption (AIBE), and hidden-vector encryption (HVE) have been brought into relation with searchable encryption [Boyen and Waters 2006; Gentry 2006; Kiltz 2007; Nishide et al. 2008]. Since the main focus of these techniques is (fine-grained) access control (AC) rather than searchable encryption, those AC techniques are mentioned in the related work section but are otherwise not our focus.

1.4. Contributions

We give a complete and comprehensive overview of the field of SE, which provides an easy entry point for nonspecialists and allows researchers to keep up with the many approaches. The survey gives beginners a solid foundation for further research. For researchers, we identify various gaps in the field and indicate open research problems. We also point out connections between the many schemes. With our extensive tables and details about efficiency and security, we allow practitioners to find (narrow down the number of) suitable schemes for the many different application scenarios of SE.

1.5. Reading Guidelines

We discuss all papers based on the following four aspects. The main features are emphasized in **bold** for easy readability:

General information: The general **idea** of the scheme will be stated.

Efficiency: The efficiency aspect focuses on the computational complexity of the **encryption/index generation** (upload phase) and the **search/test** (query phase) algorithms. For a fair comparison of the schemes, we report the number of operations required in the algorithms. Where applicable, we give information on the update complexity or interactiveness (number of rounds).

Security: To ease the comparison of SE schemes with respect to their security, we briefly outline the major fundamental security definitions in Section 2.3 such that the to-be-discussed SE schemes can be considered as being secure in a potential modification of one of these basic definitions. We provide short and intuitive explanations of these modifications and talk about the underlying security assumptions.[2]

See also: We refer to related work within the survey and beyond. For a reference within the survey, we state the original paper reference and the section number in which

[2]A detailed security analysis of each individual scheme lies outside the scope of this work. We stress that some of the mentioned modifications may have unforeseen security implications that we do not touch upon. The interested reader is recommended to look up the original references for more details.

the scheme is discussed. For references beyond the survey, we give only the paper reference. Otherwise, we omit this aspect.

This reading guideline will act as our framework to compare the different works. Several pioneering schemes [Song et al. 2000; Curtmola et al. 2006; Boneh et al. 2004b] will be discussed in more detail to get a better feeling on how searchable encryption works. Each architecture section ends with a synthesis and an overview table that summarizes the discussed schemes. The tables (I, III, V, VII) are, like the sections, arranged by the query expressiveness. The first column gives the paper and section reference. The complexity or efficiency part of the table is split into the encrypt, trapdoor, and search algorithms of the schemes and quantifies the most expensive operations that need to be computed. The security part of the table gives information on the security definitions, assumptions, and if the random oracle model (ROM) is used to prove the scheme secure. The last column highlights some of the outstanding features of the schemes.

1.6. Organization of the Article

The rest of the article is organized as follows. Section 2 gives background information on indexes and the security definitions used in this survey. The discussion of the schemes can be found in Section 3 (S/$*$) and Section 4 (M/$*$). We divided these sections into the four architectures: Section 3.1 (S/S), Section 3.2 (S/M), Section 4.1 (M/S), and Section 4.2 (M/M). In these sections, the papers are arranged according to their expressiveness. We start with single equality tests, then conjunctive equality tests, followed by extended search queries, like subset, fuzzy or range queries, or queries based on inner products. Inside these subsections, the schemes are ordered chronologically. Section 5 discusses the related work, in particular seminal schemes on access control. Section 6 concludes and discusses future work.

2. PRELIMINARIES

This section gives background information on indexes, privacy issues, and security definitions used in this survey.

2.1. Efficiency in SE Schemes

As mentioned earlier, searchable encryption schemes usually come in two classes. Some schemes directly encrypt the plaintext data in a special way, so that the ciphertext can be queried (e.g., for keywords). This results in a search time linear in the length of the data stored on the server. In our example, using n documents with w keywords yields a complexity linear in the number of keywords per document $O(nw)$, since each keyword has to be checked for a match.

To speed up the search process, a common tool used in databases is an index, which is generated over the plaintext data. Introducing an index can significantly decrease the search complexity and thus increases the search performance of a scheme. The increased search performance comes at the cost of a preprecessing step. Since the index is built over the plaintext data, generating an index is not always possible and highly depends on the data to be encrypted. The two main approaches for building an index are as follows:

—A forward index is an index per document (cf. Figure 2(a)) and naturally reduces the search time to the number of documents, that is, $O(n)$. This is because one index per document has to be processed during a query.
—Currently, the prevalent method for achieving sublinear search time is to use an inverted index, which is an index per keyword in the database (cf. Figure 2(b)). Depending on how much information we are willing to leak, the search complexity

document id	keywords
1	w_2, w_5, w_7
2	w_1, w_2, w_4, w_6, w_8
...	...
n	w_2, w_5, w_6

(a) Forward index.

keyword	document ids
w_1	$2, 3, 9$
w_2	$1, 2, 6, 7, n$
...	...
w_m	$1, 3, 8$

(b) Inverted index.

Fig. 2. Example of an unencrypted forward and inverted index.

can be reduced to $O(\log w')$ (e.g., using a hash tree) or $O(|D(w)|)$ in the optimal case, where $|D(w)|$ is the number of documents containing the keyword w.

Note that the client does not have to build an index on the plaintexts. This is the case, for example, for the scheme by Song et al. [2000] (SWP) or when deterministic encryption is used. In the latter case, it is sometimes reasonable to index the ciphertexts to speed up the search.

All schemes discussed in this survey, except for SWP, make use of a searchable index. Only the SWP scheme encrypts the message in such a way that the resulting ciphertext is directly searchable and decryptable.

2.2. Privacy Issues in SE Schemes

An SE scheme will leak information, which can be divided into three groups: index information, search pattern, and access pattern.

—Index information refers to the information about the keywords contained in the index. Index information is leaked from the stored ciphertext/index. This information may include the number of keywords per document/database, the number of documents, the documents length, document IDs, and/or document similarity.
—Search pattern refers to the information that can be derived in the following sense: given that two searches return the same results, determine whether the two searches use the same keyword/predicate. Using deterministic trapdoors directly leaks the search pattern. Accessing the search pattern allows the server to use statistical analysis and (possibly) determine (information about) the query keywords.
—Access pattern refers to the information that is implied by the query results. For example, one query can return a document x, while the other query could return x and another 10 documents. This implies that the predicate used in the first query is more restrictive than that in the second query.

Most papers follow the security definition deployed in the traditional searchable encryption [Curtmola et al. 2006]. Namely, it is required that nothing should be leaked from the remotely stored files and index beyond the outcome and the pattern of search queries. SE schemes should not leak the plaintext keywords in either the trapdoor or the index. To capture the concept that neither index information nor the search pattern is leaked, Shen et al. [2009] (SSW) formulate the definition of *full security*. All discussed papers (except for SSW and BTH$^+$ [Bösch et al. 2012]) leak at least the search pattern and the access pattern. The two exceptions protect the search pattern and are fully secure.

2.3. A Short History of Security Definitions for (S)SE

When Song et al. [2000] proposed the first SE scheme, there were no formal security definitions for the specific needs of SE. However, the authors proved their scheme to be a secure pseudo-random generator. Their construction is even *indistinguishable against chosen plaintext attacks* **(IND-CPA)** secure [Kamara et al. 2012]. Informally,

an encryption scheme is IND-CPA secure if an adversary $\mathcal{A}$ cannot distinguish the encryptions of two arbitrary messages (chosen by $\mathcal{A}$), even if $\mathcal{A}$ can adaptively query an encryption oracle. Intuitively, this means that a scheme is IND-CPA secure if the resulting ciphertexts do not even leak partial information about the plaintexts. This IND-CPA definition makes sure that ciphertexts do not leak information. However, in SE, the main information leakage comes from the trapdoor/query, which is not taken into account in IND-CPA security. Thus, IND-CPA security is not considered to be the right notion of security for SE.

The first notion of security in the context of SE was introduced by Goh [2003] (Section 3.1.1.2), who defines security for indexes as *semantic security (indistinguishability) against adaptive chosen keyword attacks* **(IND1-CKA)**. IND1-CKA makes sure that $\mathcal{A}$ cannot deduce the document's content from its index. An IND1-CKA secure scheme generates indexes that appear to contain the same number of words for equal size documents (in contrast to unequal size documents). This means that given two encrypted documents of *equal size* and an index, $\mathcal{A}$ cannot decide which document is encoded in the index. IND1-CKA was proposed for "secure indexes," a secure data structure with many uses next to SSE. Goh remarks that IND1-CKA does not require the trapdoors to be secure, since it is not required by all applications of secure indexes.

Chang and Mitzenmacher [2005] introduced a new simulation-based IND-CKA definition, which is a stronger version of IND1-CKA in the sense that an adversary cannot even distinguish indexes from two *unequal size* documents. This requires that unequal size documents have indexes that appear to contain the same number of words. In addition, Chang and Mitzenmacher tried to protect the trapdoors with their security definition. Unfortunately, their formalization of the security notion was incorrect, as pointed out by Curtmola et al. [2006], and can be satisfied by an insecure SSE scheme.

Later, Goh introduced the **IND2-CKA** security definition, which protects the document size like Chang and Mitzenmacher's definition but still does not provide security for the trapdoors. Both IND1/2-CKA security definitions are considered weak in the context of SE because they do not guarantee the security of the trapdoors; that is, they do not guarantee that the server cannot recover (information about) the words being queried from the trapdoor.

Curtmola et al. [2006] revisited the existing security definitions and pointed out that previous definitions are not adequate for SSE and that the security of indexes and the security of trapdoors are inherently linked. They introduce two new adversarial models for searchable encryption, a *nonadaptive* **(IND-CKA1)** and an *adaptive* **(IND-CKA2)** one, which are widely used as the standard definitions for SSE to date. Intuitively, the definitions require that nothing should be leaked from the remotely stored files and index beyond the outcome and the search pattern of the queries. The IND-CKA1/2 security definitions include security for trapdoors and guarantee that the trapdoors do not leak information about the keywords (except for what can be inferred from the search and access patterns). Nonadaptive definitions only guarantee the security of a scheme if the client generates all queries at once. This might not be feasible for certain (practical) scenarios [Curtmola et al. 2006]. The adaptive definition allows $\mathcal{A}$ to choose its queries as a function of previously obtained trapdoors and search outcomes. Thus, IND-CKA2 is considered a strong security definition for SSE.

In the asymmetric (public key) setting (see Boneh et al. [2004b]), schemes do not guarantee security for the trapdoors, since usually the trapdoors are generated using the public key. The definition in this setting guarantees that no information is learned about a keyword unless the trapdoor for that word is available. An adversary should not be able to distinguish between the encryptions of two challenge keywords of its choice, even if it is allowed to obtain trapdoors for *any* keyword (except the challenge keywords). Following the previous notion, we use **PK-CKA2** to denote

indistinguishability against adaptive chosen keyword attacks of public key schemes in the remainder of this survey.

Several schemes adapt these security definitions to their setting. We will explain these special purpose definitions in the individual sections and mark them in the overview tables.

Other security definitions were introduced and/or adapted for SE as follows:

—*Universal composability* **(UC)** is a general-purpose model that says that protocols remain secure even if they are arbitrarily composed with other instances of the same or other protocols. The KO scheme [Kurosawa and Ohtaki 2012] (Section 3.1.1.8) provides IND-CKA2 security in the UC model (denoted as **UC-CKA2** in the reminder), which is stronger than the standard IND-CKA2.
—*Selectively secure* **(SEL-CKA)** [Canetti et al. 2003] is similar to PK-CKA2, but the adversary $\mathcal{A}$ has to commit to the search keywords at the beginning of the security game instead of after the first query phase.
—*Fully secure* **(FS)** is a security definition in the context of SSE introduced by Shen et al. [2009] that allows nothing to be leaked, except for the access pattern.

Deterministic encryption. Deterministic encryption involves no randomness and thus always produces the same ciphertext for a given plaintext and key. In the public key setting, this implies that a deterministic encryption can never be IND-CPA secure, as an attacker can run brute force attacks by trying to construct all possible plaintext–ciphertext pairs using the encryption function. Deterministic encryption allows more efficient schemes, whose security is weaker than using probabilistic encryption. Deterministic SE schemes try to address the problem of searching in encrypted data from a practical perspective where the primary goal is efficiency. An example of an immediate security weakness of this approach is that deterministic encryption inherently leaks message equality. Bellare et al.'s [2007b] (Section 4.2.1.1) security definition for deterministic encryption in the public key setting is similar to the standard IND-CPA security definition with the following two exceptions. A scheme that is secure in Bellare et al.'s definition requires plaintexts with large min-entropy and plaintexts that are independent from the public key. This is necessary to circumvent the previously stated brute force attack; here large min-entropy ensures that the attacker will have a hard time brute-forcing the correct plaintext–ciphertext pair. The less min-entropy the plaintext has, the less security the scheme achieves. Amanatidis et al. [2007] (Section 3.1.1.5) and Raykova et al. [2009] (Section 3.2.1.3) provide a similar definition for deterministic security in the symmetric setting. Also for their schemes, plaintexts are required to have large min-entropy. Deterministic encryption is not good enough for most practical purposes, since the plaintext data usually has low min-entropy and thus leaks too much information, including document/keyword similarity.

Random oracle model versus standard model. SE schemes might be proven secure (according to the previous definitions) in the random oracle model [Bellare and Rogaway 1993] (ROM) or the standard model (STM). Other models (e.g., generic group model) exist but are not relevant for the rest of the survey. The STM is a computational model in which an adversary is limited only by the amount of resources available (i.e., time and computational power). This means that only complexity assumptions are used to prove a scheme secure. The ROM replaces cryptographic primitives by idealized versions (e.g., replacing a cryptographic hash function with a genuinely random function). Solutions in the ROM are often more efficient than solutions in the STM but have the additional assumption of idealized cryptographic primitives.

— Encrypt$(k', k'', M = \{w_i\})$:
(1) Encrypt w_i with a deterministic encryption algorithm and split $X_i = E_{k''}(w_i)$ into two parts $X_i = \langle L_i, R_i \rangle$.
(2) Generate the pseudo-random value S_i.
(3) Calculate the key $k_i = f_{k'}(L_i)$.
(4) Compute $F_{k_i}(S_i)$, where $F(\cdot)$ is a pseudo-random function, and set $Y_i = \langle S_i, F_{k_i}(S_i) \rangle$.
(5) Output the searchable ciphertext as $C_i = X_i \oplus Y_i$.
— Trapdoor(k', k'', w):
(1) Encrypt w as $X = E_{k''}(w)$, where X is split into two parts $X = \langle L, R \rangle$.
(2) Compute $k = f_{k'}(L)$.
(3) Output $T_w = \langle X, k \rangle$
— Search$(T_w = \langle X, k \rangle)$:
(1) Check whether $C_i \oplus X$ is of the form $\langle s, F_k(s) \rangle$ for some s.

Fig. 3. Algorithmic description of the Song, Wagner, and Perrig scheme.

3. SINGLE-WRITER SCHEMES (S/∗)

This section deals with the S/S and S/M schemes.

3.1. Single Writer/Single Reader (S/S)

In a single writer/single reader (S/S) scheme, the secret key owner is allowed to create searchable content and to generate trapdoors to search. The secret key should normally be known only by one user, who is the writer and the reader using a symmetric encryption scheme. However, other scenarios (e.g., using a PKE and keeping the public key secret) are also possible but result in less efficient schemes.

3.1.1. Single Equality Test. With an equality test, we mean an exact keyword match for a single search keyword.

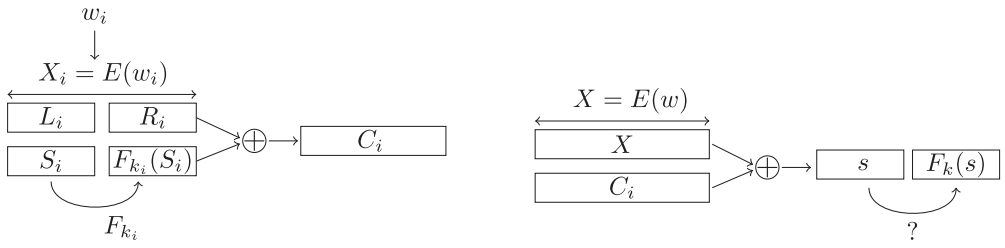
3.1.1.1. Sequential scan. Song et al. [2000] (SWP) propose the first practical scheme for searching in encrypted data by using a special two-layered encryption construct that allows searching the ciphertexts with a sequential scan. The **idea** is to encrypt each word separately and then embed a hash value (with a special format) inside the ciphertext. To search, the server can extract this hash value and check if the value is of this special form (which indicates a match).

The disadvantages of SWP are that it has to use fix-sized words, that it is not compatible with existing file encryption standards, and that it has to use their specific two-layer encryption method, which can be used only for plaintext data and not, for example, on compressed data.

Details: To create searchable ciphertext (cf. Figure 4(a)), the message is split into fixed-size words w_i and encrypted with a deterministic encryption algorithm $E(\cdot)$. Using a deterministic encryption is necessary to generate the correct trapdoor. The encrypted word $X_i = E(w_i)$ is then split into two parts $X_i = \langle L_i, R_i \rangle$. A pseudo-random value S_i is generated (e.g., with the help of a stream cipher). A key $k_i = f_{k'}(L_i)$ is calculated (using a pseudo-random function $f(\cdot)$) and used for the keyed hash function $F(\cdot)$ to hash the value S_i. This results in the value $Y_i = \langle S_i, F_{k_i}(S_i) \rangle$, which is used to encrypt X_i as $C_i = X_i \oplus Y_i$, where $\oplus$ denotes the XOR.

To search, a trapdoor is required. This trapdoor contains the encrypted keyword to search for $X = E(w) = \langle L, R \rangle$ and the corresponding key $k = f_{k'}(L)$. With this trapdoor, the server is now able to search (cf. Figure 4(b)), by checking for all stored ciphertexts C_i, if $C_i \oplus X$ is of the form $\langle s, F_k(s) \rangle$ for some s. If so, the keyword was found. The detailed algorithm is shown in Figure 3.

Efficiency: The complexity of the encryption and search algorithms is linear in the total number of words per document (i.e., worst case). To **encrypt**, one encryption, one XOR, and two pseudo-random functions have to be computed per word per



(a) SWP: Encryption (b) SWP: Sequential search

Fig. 4. Song et al. (SWP) [2000] scheme.

document. The trapdoor requires one encryption and a pseudo-random function. The **search** requires one XOR and one pseudo-random function per word per document.

Security: SWP is the first searchable encryption scheme and uses **no formal security definition for SE**. However, SWP is **IND-CPA** secure under the assumption that the underlying **primitives are proven secure/exist** (e.g., pseudo-random functions). IND-CPA security does not take queries into account and is thus of less interest in the context of SE. SWP **leaks** the potential positions (i.e., positions where a possible match occurs, taking into account a false-positive rate, e.g., due to collisions) of the queried keywords in a document. After several queries, it is possible to learn the words inside the documents with statistical analysis.

See also: Brinkman et al. [2004] show that the scheme can be applied to XML data. SWP is used in CryptDB [Popa et al. 2011].

3.1.1.2. Secure indexes per document. Goh [2003] addresses some of the limitations (e.g., use of fixed-size words, special document encryption) of the SWP scheme by adding an index for each document, which is independent of the underlying encryption algorithm. The **idea** is to use a Bloom filter (BF) [Bloom 1970] as a per-document index.

A BF is a data structure that is used to answer set membership queries. It is represented as an array of b bits that are initially set to 0. In general, the filter uses r independent hash functions h_t, where $h_t : \{0,1\}^* \rightarrow [1,b]$ for $t \in [1,r]$, each of which maps a set element to one of the b array positions. For each element e (e.g., keywords) in the set $\mathcal{S} = \{e_1, \ldots e_m\}$, the bits at positions $h_1(e_i), \ldots, h_r(e_i)$ are set to 1. To check whether an element x belongs to the set $\mathcal{S}$, check if the bits at positions $h_1(x), \ldots, h_r(x)$ are set to 1. If so, x is considered a member of set $\mathcal{S}$.

By using one BF per document, the search time becomes linear in the number of documents. An inherent problem of using Bloom filters is the possibility of **false positives**. With appropriate parameter settings, the false-positive probability can be reduced to an acceptable level. Goh uses BF, where each distinct word in a document is processed by a pseudo-random function twice and then inserted into the BF. The second run of the pseudo-random function takes as input the output of the first run and, in addition, a unique document identifier, which makes sure that all BFs look different, even for documents with the same keyword set. This avoids leaking document similarity upfront.

Efficiency: The **index generation** has to generate one BF per document. Thus, the algorithm is linear in the number of distinct words per document. The BF lookup is a constant time operation and has to be done per document. Thus, the time for a **search** is proportional to the number of documents, in contrast to the number of words in the SWP scheme. The size of the document index is proportional to

the number of distinct words in the document. Since a Bloom filter is used, the asymptotic constants are small (i.e., several bits).

Security: The scheme is proven **IND1-CKA** secure. In a later version of the paper, Goh proposed a modified version of the scheme that is IND2-CKA secure. Both security definitions do not guarantee the security of the trapdoors; that is, they do not guarantee that the server cannot recover (information about) the words being queried from the trapdoor.

A disadvantage of BF is that the number of 1s is dependent on the number of BF entries, in this case the number of distinct keywords per document. As a consequence, the scheme leaks the number of keywords in each document. To avoid this leakage, padding of arbitrary words can be used to make sure that the number of 1s in the BF is nearly the same for different documents. The price to pay is a higher false-positive rate or a larger BF compared to the scheme without padding.

3.1.1.3. Index per document with prebuilt dictionaries. Chang and Mitzenmacher [2005] develop two index schemes (CM-I, CM-II), similar to Goh [2003]. The **idea** is to use a prebuilt dictionary of search keywords to build an index per document. The index is an m-bit array, initially set to 0, where each bit position corresponds to a keyword in the dictionary. If the document contains a keyword, its index bit is set to 1. CM-$*$ assume that the user is **mobile** with limited storage space and bandwidth, so the schemes require only a small amount of communication overhead. Both constructions use only pseudo-random permutations and pseudo-random functions. CM-I stores the dictionary at the client and CM-II encrypted at the server. Both constructions can handle secure updates to the document collection in the sense that CM-$*$ ensure the security of the consequent submissions in the presence of previous queries.

Efficiency: The CM-$*$ schemes associate a masked keyword index to each document. The **index generation** is linear in the number of distinct words per document. The time for a **search** is proportional to the total number of documents. CM-II uses a **two-round** retrieval protocol, whereas CM-I only requires one round for searching.

Security: CM introduced a new simulation-based IND-CKA definition, which is a stronger version of IND1-CKA. This new security definition has been broken by Curtmola et al. [2006]. CM-$*$ still are at least **IND2-CKA** secure.

In contrast to other schemes, which assume only an honest-but-curious server, the authors discuss some security improvements that can deal with a malicious server that sends either incorrect files or incomplete search results back to the user.

3.1.1.4. Index per keyword and improved definitions. Curtmola et al. [2006] (CGK$^+$) propose two new constructions (CGK$^+$-I, CGK$^+$-II), where the **idea** is to add an inverted index, which is an index per distinct word in the database instead of per document (cf. Figure 2(b)). This reduces the search time to the number of documents that contain the keyword. This is not only sublinear but also optimal.

Details (CGK$^+$-I): The index consists of (1) an array A made of a linked list L per distinct keyword and (2) a look-up table T to identify the first node in A. To build the array A, we start with a linked list $\mathsf{L_i}$ per distinct keyword w_i (cf. Figure 6(a)). Each node $\mathsf{N}_{i,j}$ of $\mathsf{L_i}$ consists of three fields $\langle a||b||c\rangle$, where a is the document identifier of the document containing the keyword, b is the key $\kappa_{i,j}$ that is used to encrypt the next node, and c is a pointer to the next node or $\emptyset$. The nodes in array A are scrambled in a random order and then encrypted. The node $\mathsf{N}_{i,j}$ is encrypted with the key $\kappa_{i,j-1,}$ which is stored in the previous node $\mathsf{N}_{i,j-1}$. The table T is a look-up table that stores per keyword w_i a

E is a semantic secure symmetric encryption scheme, f is a pseudo-random function and π, ψ are two pseudo-random permutations. $\mathcal{D}(w)$ denotes the set of ids of documents that contain keyword w.

— $\mathsf{Keygen}(1^k, 1^l)$: Generate random keys $s, y, z \xleftarrow{R} \{0,1\}^k$ and output $K = (s, y, z, 1^l)$.
— $\mathsf{BuildIndex}(K, \mathcal{D} = \{D_j\})$:
(1) Initialization:
(a) scan $\mathcal{D}$ and build Δ', the set of distinct words in $\mathcal{D}$. For each word $w \in \Delta'$, build $\mathcal{D}(w)$;
(b) initialize a global counter $\mathsf{ctr} = 1$.
(2) Build array A:
(a) for each $w_i \in \Delta'$: (build a linked list L_i with nodes $\mathsf{N}_{i,j}$ and store it in array A)
i. generate $\kappa_{i,0} \xleftarrow{R} \{0,1\}^l$
ii. for $1 \leq j \leq |\mathcal{D}(w_i)|$:
— generate $\kappa_{i,j} \xleftarrow{R} \{0,1\}^l$ and set node $\mathsf{N}_{i,j} = \langle \mathsf{id}(D_{i,j}) || \kappa_{i,j} || \psi_s(\mathsf{ctr}+1) \rangle$, where $\mathsf{id}(D_{i,j})$ is the j^{th} identifier in $\mathcal{D}(w_i)$;
— compute $E_{\kappa_{i,j-1}}(\mathsf{N}_{i,j})$ and store it in $\mathsf{A}[\psi_s(\mathsf{ctr})]$;
— $\mathsf{ctr} = \mathsf{ctr} + 1$
iii. for the last node of L_i (i.e., $\mathsf{N}_{i,|\mathcal{D}(w_i)|}$), before encryption, set the address of the next node to NULL;
(b) let $m' = \sum_{w_i \in \Delta'} |\mathcal{D}(w_i)|$. If $m' < m$, then set remaining $(m - m')$ entries of A to random values of the same size as the existing m' entries of A.
(3) Build look-up table T:
(a) for each $w_i \in \Delta'$:
i. $\mathsf{value} = \langle \mathsf{addr}(\mathsf{A}(\mathsf{N}_{i,1})) || \kappa_{i,0} \rangle \oplus f_y(w_i)$;
ii. set $\mathsf{T}[\pi_z(w_i)] = \mathsf{value}$.
(b) if $|\Delta'| < |\Delta|$, then set the remaining $(|\Delta| - |\Delta'|)$ entries of T to random values.
(4) Output $\mathcal{I} = (\mathsf{A}, \mathsf{T})$.
— $\mathsf{Trapdoor}(w)$: Output $T_w = (\pi_z(w), f_y(w))$.
— $\mathsf{Search}(\mathcal{I}, T_w)$:
(1) Let $T_w = (\gamma, \eta)$. Retrieve $\theta = \mathsf{T}[\gamma]$. Let $\langle \alpha || \kappa \rangle = \theta \oplus \eta$.
(2) Decrypt L starting with the node at address α encrypted under key κ.
(3) Output the list of document identifiers in L.

Fig. 5. Algorithmic description of the first Curtmola et al. [2006] scheme (CGK$^+$-I). This scheme uses an inverted index and achieves sublinear (optimal) search time.

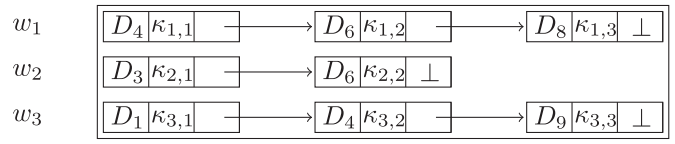
node $\mathsf{N}_{i,0}$ that contains the pointer to the first node $\mathsf{N}_{i,1}$ in L_i and the corresponding key $\kappa_{i,0}$ (cf. Figure 6(b)). The node $\mathsf{N}_{i,0}$ in the look-up table is encrypted (cf. Figure 6(c)) with $f_y(w_i)$, which is a pseudo-random function dependent on the keyword w_i. Finally, the encrypted $\mathsf{N}_{i,0}$ is stored at position $\pi_z(w_i)$, where π is a pseudo-random permutation. Since the decryption key and the storage position per node are both dependent on the keyword, trapdoor generation is simple and outputs a trapdoor as $T_w = (\pi_z(w), f_y(w))$.

The trapdoor allows the server to identify and decrypt the correct node in T, which includes the position of the first node and its decryption key. Due to the nature of the linked list, given the position and the correct decryption key for the first node, the server is able to find and decrypt all relevant nodes to obtain the document's identifiers. The detailed algorithm is shown in Figure 5.

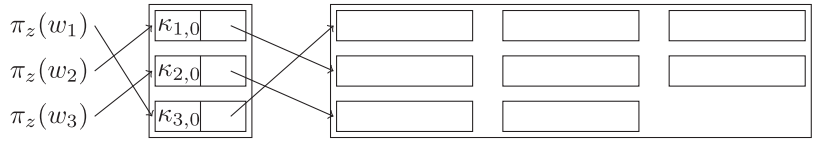
Efficiency: CGK$^+$ propose the first sublinear scheme that achieves **optimal search time**. The **index generation** is linear in the number of distinct words per document. The server computation per **search** is proportional to $|\mathcal{D}(w)|$, which is the number of documents that contain a word w. CGK$^+$-II **search** is proportional to $|\mathcal{D}''(w)|$, which is the maximum number of documents that contain a word w.

Both CKG schemes use a special data structure (FKS dictionary [Fredman et al. 1984]) for a look-up table. This makes the index more compact and reduces the look-up time to $O(1)$. **Updates** are expensive due to the representation of the data. Thus, the scheme is more suitable for a static database than a dynamic one.

Security: CGK-I is consistent with the new **IND-CKA1** security definition. CGK-II achieves **IND-CKA2** security but requires higher communication costs and storage on the server than CGK-I.



(a) CGK$^+$: Linked lists L_i



(b) CGK$^+$: Index table T and encrypted linked lists L_i

(c) CGK$^+$: Encrypted index table T and array A consisting of encrypted and scrambled linked lists L_i.

Fig. 6. BuildIndex algorithm of Curtmola et al. (CGK-I) [2006].

3.1.1.5. Efficiently searchable authenticated encryption. Amanatidis et al. [2007] (ABO) propose two schemes using **deterministic** message authentication codes (MACs) to search. The **idea** of ABO-I (MAC-and-encrypt) is to append a deterministic MAC to an IND-CPA secure encryption of a keyword. The **idea** of ABO-II (encrypt-with-MAC) is to use the MAC of the plaintext (as the randomness) inside of the encryption. The schemes can use any IND-CPA secure symmetric encryption scheme in combination with a deterministic MAC. ABO also discuss a prefix-preserving search scheme. To search with ABO-I, the client simply generates the MAC of a keyword and stores it together with the encrypted keyword on the server. The server searches through the indexed MACs to find the correct answer. In ABO-II, the client calculates the MAC and embeds it inside the ciphertext for the keyword. The server searches for the queried ciphertexts.

Efficiency: In ABO, the **index generation** per document is linear in the number of words. Both schemes require a MAC and an encryption per keyword. The **search** is a simple database search and takes **logarithmic-time** $O(\log v)$ in the database size.

Security: ABO define security for searchable deterministic symmetric encryption like Bellare et al. [2007b] (Section 4.2.1.1), which ABO call IND-EASE. Both schemes are proven **IND-EASE** secure. ABO-I is secure under the assumption that the encryption scheme is IND-CPA secure and the MAC is unforgeable against chosen message attacks (uf-cmas) and privacy preserving. ABO-II is secure, if the encryption scheme is IND-CPA secure and the mac is a pseudo-random function.

See also: Deterministic encryption in the M/M setting [Bellare et al. 2007b] (Section 4.2.1.1).

3.1.1.6. Index per keyword with efficient updates. Van Liesdonk et al. [2010] propose two schemes (LSD-I, LSD-II) that offer efficient search and update, which differ in the communication and computation cost. LSD-$*$ use the same **idea** and are closely related to the CGK schemes (one index per keyword), but in contrast, the LSD schemes support **efficient updates** of the database.

Efficiency: In LSD-I, the **index generation** per document is linear in the number of distinct words. The algorithm uses only simple primitives like pseudo-random functions. The **search** time is logarithmic in the number of unique keywords stored on the server. LSD-I is an **interactive** scheme and requires **two rounds** of communication for the index generation, update, and search algorithms. LSD-II is noninteractive by deploying a hash chain at the cost of more computation for the search algorithm.
Security: The authors prove their schemes **IND-CKA2** secure.

3.1.1.7. Structured encryption for labeled data. Chase and Kamara [2010] (CK) proposed an adaptively secure construction that is based on CGK$^+$-I. The **idea** is to generate an inverted index in the form of a padded and permuted dictionary. The dictionary can be implemented using hash tables, resulting in optimal search time.

Efficiency: **Index generation** requires one initial permutation and two pseudo-random functions per distinct keyword in the database. To **search**, the server searches for the position of the desired query keyword and decrypts the stored values, which are the document IDs of the matching documents.
Security: CK define a generalization of IND-CKA2 security where the exact leakage (e.g., the access or search pattern) can be influenced through leakage functions. This allows them to also hide the data structure from adversaries. However, their actual construction still leaks the access and search pattern. Conceptually, their scheme is **IND-CKA2** secure and in addition hides the data structure.
See also: CK is based on CGK$^+$-I [Curtmola et al. 2006] (cf. Section 3.1.1.4).

3.1.1.8. Verifiable SSE. Kurosawa and Ohtaki [2012] (KO) propose a verifiable SSE scheme that is secure against active adversaries and/or a malicious server. The **idea** is to include a MAC tag inside the index to bind a query to an answer. KO use only PRFs and MACs for building the index. KO define security against active adversaries, which covers keyword privacy as well as reliability of the search results.

Efficiency: **Index generation** requires n PRFs and n MACs per keyword in the database, where n is the number of documents. To **search**, the server performs n table look-ups. **Verification** of the results requires n MACs.
Security: KO is proven **universally composable**(UC) secure. KO's UC security is stronger than IND-CKA2 (cf. Section 2.3)
See also: KO is based on CGK$^+$-II [Curtmola et al. 2006] (cf. Section 3.1.1.4).

3.1.1.9. Dynamic SSE. Kamara et al. [2012] (KPR) propose an extension for the CGK$^+$-I scheme, to allow efficient updates (add, delete, and modify documents) of the database. The **idea** is to add a *deletion array* to keep track of the search array positions that need to be modified in case of an update. In addition, KPR use homomorphically encrypted array pointers to modify the pointers without decrypting. To add new documents, the server uses a *free list* to determine the free positions in the search array. KPR uses only PRFs and XORs.

Efficiency: KPR achieves optimal search time while at the same time handling efficient updates. **Index generation** requires eight PRFs per keyword. To **search**, the server performs a table look-up for the first node and decrypts the following nodes by performing an XOR operation per node. Each node represents a document that contains the search keyword.
Security: KPR define a variant of IND-CKA2 security that, similar to CK (cf. Section 3.1.1.7), allows for parameterized leakage and in addition is extended to include dynamic operations (like adding and deleting items). Conceptually,

their security definition is a generalization of **IND-CKA2**. Updates leak a small amount of information (i.e., the trapdoors of the keywords contained in an updated document). They prove the security in the **random oracle (RO)** model.

See also: KPR is an extension of CGK$^+$-I [Curtmola et al. 2006] (cf. Section 3.1.1.4).

3.1.1.10. Parallel and dynamic SSE. Kamara and Papamanthou [2013] (KP) use the advances in multicore architectures to propose a new dynamic SSE scheme that is highly parallelizable. KP provide a new way to achieve sublinear search time that is not based on Curtmola et al.'s scheme. The **idea** is to use a tree-based multimap data structure per keyword, which they call **keyword red-black** (KRB) trees. KRB trees are similar to binary trees with pointers to a file as leaves. Each node stores information if at least one of its following nodes is a path to a file identifier containing the keyword. These KRB trees can be searched in $O(D(v)\log n)$ sequential time or in parallel $O(\frac{D(v)}{p}\log n)$, where p is the number of processors. KP also allows efficient updates, but with 1.5 rounds of interaction.

Efficiency: **Encryption** requires per distinct keyword in the database $2n-1$ (nodes per tree) encryptions, where n is the number of documents. That is each node of a KRB tree per keyword. **Search** requires $(D(v)\log n)$ decryptions.

Security: KP define a variant of CKA2 security, which is slightly stronger than KPR's (cf. Section 3.1.1.9) CKA2 variant. The difference is that during an update operation (performed before any search operation), no information is leaked. Conceptually, their security definition is a generalization of **IND-CKA2**. KP prove the security in the **RO** model.

3.1.2. Conjunctive Keyword Search. With conjunctive keyword search, we mean schemes that allow a client to find documents containing all of several keywords in a single query (i.e., single run over the encrypted data). Building a conjunctive keyword search scheme from a single keyword scheme in a naïve way provides the server with a trapdoor for each individual keyword. The server performs a search for each of the keywords separately and returns the intersection of all results. This approach leaks which documents contain each individual keyword and may allow the server to run statistical analysis to deduce information about the documents and/or keywords.

3.1.2.1. First conjunctive search schemes. Golle et al. [2004] (GSW) pioneer the construction of conjunctive keyword searches and present two schemes (GSW-I, GSW-II). Their **idea** for conjunctive searches is to assume that there are special keyword fields associated with each document. Emails, for example, could have the keyword fields: "From," "To," "Date," and "Subject." Using keyword fields, the user has to know in advance where (in which keyword field) the match has to occur. The communication and storage cost linearly depend on the number of stored data items (e.g., emails) in the database. Hence, GSW-$*$ are not suitable for large-scale databases.

Efficiency: **Encryption** in GSW-I requires $1+v$ exponentiations per document, where v is the number of keywords per document. GSW-I requires two modular exponentiations per document for each **search**. The size of a trapdoor is linear in the total number of documents. Most of the communication can be done offline, because the trapdoor is split into two parts, and the first part, which is independent of the conjunctive query that the trapdoor allows, can be transmitted long before a query. The second part of the trapdoor is a constant amount of data, which depends on the conjunctive query that the trapdoor allows and therefore must be sent online at query time. After receiving a query, the server combines it with the first part to obtain a full trapdoor.

Encryption in GSW-II requires the client to compute $2v + 1$ exponentiations. To **search**, the server has to perform $2k + 1$ symmetric prime order pairings per document (k is the number of keywords to search). The size of a trapdoor is constant in the number of documents but linear in the number of keyword fields. GSW-II doubles the storage size on the server compared to GSW-I.

Security: GSW extend the IND1-CKA definition to conjunctive keyword searches, meaning that for empty conjunctions (i.e., when querying a single keyword), the definition is the same as IND1-CKA. Therefore, we can say that GSW-I is proven **IND1-CKA** secure in the **RO** model. The security relies on the Decisional Diffie-Hellman **(DDH)** [Boneh 1998] assumption.

The security of GSW-II relies on a new, nonstandard, hardness assumption and is also proven to be **IND1-CKA** secure.

3.1.2.2. Secure in the standard model. Ballard et al. [2005b] (BKM) propose a construction for conjunctive keyword searches, where the **idea** is to use Shamir's Secret Sharing Shamir [1979] (SSS). BKM require keyword fields.

Efficiency: BKM requires a trapdoor size that is linear in the number of documents being searched. **Index generation** uses a pseudo-random function per keyword. The trapdoor and **search** algorithms need to perform a standard polynomial interpolation for the SSS per document.

Security: BKM is proven secure under the same extended **IND1-CKA** definition as GSW (cf. Section 3.1.2.1). The security is based on the security of **SSS** in the **standard model (ST)**.

3.1.2.3. Constant communication and storage cost. Byun et al. [2006a] (BLL) construct a conjunctive keyword search scheme with constant communication and storage cost. The **idea** is to improve the communication and storage costs necessary for large databases by using bilinear maps. Communication of BLL is more efficient than both schemes by Golle et al., but encryption is less efficient. BLL requires keyword fields.

Efficiency: BLL uses symmetric prime order bilinear maps. The **encryption** requires one bilinear map per keyword in a document. The **search** requires two bilinear maps per document.

Security: BLL use the same extended IND1-CKA definition for conjunctive queries as GSW (cf. Section 3.1.2.1). The security of the scheme relies on a new multidecisional bilinear Diffie-Hellman **(MBDH)** assumption, which the authors prove to be equivalent to the decisional Bilinear Diffie-Hellman **(BDH)** assumption [Joux 2002; Boneh and Franklin 2003]. BLL is proven secure under the mentioned extended version of **IND1-CKA** in the **RO** model under the **BDH** assumption.

3.1.2.4. Smaller trapdoors. Ryu and Takagi [2007] (RT) propose an efficient construction for conjunctive keyword searches where the size of the trapdoors for several keywords is nearly the same as for a single keyword. The **idea** is to use Kiltz and Galindo's work [2006] on identity-based key encapsulation. RT requires keyword fields.

Efficiency: RT uses asymmetric pairings [Boneh and Franklin 2003] in groups of prime order. **Encryption** requires one pairing per document and the server has to perform two pairings per document to **search**. RT achieves better performance than previous schemes (computational and communication costs) and has almost the same communication cost as that of searching for a single keyword.

Security: RT use the extended IND1-CKA definition for conjunctive queries (cf. GSW in Section 3.1.2.1). RT is proven secure under their extended **IND1-CKA** definition in the **RO** model under their new variant of the External Diffie-Hellman (XDH)

assumption, in which the DDH problem is mixed with a random element of G_2. They call this the external co-Diffie-Hellman **(coXDH)** assumption. The XDH assumption was first introduced by Scott [2002] and later formalized by Boneh et al. [2004a] and Ballard et al. [2005a].

3.1.2.5. Keyword-field-free conjunctive keyword search. Wang et al. [2008b] (WWP-III) present the first keyword-field-free conjunctive keyword search scheme that is proven secure in the ST model. The **idea** is to remove the keyword fields by using a bilinear map per keyword per document index.

Efficiency: WWP-III uses symmetric bilinear pairings of prime order. The **index generation** constructs a v'-degree polynomial per document, where v' is the number of distinct keywords contained in the document. The algorithm requires $v' + 1$ exponentiations per document. A **search** requires a bilinear map per keyword per document index. The size of a query/trapdoor is linear in the number of keywords contained in the index.

Security: WWP-III is proven secure in the **ST** model under the extended version of **IND1-CKA** from GSW (cf. Section 3.1.2.1). The security is based on the discrete logarithm **(DL)** assumption [Diffie and Hellman 1976] and the l-decisional Diffie-Hellman inversion **(l-DDHI)** assumption [Camenisch et al. 2005].

See also: The authors also extend WWP-III to dynamic groups in the M/M setting (cf. Section 4.2.2.4). The first keyword-field-free conjunctive keyword search scheme in the **RO** model is due to Wang et al. [2008a] (cf. Section 4.2.2.3).

3.1.2.6. Sublinear conjunctive keyword search. Cash et al. [2013] (CJJ$^+$) recently proposed the first sublinear SSE construction supporting conjunctive queries for arbitrarily structured data. The construction is based on the inverted index approach of Curtmola et al. [2006] (Section 3.1.1.4). CJJ$^+$ provide a highly scalable implementation. The **idea** is to query for the estimated least frequent keyword first and then filter the search results for the other keywords. The search protocol is **interactive** in the sense that the server replies to a query with encrypted document IDs. The client has to decrypt these IDs before retrieving the corresponding documents.

Efficiency: The **index generation** requires for each distinct keyword v'' in the database that for all $D(v)$ (documents that contain the keyword), six pseudo-random functions, one encryption, and one exponentiation are computed. A **search** requires the server to perform two PRF, one XOR, and $(k - 1)$ exponentiation per document that contain the query keyword $D(v)$, where k is the number of keywords in the trapdoor.

Security: CJJ$^+$ define a generalization of IND-CKA2 for conjunctive queries, which is parameterized by leakage functions. CJJ$^+$ is proven **IND-CKA2** secure under the generalized definition under the **DDH** assumption.

3.1.3. Extended Queries. In this section, we will discuss schemes that allow more powerful queries (e.g., fuzzy search and inner products).

3.1.3.1. Fuzzy/similarity search using Hamming distance. Park et al. [2007] (PKL$^+$) propose a method to search for keywords with errors over encrypted data, based on approximate string matching. To search for similar words, the **idea** is to encrypt a word character by character and use the Hamming distance to search for similar keywords. Because character-wise encryption is not secure (domain is too limited), they design a new encryption algorithm. PKL$^+$ comes in two versions. PKL$^+$-I is more secure (i.e., achieves query privacy) and PKL$^+$-II is more efficient.

Efficiency: PKL$^+$-$*$ use only pseudo-random functions, pseudo-random generators, one-way functions, and exponentiations. The **index generation of PKL$^+$-I** requires one PRF, one hash, and one exponentiation per character per keyword per document. The trapdoor generation requires a PRF per character of the keyword. To search, the server has to generate a pattern that requires a hash and two exponentiations per character per keyword per stored index. The **search of PKL$^+$-I** is linear in the number of documents and requires the server to compute the Hamming distance between the pattern and a keyword, per keyword per index.

The **index generation of PKL$^+$-II** requires a PRF and a hash per character per keyword per document. The trapdoor algorithm takes ml PRF, where m is the number of keyword fields and l the number of characters of the keyword. The pattern generation requires ml hash functions and the **search of PKL$^+$-II** has to calculate m Hamming distances per index stored on the server.

Security: PKL$^+$ redefine IND1-CKA to their setting by allowing the Hamming distance to leak. The security of PKL$^+$ is based on the **DDH** assumption. Both PKL$^+$ schemes are proven secure under their **IND1-CKA** definition in the **RO** model. PKL$^+$-II does not achieve query privacy, since no random factor in the trapdoor generation is used.

3.1.3.2. Fuzzy search using locality sensitive hashing. Adjedj et al. [2009] (ABC$^+$) propose a fuzzy search scheme for biometric identification. The **idea** is to use locality-sensitive hashing (LSH) to make sure that similar biometric readouts from the same person are hashed to the same value. LSH outputs (with high probability) the same hash value for inputs with *small* Hamming distance. The LSH values are then used in combination with the CGK$^+$-II scheme (Section 3.1.1.4). After a search, the results have to be decrypted on the client.

Efficiency: **Encryption** requires b hash functions, PRPs, and encryptions per document (here: user of the identification system), where b is the number of hash functions used for the LSH. The **search** consists of $b\mathcal{D}''(w)$ database searches, where $\mathcal{D}''(w)$ is the maximum number of user identifiers for a biometric template w.

Security: ABC$^+$ use the standard CGK$^+$-II scheme and is thus **IND-CKA2** secure.

See also: Curtmola et al. [2006] (Section 3.1.1.4).

3.1.3.3. Fully secure search based on inner products. Shen et al. [2009] (SSW) present a symmetric-key predicate encryption scheme that is based on inner products. The **idea** is to represent the trapdoor and the searchable content as vectors and calculate the inner product of those during the search phase. Thus, SSW does not leak which of the search terms matches the query. SSW introduce the notion of predicate privacy (tokens leak no information about the encoded query predicate). SSW also give a definition for **fully secure** predicate encryption, which means that nothing should be leaked, except for the access pattern. The dot product enables more complex evaluations on disjunctions, polynomials, and CNF/DNF formulae.

Efficiency: SSW uses composite order symmetric bilinear pairings where the order of the group is the product of four primes. **Encryption** requires $6v + 2$ exponentiations per document, where v is the number of keywords. Trapdoor generation requires $8v$ exponentiations and the **search** algorithm requires $2v + 2$ pairings per document.

Security: The security of SSW relies on three assumptions: (1) the generalized Assumption 1 from Katz et al. [2008] **(GKA1)**, (2) the generalized three-party Diffie-Hellman **(C3DH)** [Boneh and Waters 2007] assumption, and (3) the decisional

linear **(DLIN)** assumption [Boneh et al. 2004a]. SSW is proven **single challenge** (SC) (attacker is limited to a single instance of the security game) fully secure (FS) in the **selective model** (SEL) [Canetti et al. 2003], where an adversary commits to an encryption vector at the beginning of the security game. SSW hides the search pattern.

3.1.3.4. Fuzzy search using Edit distance. Li et al. [2010] (LWW$^+$) propose a search scheme for fuzzy keyword searches based on prespecified similarity semantics using the Edit distance (number of operations (substitution, deletion, insertion) required to transform one word into another). The **idea** is to precompute fuzzy keyword sets $S_{k,d} = \{S'_{k,0}, S'_{k,1}, \ldots, S'_{k,d}\}$ with Edit distance d per keyword k and store them encrypted on the server. The trapdoors are generated in the same manner, so that the server can test for similarity. The set $S_{\texttt{CAT},1}$ can be constructed as follows, where each $*$ represents an edit operation on that position: $S_{\texttt{CAT},1} = \{\texttt{CAT}, *\texttt{CAT}, *\texttt{AT}, \texttt{C}*\texttt{AT}, \texttt{C}*\texttt{T}, \texttt{CA}*\texttt{T}, \texttt{CA}*, \texttt{CAT}*\}$. The number of set elements is $\sum_{y=0}^{d} \sum_{x=l}^{l+y} \binom{x}{y}$, where d is the distance and l the length of the keyword in characters. The search is **interactive** and requires **two rounds** to retrieve the documents.

Efficiency: **Encryption** requires the client to first construct the fuzzy sets. For each element of the set, a pseudo-random function has to be computed. Upon receiving the trapdoor keyword set, the **search** consists of a comparison per set element per document.

Security: LWW$^+$ slightly modify the IND-CKA1 definition by allowing the encrypted index to leak the Edit distance between the plaintexts underlying the ciphertexts. They prove their scheme secure in this modified **IND-CKA1** definition.

3.1.3.5. Efficient fully secure search. Bösch et al. [2012] (BTH$^+$) propose a scheme that is also based on inner products (cf. SSW in Section 3.1.3.3). It uses the index generation technique from Chang and Mitzenmacher [2005] in combination with *somewhat* homomorphic encryption [Gentry 2010]. The **idea** is to separate the query phase from the document retrieval by introducing **another round** of communication. This makes BTH$^+$ more flexible and allows one to selectively retrieve documents efficiently. BTH$^+$ uses recent advantages in Lattice-based cryptography [Brakerski and Vaikuntanathan 2011], which makes BTH$^+$ more efficient ($\sim$1,250$\times$ faster than SSW) than pairing-based schemes. BTH$^+$ can be combined with techniques like PIR to hide the access pattern, but it requires an additional round of communication.

Efficiency: **Index generation** requires v'' encryptions per document, where v'' is the total number of distinct keywords in the document collection. To **search**, the server has to compute v'' polynomial multiplications and $v'' - 1$ polynomial additions per index.

Security: BTH$^+$ is proven **fully secure** under the assumption that the homomorphic encryption scheme is IND-CPA secure. BTH$^+$ hides the search pattern and can be extended with PIR to also hide the access pattern.

3.1.3.6. Efficient similarity search. Kuzu et al. [2012] (KIK) propose a generic similarity search construction based on LSH and BF (cf. Adjedj et al. [2009] in Section 3.1.3.2 and Bringer et al. [2009] in Section 4.1.3.3). The **idea** for their keyword search scheme is to represent keywords as n-grams and insert each n-gram into the BF using LSH. To measure the distance for the similarity search, the Jaccard distance is used. The protocol is **interactive** and requires **two rounds** of communication to retrieve the matching documents.

Efficiency: **Index generation** requires a metric space translation for each distinct keyword per document, b LSH functions per keyword, and two encryptions per BF bucket. To **search**, the server has to search for b buckets in the first round. The client decrypts the search result and sends some document identifiers to the server. The server replies with the encrypted documents.

Security: KIK adapt the IND-CKA2 security definition of Curtmola et al. [2006] (cf. Section 3.1.1.4) to their setting (allow the leakage of the similarity pattern) and prove their scheme **IND-CKA2** secure under the adapted definition.

3.1.4. Synthesis. The S/S architecture has been the subject of active research for over a decade now and still new schemes are developed. Most of the schemes focus on single and conjunctive keyword searches, but more powerful queries are also possible. The schemes that try to achieve a higher level of security or a better query expressiveness are likely to be more complex or use more expensive primitives and are thus less efficient.

In the beginning of SE research with the S/S architecture, there were no formal security definitions for searchable encryption. It took several years until the first definitions were available, and still researchers do not use a common security model to prove their schemes secure. Some schemes are based on new assumptions and not proven secure under standard or well-known assumptions, which makes it hard to assess the security of a scheme and compare it to others. Also, some authors allow the leakage of the search pattern in their schemes, whereas others want to hide as much information as possible.

Twenty-six out of 28 schemes in the S/S setting leak at least the access pattern and the search pattern. The SSW scheme does only leak the access pattern, and BTH$^+$ can even be extended to also hide the access pattern if desired. SSW and BTH$^+$ both calculate the dot product of the trapdoor and the searchable content. Thus, the schemes do not leak which of the keywords match the query, but the search complexity is linear in the number of keywords.

All but eight papers (cf.Table I) propose schemes that achieve at best a search complexity of $O(n)$, which is linear in the number of documents stored in the database. The eight exceptions (cf. gray search fields in Table I) introduce schemes, which achieve sublinear search times. The schemes achieve at least a search complexity logarithmic in the total number of keywords in the database, since the search consists of a standard database search, which can be realized using a binary or hash tree (LSD$^+$). Some schemes (CGK$^+$, CK, KPR, KP, CJJ$^+$) even achieve optimal search time (i.e., the number of documents that contain the query keyword). These schemes require deterministic trapdoors that inherently leak the search pattern, since the server can directly determine whether two searches use the same predicate. Another drawback of some of these schemes is interactiveness, either in the database update (CKG$^+$) or in the update, search, and encrypt phase (LSD$^+$). This is due to the fact that the database consists of an index per keyword (inverted index) instead of an index per document (forward index). The schemes achieve the best search complexity, but since the update operation is expensive, they are best suited for static databases. The implementation of the CJJ$^+$ scheme is the most scalable but uses an interactive search protocol.

Table I gives a detailed overview of the computational complexity and the security of the different algorithms of the discussed schemes. The digest of the table can be found in the reading guidelines in Section 1.5 and the legend in Table II.

3.2. Single Writer/Multireader (S/M)

In a single writer/multireader (S/M) scheme, the secret key owner is allowed to create searchable content, whereas a user-defined group is allowed to generate trapdoors.

Table I. Comparison of Different S/S Schemes. The Legend is in Table II.

Scheme/Section	Efficiency of algorithms: Encrypt	Trapdoor	Search	Security: Def.	Ass.	ROM	Notes
Single Keyword Equality Test							
SWP / 3.1.1.1	$vn\mathsf{E}$	$k\mathsf{E}$	$vn\mathsf{E}$	IND-CPA	SP*	-	sequential scan
Goh / 3.1.1.2	$v'n\mathsf{f}$	$k\mathsf{f}$	$kn\mathsf{f}$	IND1-CKA	SP	-	BF index
CM-I / 3.1.1.3	$n(v'+v'')\mathsf{f}$	$2k\mathsf{f}$	$n\mathsf{f}$	IND2-CKA	SP	-	dictionary on client
CM-II / 3.1.1.3	$n(v'+2v'')\mathsf{f}$	$2k\mathsf{f}$	$n\mathsf{f}$	IND2-CKA	SP	-	trapdoor interactive
CGK$^+$-I / 3.1.1.4	$3v''n\mathsf{f}+nD''(v)\mathsf{E}$	$2k\mathsf{f}$	$D(v)\mathsf{D}$	IND-CKA1	SP	-	optimal search time, static $\mathcal{DB}$
CGK$^+$-II / 3.1.1.4	$nv''D''(v)\mathsf{f}$	$kD''(v)\mathsf{f}$	$D''(v)\mathsf{TLU}$	IND-CKA2	SP	-	static $\mathcal{DB}$
ABO-I / 3.1.1.5	$nv(\mathsf{h}+\mathsf{E})$	$k(\mathsf{h}+\mathsf{E})$	$k\mathsf{s}$	deterministic	SP	-	deterministic macs
ABO-II / 3.1.1.5	$nv(\mathsf{h}+\mathsf{E})$	$k(\mathsf{h}+\mathsf{E})$	$k\mathsf{s}$	deterministic	SP	-	deterministic macs
LSD$^+$-I / 3.1.1.6	$nv'\mathsf{f}+nD(v)(\mathsf{D}+\mathsf{E})$	$k\mathsf{f}$	$k(\mathsf{D}+\mathsf{f})$	IND-CKA2	SP	-	interactive, dynamic $\mathcal{DB}$
LSD$^+$-II / 3.1.1.7	$nv'(\mathsf{H}+\mathsf{E})$	$k(\mathsf{f}+\mathsf{H})$	$D(v)(\mathsf{h}+\mathsf{D})$	IND-CKA2	SP	-	dynamic $\mathcal{DB}$
CK / 3.1.1.8	$2v''\mathsf{f}+D(v)\mathsf{E}$	$2k\mathsf{f}$	$D(v)\mathsf{D}$	IND-CKA2^t	SP	-	static $\mathcal{DB}$
KO / 3.1.1.8	$2nv''\mathsf{f}$	$nk\mathsf{f}$	$n\,\mathsf{TLU}$	UC-CKA2	SP	-	secure against malicious adv.
KPR / 3.1.1.9	$8nv''\mathsf{f}$	$3k\mathsf{f}$	$D(v)\mathsf{D}$	IND-CKA2^t	SP	✓	dynamic $\mathcal{DB}$
KP / 3.1.1.10	$(2n-1)v'(3\mathsf{f}+\mathsf{E})$	$2k\mathsf{f}$	$D(v)\log n\mathsf{D}$	IND-CKA2^t	SP	✓	dynamic $\mathcal{DB}$, parallel
Conjunctive Keyword Equality Test							
GSW-I / 3.1.2.1	$n(v+1)\mathsf{e}$	$n\mathsf{e}+k\mathsf{f}$	$2n\mathsf{e}$	IND1-CKA^t	DDH	✓	split trapdoor
GSW-II / 3.1.2.1	$n(2v+1)\mathsf{e}$	$3\mathsf{e}+k\mathsf{f}$	$n(2k+1)\mathsf{e}$	IND1-CKA^t	NS**	-	
BKM / 3.1.2.2	$v'n\mathsf{f}$	$k\mathsf{i}$	$n\mathsf{i}$	IND1-CKA^t	SP	-	Shamir's Secret Sharing
BLL / 3.1.2.3	$vn\mathsf{p}_\mathsf{p}^\mathsf{s}$	$3k\mathsf{e}$	$2n\mathsf{p}_\mathsf{p}^\mathsf{s}$	IND1-CKA^t	BDH	✓	
RT / 3.1.2.4	$n(v+1)\mathsf{e}+n\mathsf{p}_\mathsf{p}^\mathsf{a}$	$(m+1)\mathsf{e}$	$2n\mathsf{p}_\mathsf{p}^\mathsf{a}$	IND1-CKA^t	coXDH	✓	smaller trapdoors
WWP-III / 3.1.2.5	$v'n\mathsf{e}$	$2kv'\mathsf{e}$	$v'n\mathsf{p}_\mathsf{p}^\mathsf{s}$	IND1-CKA^t	DL, 1-DDHI	-	no keyword fields
CJJ$^+$ / 3.1.2.6	$v''D(v)(6\mathsf{f}+\mathsf{E}+\mathsf{e})$	$D(v)(k-1)\mathsf{e}$	$D(v)(k-1)\mathsf{e}$	IND-CKA2^t	DDH	-	interactive search (client D)

Table I. Cotinued

Scheme/Section	Efficiency of algorithms Encrypt	Trapdoor	Search	Security Def.	Ass.	ROM	Notes
Single Fuzzy Keyword Test							
PKL$^+$-I / 3.1.3.1	$lvn\mathsf{e}$	$kl\mathsf{f}$	$2lvn(\mathsf{e}+H)$	IND1-CKA^t	DDH	✓	character-wise encryption
PKL$^+$-II / 3.1.3.1	$2lvn\mathsf{f}$	$ml\mathsf{f}$	$ml\mathsf{f}+nmH$	IND1-CKA^t	DDH	✓	character-wise encryption
ABC$^+$ / 3.1.3.2	$nv'b(\mathsf{h}+\mathsf{f}+\mathsf{E})$	$k(b\mathsf{h}+D''(v)\mathsf{f})$	$bD''(v)\mathsf{s}$	IND-CKA2	SP	-	LSH, BFS
LWW$^+$ / 3.1.3.4	$n\lvert S\rvert\mathsf{f}$	$k\lvert S\rvert\mathsf{f}$	$n\lvert S\rvert\mathsf{c}$	IND-CKA1^t	SP	-	pre-computed sets, interactive
KIK / 3.1.3.6	$nv'b(\mathsf{h}+\mathsf{f}+\mathsf{E})$	$kb(\mathsf{h}+\mathsf{f})$	$b\mathsf{s}$	IND-CKA2^t	SP	-	LSH, BF, interactive
Keyword Search based on Inner Product							
SSW / 3.1.3.3	$n(6v+2)\mathsf{e}$	$8v\mathsf{e}$	$n(2v+2)\mathsf{p}^{\mathsf{s}}_{\mathsf{c}^4}$	FS	C3DH, DLIN	-	hides Search Pattern
BTH$^+$ / 3.1.3.5	$nv''\mathsf{E}$	$v''\mathsf{E}$	$nv''\mathsf{m}$	FS	SP	-	hides Search Pattern

* Secure primitives - scheme is secure if the underlying primitives exist/are secure (generic construction).
** New nonstandard hardness assumption.
^{t} Security definition conceptually as the one stated, but tailored to a specific setting (see respective Section).
gray Sublinear search time (optimal include $D(v)$).

Table II. Legend for S/S Schemes

Amount		Primitive	
n	number of documents	$\mathsf{p}_{\mathsf{p}}^{\mathsf{s}}$	symmetric prime order pairing
v	number of keywords per document	$\mathsf{p}_{\mathsf{p}}^{\mathsf{a}}$	asymmetric prime order pairing
v'	number of distinct keywords per document	$\mathsf{p}_{\mathsf{c}^4}^{\mathsf{s}}$	composite order pairing of degree 4
v''	number of distinct keywords in the database	e	exponentiation
k	number of keywords per trapdoor	f	pseudo-random function, permutation
l	length of keyword in characters	h	hash function, MAC
m	number of keyword fields	H	hash chain
$D(v)$	number of documents containing word v	H	Hamming distance
$D''(v)$	max. number of documents containing word v	m	polynomial multiplication
d	Edit distance	i	polynomial interpolation
$\|S\|$	$\sum_{y=0}^{d} \sum_{x=l}^{l+y} \binom{x}{y}$ (size of set)	E, D	encryption, decryption
b	number of LSH functions	s, c	search, comparison
		TLU	table look-up

For historical reasons, we start this section with a nonproven seminal scheme that is worth mentioning. The discussed schemes in this section allow only single equality test queries.

3.2.1. Single Equality Test. Exact keyword match for a single search keyword.

3.2.1.1. Worth mentioning. The following scheme does not fit in the structure of the survey by means of our selection criteria, since the authors do not provide a security proof. Nevertheless, the idea of the authors is worth mentioning.

Using Bloom filter with group ciphers. Bellovin and Cheswick [2004] (BC) present a multiuser scheme based on encrypted Bloom filters and **group ciphers** such as Pohlig-Hellman encryption. They introduce a semitrusted **third party,** which is able to cryptographically transform an encrypted search query for a user's database to a query for another user's database, without leaking the query to either the third party or the database owner. BC, like Goh, uses one Bloom filter per document. Instead of hash functions, a group cipher is used where operations can be done on encrypted data. Due to the use of a Bloom filter per document, BC allows **false positives**.

3.2.1.2. Using broadcast encryption on top of single-user SSE. Curtmola et al. [2006] define SSE in a multiuser setting, where only the data owner is able to write to the document collection, but an arbitrary group of users is allowed to query the data. They propose a **general construction**, where the **idea** is to use **broadcast encryption** (BE) [Fiat and Naor 1994] on top of a single-user scheme. BE allows the data owner to distribute the secret key that is used for the SE scheme to a group of users. This allows all users in possession of the key to create trapdoors and thus to search. As an example, they use their single-user SSE scheme as described in Section 3.1.1.4.

Efficiency: The efficiency depends on the underlying SE scheme.
Security: The security depends on the underlying SE scheme. Curtmola et al. provide a proof that the new multiuser scheme achieves revocation; that is, revoked users are no longer able to perform searches.

3.2.1.3. Using reroutable encryption. The **idea** of Raykova et al. [2009] (RVB$^+$) is to introduce reroutable encryption that allows one to transform encryptions under different keys without leaking the encrypted message. They use another entity **(third party),** a so-called query router that protects the identity of the clients and checks

their authorization on behalf of the server. Thus, their scheme allows one to search other users' data anonymously.

A client can submit an encrypted query to the query router, who checks the authorization of the user. If the user is in the group of authorized users, the query router transforms the query and forwards it to the server. The server sends back the search results to the query router, which transforms the results and forwards them to the user. Due to the use of a Bloom filter per document, RVB$^+$ allows for **false positives**.

Efficiency: To achieve more efficiency (**sublinear** in the size of the data), RVB$^+$ sacrifice the strict definitions of security and privacy by using private key deterministic encryption and a Bloom filter index per document. The **index generation** algorithm has to create a Bloom filter per document. This takes time linear in the number of distinct keywords per document. The trapdoor generation is a single encryption of the search word for the client and a transformation step for the query router. The **search** operation is a Bloom filter look-up per document.

Security: RVB$^+$ is the second discussed scheme that uses deterministic encryption. RVB$^+$ define DET-CCA security, following the idea of Bellare et al. [2007b] (Section 4.2.1.1). The construction is **DET-CCA** secure in the **RO** model under the **DL** hardness assumption. The system **leaks** the search pattern to the query router. The security is based on a trust assumption, which is achieved by splitting the server into several parties.

See also: The idea of using deterministic encryption as a tradeoff between security and efficiency was first introduced by Bellare et al. [2007b], who defined deterministic encryption in the public key setting (see Section 4.2.1.1), and by Amanatidis et al. [2007], who defined it in the symmetric key setting (Section 3.1.1.5).

3.2.1.4. Using bilinear maps. Yang et al. [2011] (YLW) propose a new scheme, which is an adaptation of the M/M scheme from earlier work by Yang et al. [2009], discussed in Section 4.2.1.3. In YLW, each authorized user has a distinct query key, which allows easy user revocation and accountability. Revoked users lose all their search privileges, also on old data. The search algorithm uses symmetric bilinear maps of prime order. The **idea** is that with the bilinear map, the user's trapdoor (which includes the distinct user key), and the user's helper key, the server can calculate a common key to search the index.

YLW requires a **secure channel** to send the query result back to the querying user, since all users share a single record encryption key, which allows also revoked users to decrypt the search result. The authors suggest using a public key encryption to decrypt the results. The authors also present straightforward **extensions** for conjunctive and wildcard searches.

Efficiency: **Encryption** requires the client to compute a symmetric bilinear map of prime order per distinct keyword per document. The **search** algorithm needs to perform one pairing operation per search.

Security: YLW extend the IND-CKA2 security definition to the multiuser setting. Search patterns leak per user, such that queries from different users are unlinkable. YLW is proven secure in the **RO** model under the **DDH** and the computational Diffie-Hellman **(CDH)** [Diffie and Hellman 1976] assumptions in their extended **IND-CKA2** definition.

3.2.2. Synthesis. The S/M architecture has not received a lot of research attention yet. Curtmola et al. [2006] proposed a generic combination of broadcast encryption and any S/S scheme. Recently, two provably secure schemes were proposed. Both schemes support only single-keyword equality tests and are an example for the tradeoff of security versus efficiency. The more secure a scheme is, the more complex it gets and

Table III. Comparison of Different S/M Schemes. The Legend is in Table IV

Scheme/Section	Efficiency			Security			Notes
	Encrypt	Trapdoor	Search	Def.	Ass	ROM	
Single Keyword Equality Test							
CGK$^+$ / 3.2.1.2	generic construction: dependent on the underlying SSE and BE schemes						BE
RVB$^+$ / 3.2.1.3	$v'n\mathsf{E}$	$k\mathsf{E}$	$n\mathsf{B}$	deterministic	DL	✓	FP
YLW / 3.2.1.4	$v'n\mathsf{p}_\mathsf{p}^\mathsf{s}$	$k\mathsf{e}$	$1\mathsf{p}_\mathsf{p}^\mathsf{s} + nv'\mathsf{h}$	IND-CKA2^t	DDH, CDH	✓	

^{t}Conceptually IND-CKA2, but tailored to a specific setting (see respective section).

Table IV. Legend for S/M Schemes

Amount		Primitive	
n	number of documents	$\mathsf{p}_\mathsf{p}^\mathsf{s}$	symmetric prime order pairing
v'	number of distinct keywords per document	e	exponentiation
k	number of keywords per trapdoor	h	hash function
		E	encryption
		B	Bloom filter look-up

is thus less efficient. The search algorithm of Raykova et al. [2009] is linear in the number of documents, but the scheme uses deterministic encryption and directly leaks the search pattern in addition to the access pattern. Yang et al. [2011] achieve a higher level of security, but the search is linear in the number of keywords per document. The schemes in this setting usually introduce a TTP for user authentication or re-encryption of the trapdoors.

Table III gives a detailed overview of the computational complexity and the security of the different algorithms of the discussed schemes. The digest of the table can be found in the reading guidelines in Section 1.5 and the legend in Table IV.

4. MULTIWRITER SCHEMES (M/∗)

This section deals with the M/S and M/M schemes.

4.1. Multiple Writer/Single Reader (M/S)

Most of the schemes in this section are variants of PEKS. The main scenarios for PEKS-like schemes are retrieving emails or documents from a server and allowing a server to redirect/route emails. Usually, multiple users (in possession of the public key) can generate searchable ciphertexts, which can be searched by the private key holder.

4.1.1. Single Equality Test. With an equality test, we mean an exact keyword match for a single search keyword.

4.1.1.1. PEKS - public key encryption with keyword search. Boneh et al. [2004b] (BCO$^+$) propose the first searchable encryption scheme using a public key system. The **idea** for their PEKS scheme is to use identity-based encryption (IBE), in which the keyword acts as the identity. Due to the use of a PKE, each user in BCO$^+$ is allowed to create searchable content with the recipient's public key. Only the private key holder is able to generate a trapdoor to search inside the encrypted data. The construction is based on Boneh and Franklin's work on IBE [2001, 2003].

Details: To create a searchable ciphertext, the sender encrypts his or her message with a standard public key system and appends the PEKS of each keyword (i.e., a publicly known string encrypted under the public key associated with the keyword as identity) (cf. Figure 8). The sender then sends the following ciphertext:

$$E_{K_{pub}}(M)||C_1 = \mathsf{PEKS}(K_{pub}, w_1)|| \ldots ||C_m = \mathsf{PEKS}(K_{pub}, w_m).$$

The size p of G_1, G_2 is determined by the security parameter. The scheme requires two hash functions $H_1 : \{0,1\}^* \to G_1$ and $H_2 : G_2 \to \{0,1\}^{\log p}$ and a bilinear map $e : G_1 \times G_1 \to G_2$.

— Keygen: Pick a random $\alpha \in \mathbb{Z}_p^*$ and a generator g of G_1. It outputs $K_{pub} = (g, h = g^\alpha)$ and $K_{priv} = \alpha$
— PEKS(K_{pub}, w):
(1) Compute $t = e(H_1(w), h^r) \in G_2$ for a random $r \in \mathbb{Z}_p^*$.
(2) Output $C = [g^r, H_2(t)]$.
— Trapdoor(K_{priv}, w): Output $T_w = H_1(w)^\alpha \in G_1$.
— Search(A_{pub}, C, T_w): let $C = [A, B]$. Test if $H_2(e(T_w, A)) = B$, which is equal to testing if $H_2(e(H_1(w)^\alpha, g^r)) = H_2(e(H_1(w), g^{\alpha r}))$. If so, output 'yes'; if not, output 'no'.

Fig. 7. Public key encryption with keyword search (PEKS) [Boneh et al. 2004b] (Section 4.1.1.1).

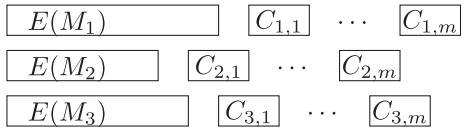


Fig. 8. PEKS. The ciphertexts $C_{i,j}$ use the keywords as identity for the IBE system and are then appended to the encrypted message $E(M_i)$ (Section 4.1.1.1).

To search, the receiver uses the master secret key to derive a secret key for a specific keyword it wants to search for (i.e., the keyword is the identity used for the secret key). The resulting secret key is used as the trapdoor and sent to the server (e.g., email server). The server tries to decrypt all the IBE ciphertexts. If the decryption is successful (i.e., results in the publicly known string), the attached encrypted message contains the keyword. The detailed algorithm is shown in Figure 7.

Efficiency: BCO$^+$ uses symmetric prime order pairings. The **encryption** requires the server to perform one pairing computation, two exponentiations, and two hashes per keyword. The **search** complexity is linear (one map, one hash) in the number of keywords per document.

Security: BCO$^+$ is proven **PK-CKA2** secure in the **RO** model under the **BDH** assumption. BCO$^+$ requires a **secure channel** to transmit the trapdoors, so that an eavesdropper cannot get hold of a trapdoor. The trapdoors for a keyword are never refreshed. The scheme is vulnerable to an of-line keyword-guessing attack [Byun et al. 2006b; Yau et al. 2008], as explained in Section 4.1.4. In the current model, the server is able to store a trapdoor and use it for future documents, which means that the current PEKS is a one-time system.

See also: Baek et al. [2008] (Section 4.1.1.6) address the problem of the secure channel and the trapdoor refreshing. Abdalla et al. [2008] (Section 4.1.1.2) formally define AIBE and present generic transformations from (H/A)IBE to PEKS.

4.1.1.2. Temporary keyword search (PETKS). Abdalla et al. [2008] (ABC^{++}) formalize anonymous IBE (AIBE) and present a generic SE construction by transforming an AIBE scheme into a searchable encryption scheme. The idea underlying the aibe-2-peks transformation was first given by Boneh et al. [2004b] (Section 4.1.1.1). ABC^{++} also give a hierarchical IBE (HIBE) transformation (hibe-2-petks), which allows one to transform an HIBE scheme into a PETKS. The **idea** behind PETKS is to generate a trapdoor that is only valid in a specific time interval. With the time interval included in the trapdoor, the server cannot use the trapdoors to search in past or future ciphertexts (outside the time interval).

Efficiency: The efficiency depends on the used HIBE scheme.

Security: The new PETKS scheme that results from the hibe-2-petks transformation is **PK-CKA2** secure in the **RO** model if the HIBE scheme is IND-CPA secure.

See also: Boneh et al. [2004b] (Section 4.1.1.1)

4.1.1.3. Combining PKE and PEKS in a secure way. Baek et al. [2006] (BSS-I) discuss the problems that arise from combining public key encryption (PKE) schemes with PEKS. They give a concrete construction where the **idea** is to combine a variation of the ElGamal cryptosystem ElGamal [1985], PEKS, and the randomness reuse technique of Kurosawa [2002]. The authors also give a generic PKE/PEKS construction. We discuss only their ElGamal construction, which is proven secure in the paper. The authors give two extensions to their scheme to the multireceiver setting and the multikeyword setting.

Efficiency: BSS-I uses symmetric bilinear maps of prime order. The **encryption** algorithm needs to perform three exponentiations and one mapping per keyword per document. The **search** algorithm requires one bilinear map per keyword per document.

Security: BSS-I is proven **PK-CKA2** secure in the **RO** model assuming that the **CDH** problem is intractable.

4.1.1.4. PEKS based on Jacobi Symbols. Crescenzo and Saraswat [2007] (CS) present the first PEKS scheme that is not based on bilinear maps, but on Jacobi symbols. Their **idea** is to use a transformation of Cocks' identity-based encryption scheme [2001], which is based on the quadratic residuosity problem.

Efficiency: To **encrypt** the data, $4k$ Jacobi symbols per keyword have to be calculated, where k (the length of a keyword in bit) is a scheme's parameter to guarantee the consistency. The authors choose $k = 160$ as an example. The **search** algorithm is linear $(4k)$ in the number of ciphertexts. The storage and communication complexity is high.

Security: The security of CS is based on a variant of the well-known quadratic residuosity problem, namely, the quadratic indistinguishability problem **(QIP)**. CS is proven **PK-CKA2** secure in the **RO** model.

4.1.1.5. K-resilient PEKS (KR-PEKS). Khader [2007] constructs a scheme based on k-resilient IBE [Heng and Kurosawa 2004, 2006]. The **idea** is to use the ability of constructing a PEKS scheme from an IBE. Khader also gives a construction for **multiple keywords** and a **secure-channel-free** PEKS scheme. The main goal of the work was to construct a PEKS scheme that is secure in the standard model.

Efficiency: **Encryption** requires $5 + 3v$ exponentiations, where v is the number of keywords per document. To **search**, four exponentiations have to be calculated per keyword per ciphertext.

Security: Khader's scheme is proven **PK-CKA2** secure in the **ST** model under the **DDH** assumption.

4.1.1.6. Secure-channel-free PEKS. Baek et al. [2008] (BSS-II) remove the need for a secure channel for transmitting the trapdoors in the original PEKS [Boneh et al. 2004b] scheme. The **idea** is to add a server public/private key pair to PEKS and use the aggregation technique from Boneh et al. [2003]. By adding a server key pair, only the server chosen by the sender **(designated tester)** is able to search. The authors also address the problem of refreshing the trapdoors for the same keyword from PEKS.

Efficiency: BSS-II uses symmetric prime order pairings. **Index generation** requires two pairings and an exponentiation per keyword per document. To **search**, the server has to compute one pairing per keyword per ciphertext.

Security: BSS-II is proven **PK-CKA2** secure in the **RO** model under the **BDH** assumption.

4.1.1.7. PEKS with designated tester. Rhee et al. [2009] (RPS$^+$-I) enhance the security model of the PEKS construction of Baek et al. [2008] (cf. Section 4.1.1.6) and construct a PEKS scheme that is secure in the enhanced model. The **idea** is to use a new public key structure, where the public key consists of three components. Their enhanced security model allows an attacker to obtain the relation between ciphertexts and a trapdoor. In addition, the attacker publishes only the public key and not the secret key as in Baek et al.'s security model. RPS$^+$-I is proven secure in their enhanced model.

Efficiency: RPS$^+$-I uses symmetric prime order groups. **Encryption** requires initially seven pairings and then one pairing operation and two exponentiations per keyword. To **search**, the server has to perform one pairing and one exponentiation per keyword per document.
Security: RPS$^+$-I is proven **PK-CKA2** secure in the **RO** model under the **BDH** assumption and the bilinear Diffie-Hellman inversion **(1-BDHI)** assumption [Mitsunari et al. 2002; Boneh and Boyen 2004].
See also: This is an improved version of Baek et al. [2008] (Section 4.1.1.6).

4.1.1.8. Outsource partial decryption. Liu et al. [2009] (LWW) propose a new scheme where the **idea** is to outsource parts of the decryption process to the service provider and thus reduce the computational decryption overhead of the user.

Efficiency: LWW uses symmetric prime order pairings. **Index generation** requires one pairing and one exponentiation per keyword. To **search**, the server has to compute two pairings.
Security: LWW is proven **PK-CKA2** secure in the **RO** model under the **BDH** assumption.

4.1.1.9. Registered keyword search (PERKS). Tang and Chen [2009] (TC) propose the concept of public key encryption with registered keyword search (PERKS). The **idea** is to allow a writer to build searchable content only for the keywords that were previously registered by the reader. This makes TC more robust against an offline keyword-guessing attack.

Efficiency: TC uses symmetric prime order pairings. The **encryption** requires two exponentiations and one mapping per distinct keyword. To **search**, the server has to compute one pairing per distinct keyword per index. **Keyword registration** requires computing one hash value.
Security: TC is proven **PK-CKA2** secure in the **RO** model under the **BDH** assumption.

4.1.1.10. Combining PEKS with PKE (PEKS/PKE). Zhang and Imai's [2009] (ZI) **idea** is to use a hybrid model to combine PKE and PEKS into a single scheme that uses the same key pair for both primitives. The authors give a generic construction and a concrete instantiation using an anonymous IBE by Gentry [2006] and the tag-KEM/DEM (key/data encapsulation mechanism) by Kurosawa-Desmedt [2004].

Efficiency: The instantiation of ZI uses symmetric bilinear groups of prime order. The **encryption** requires two pairings and eight exponentiations per keyword and the **search** one pairing and one exponentiation per keyword.
Security: ZI is proven **PK-CKA2/CCA** secure **without ROs** under the assumption that the **Kurosawa-Desmedt tag-KEM/DEM is secure** and the **Gentry IBE is anonymous**. Stand-alone PEKS/PKE may lose data privacy (CCA) [Baek et al. 2006].
See also: KEM/DEM [Shoup 2004], Tag-KEM/DEM [Abe et al. 2005].

4.1.1.11. Trapdoor security in PEKS with designated tester. Rhee et al. [2010] (RPS$^+$-II) propose a scheme that is secure against keyword-guessing attacks (only for outside attackers). The **idea** is to make the trapdoors indistinguishable by introducing a random variable in the trapdoor computation.

Efficiency: RPS$^+$-II uses symmetric prime order groups. **Encryption** requires two exponentiations and one pairing per keyword. To **search**, the server has to perform one pairing and two exponentiations per keyword per document.

Security: RPS$^+$-II is proven **PK-CKA2** secure in the **RO** model under the **BDH** assumption and the **1-BDHI** assumption.

4.1.1.12. Delegated search (PKEDS). Ibraimi et al. [2011] (INH$^+$) give a construction for a public key encryption with delegated search (PKEDS), which is an extension of ElGamal [1985]. The **idea** of INH$^+$ is to allow the server to search each part of the encrypted data, in contrast to previous schemes where only the metadata is searchable. This can be used, for example, to let a server scan messages for malware. INH$^+$ allows two different kinds of trapdoors. One allows searching for a keyword inside a trapdoor and the other allows the server to search directly for a keyword.

Efficiency: INH$^+$ uses bilinear groups of prime order. **Encryption** is the same as ElGamal and requires two exponentiations per keyword. Delegation requires five exponentiations. The trapdoor generation requires two asymmetric pairings and two exponentiations per keyword. To **search** for a keyword inside a trapdoor, three asymmetric pairings and three exponentiations per keyword per ciphertext are required. To **search** for a keyword, the server has to perform three asymmetric pairings and two exponentiations per keyword per ciphertext.

Security: INH$^+$ is proven to be ciphertext and trapdoor indistinguishable (i.e., an adversary (except the server) cannot learn any information about the plaintext keyword) under the symmetric external Diffie-Hellman **(SXDH)** [Ballard et al. 2005a] assumption. INH$^+$ achieves ciphertext one-wayness under the modified CDH **(mCDH)** assumption, which is a stronger variant of the CDH assumption. The mCDH assumption is implied in the BDH problem in Type 3 pairings (BDH-3) [Chatterjee and Menezes 2009]. INH$^+$ is proven secure in the **ST** model. The security model is *weaker* than PEKS, since the server can generate any trapdoor.

4.1.2. Conjunctive Equality Search. See Section 3.1.2 for information on conjunctive keyword searches.

4.1.2.1. PECKS: public key encryption with conjunctive field keyword search. Park et al. [2004] (PKL) study the problem of public key encryption with conjunctive field keyword search (PECKS). The **idea** is to extend PEKS to allow conjunctive keyword searches (CKSs) in the public key setting. PKL present the first two constructions PKL-I and PKL-II with constant trapdoor size that allow CKSs.

Efficiency: Both schemes use symmetric prime order pairings. PKL-I requires the user to perform one pairing computation per distinct keyword for **encryption**. To **search,** the server has to perform one pairing operation per ciphertext.

In PKL-II, a user has to store private keys in proportion to the number of keyword fields. **Encryption** needs one exponentiation per document and the **search** requires two pairings per ciphertext.

Security: PKL adapt the extended version of IND1-CKA from GSW for conjunctive queries to the public key setting by removing encryption oracle queries (since any user can generate trapdoors with the help of the public key). Their adapted definition is basically PK-CKA2. The security of PKL-I is based on the **BDH**

assumption. PKL-II is based on the **BDHI** assumptions. Both schemes are proven secure in the **RO** model in their adapted **PK-CKA2** definition. **Remark:** The proofs do not satisfy their model and PKL-I is broken by Hwang and Lee [2007], who also showed that the proof of PKL-II is incomplete.

4.1.2.2. More secure searchable keyword-based encryption. Park et al. [2005] (PCL) propose a new mechanism that is more secure than previous schemes in certain applications like email gateways. The **idea** is to construct a scheme from PECKS (Section 4.1.2.1) by using a hybrid encryption technique. A user can either create a ***decrypt trapdoor*** or a ***search trapdoor*** for specific keywords. The enhanced security is achieved by introducing the decrypt trapdoor, which can decrypt ciphertexts without the need for the user's private decryption key. In case of email routing, each device could have a different decrypt trapdoor for certain keywords. Thus, the user's private decryption key does not need to be on each device, which makes the scheme more secure against key compromise. The search trapdoor can test whether a ciphertext contains all of the keywords. PCL requires nonempty **keyword fields**.

Efficiency: **Encryption** requires two exponentiations per document. PCL requires two symmetric prime order pairing operations per ciphertext to **search**.

Security: PCL adapt the PK-CKA2 security definition to PK-CCA2 (public key–adaptive chosen ciphertext attack) by allowing an adversary to query a decryption oracle next to the normally allowed trapdoor queries. The security of PCL is based on the q-BDHI assumption and the bilinear collusion attack (q-BCA) assumption [Chen and Cheng 2005]. The q-BCA assumption is equivalent to the $(q+1)$-BDHI assumption [Chen and Cheng 2005]. PCL is proven secure in the tailored **PK-CCA2** definition under the $\boldsymbol{(q+1)}$**-BDHI** assumption in the **RO** model.

4.1.2.3. PECKS with shortest ciphertext and private key size. Hwang and Lee [2007] (HL) propose a public key encryption with conjunctive keyword search (PECK) and introduce a new concept called multiuser PECKS (mPECKS) as described in Section 4.2.2.1. The **idea** of HL is to minimize the communication and storage overhead for the server and also for the user. Hwang and Lee compare the efficiency of their scheme with both PKL schemes [Park et al. 2004] (cf. Section 4.1.2.1).

Efficiency: **Index generation** requires $2+2v$ exponentiations, where v is the number of keywords per document. PECK uses three symmetric bilinear maps of prime order per ciphertext to **search**. HL has the **shortest** ciphertext size compared with previous PECKS schemes and requires only **one private key**.

Security: HL prove their scheme secure in the adapted **PK-CKA2** definition from PKL (cf. Section 4.1.2.1) under the **DLIN** assumption in the **RO** model.

4.1.3. Extended Queries.

4.1.3.1. Conjunctive, subset, and range queries. Boneh and Waters [2007] (BW) develop a PEKS scheme for conjunctive keyword searches from a generalization of AIBE. The **idea** is to use *hidden vector encryption* (HVE) [Boyen and Waters 2006; Shi et al. 2007] for searching in encrypted data. BW supports equality, comparison, general subset queries, and arbitrary conjunctions of those. The authors also present a general framework for analyzing and constructing SE schemes.

Efficiency: **Encryption** requires $5k+3$ exponentiations per keyword per document, where k is the number of characters per keyword. For an equality **search**, the server has to perform $2k-w+1$ symmetric composite order bilinear pairing operations, where k is the number of characters of the searchable keywords and w

the number of wildcard characters in the keyword. The trapdoor size is linear in the number of conjunctive keywords. The ciphertext size is relatively large, due to the use of composite order bilinear groups [Boneh et al. 2005b].

Security: BW is proven **SEL-CKA** secure under the **C3DH** assumption and the **BDH** assumption in the **selective model** (SEL) [Canetti et al. 2003], where an adversary commits to an encryption vector at the beginning of the security game. A security advantage of BW is that it does **not leak** the attribute values upon decryption like other schemes.

4.1.3.2. Multidimensional range queries (MRQED). Shi et al. [2007] (SBC$^+$) propose a scheme that can create trapdoors for a conjunction of range queries over multiple attributes. The **idea** is that each tuple that should be encrypted can be represented as a point in a multidimensional space. Then, a multidimensional range query is equivalent to testing whether a point falls inside a hyperrectangle. To represent ranges, the authors use binary interval trees over integers and use one interval tree per dimension.

Efficiency: SBC$^+$ can be constructed using either asymmetric or symmetric bilinear maps of prime order. **Encryption** requires $8DL + 2$ exponentiations, where D is the number of dimensions and L the depth of a node in the corresponding tree. The **search** algorithm requires the server to compute $5D$ pairing operations per ciphertext.

Security: SBC$^+$ is proven **SEL-CKA** secure under the **BDH** assumption and the **DLIN** assumption in the SEL model. SBC$^+$ **leaks** the attribute values after successful decryption. The authors argue that this is acceptable for the application of encrypted network audit logs.

4.1.3.3. Error-tolerant searchable encryption. The **idea** of Bringer et al. [2009] (BCK) is to use locality-sensitive hashing (LSH) to enable error-tolerant queries. An LSH function outputs the same hash values for similar items, where similarity is measured in the Hamming distance. BCK inserts these LSH values into one Bloom filter with storage [Boneh et al. 2007] (BFS) in encrypted form. If two keywords k, k' are close enough, the LSH outputs the same hash values as input for the BFS, thus allowing error-tolerant queries. The search in BCK is **interactive**. To query the BFS, the scheme uses a PIR protocol to retrieve the encrypted BF positions. The client decrypts all positions and computes the intersection. The result is a set of file identifiers that can be retrieved in a second round.

Efficiency: **Encryption** includes two sets of hash functions (LSH + BFS) and semantically secure PKE per keyword per document. Each modified BFS position will be updated with a private information storage (PIS) protocol. To **search**, a PIR protocol is run to retrieve the content of the BFS positions, which need to be decrypted to obtain the document IDs. Document retrieval requires another round of communication.

Security: BCK is proven **PK-CKA2** secure. BCK hides the search pattern using PIR.

See also: Adjedj et al. [2009] (Section 3.1.3.2).

4.1.3.4. Wildcard PEKS. The **idea** of Sedghi et al. [2010] (SLN$^+$) is to construct a new scheme based on HVE that can be used for wildcard searchable encryption. SLN$^+$ allows wildcard searches over any alphabet, in contrast to previous schemes [Blundo et al. 2009; Iovino and Persiano 2008; Nishide et al. 2009] that work only over binary symbols.

Efficiency: SLN$^+$ uses symmetric bilinear pairings of prime order. **Encryption** requires $(N + 1)(l + 1) + 4$ exponentiations per keyword, where N is an upper bound on the number of wildcard symbols in decryption vectors and l the length of the keyword. The **search** requires three bilinear maps and w exponentiations per keyword, where w is the number of wildcard characters.

While in previous works the size of the decryption key and the computational complexity for decryption is linear in the number of nonwildcard symbols, in SLN$^+$ these are constant.

Security: SLN$^+$ is proven **SEL-CKA** secure under the **DLIN** assumption in the SEL model.

4.1.4. Synthesis. Since 2004, research in the M/S architecture has obtained significant attention and is still an active research direction. As in the S/S architecture, most schemes focus on single and conjunctive keyword searches, but more powerful queries are also possible.

Since there is a wide spectrum of different techniques for PKE, PEKS schemes can be realized using different primitives, such as IBE, first used by BCO$^+$; AIBE and HIBE used by ABC^{++}; or HVE used by BW and SLN$^+$ in the context of SE.

The M/S architecture is a good example of the aforementioned tradeoffs, namely, expressiveness versus efficiency and security versus efficiency. The M/S architecture focuses mainly on theoretical research that tries to achieve a certain level of security or query expressiveness and is not so much focused on efficiency. All but four (16/20) schemes make heavy use of pairing operations (at least for the search algorithm). Most schemes use at least one pairing per document in the search algorithm and some schemes even use one pairing per keyword per document, which is inefficient in practice. Only four schemes (BBO, CS, Khader, and BCK) do not use pairings. The search complexity of all (except one) schemes in this section is at best linear in the number of documents stored on the server. The exception (BBO) uses deterministic encryption and achieves sublinear (logarithmic) search time. If the data is from a small space (low min-entropy), for example, well-known keywords, using deterministic public key encryption is vulnerable to brute force attacks and thus considered insecure for practical purposes.

Seven of the 20 schemes are proven secure in the standard model, whereas 13 schemes are proven secure with random oracles. All of the schemes leak the search pattern and the access pattern. The search pattern is leaked either directly by using a deterministic procedure to generate the trapdoors or indirectly by an of-line keyword-guessing attack as follows.

Offline keyword-guessing attack. A problem of the main PEKS concept is that there is no keyword/predicate privacy. Most PEKS schemes are vulnerable to an offline keyword guessing attack, which allows an attacker to recover the predicate from a trapdoor. The leakage of the access pattern makes this attack possible. The attack is based on the fact that (1) the keyword space is small (and users choose well-known words to search their documents) and (2) the encryption key is public. The attack works as follows:

(1) The attacker captures a valid trapdoor T_w.
(2) With the user's public key and an appropriate chosen keyword w', the attacker runs the Encrypt algorithm to get a searchable ciphertext.
(3) The user's public key, the captured trapdoor, and the ciphertext from (2) are then used to check whether the ciphertext satisfies the trapdoor or not. If so, the chosen keyword is a valid keyword. Otherwise, the attacker continues with (2).

This allows an attacker to recover the keyword inside a trapdoor. Thus, there is no keyword privacy in the M/S architecture when using a PKE.

Table V gives a detailed overview of the computational complexity and the security of the different algorithms of the discussed schemes. The digest of the table can be found in the reading guidelines in Section 1.5 and the legend in Table VI.

4.2. Multiple Writer/Multiple Reader (M/M)

This section deals with the M/M schemes. The main focus of the discussed schemes in this architecture lies on single and conjunctive keyword searches. More powerful queries are not proposed yet.

4.2.1. Single Equality Test. Exact keyword match for a single search keyword.

4.2.1.1. Using deterministic encryption. The **idea** of Bellare et al. [2007b] (BBO) is to make SE more efficient by using deterministic encryption, at the cost of a weaker security model. In particular, the encrypted index—in contrast to the tokens in asymmetric SE—is directly vulnerable to dictionary attacks. To make the ciphertext searchable, a deterministic hash of the keyword is appended to the encryption of the keyword.

Efficiency: BBO can use any public key encryption scheme in combination with any (deterministic) hash function. The **encryption** requires one encryption and one hash per keyword. The **search** consists of a database search for the hash value.

Security: BBO provide a semantic-security definition of privacy for deterministic encryption called PRIV secure. The security definition for deterministic encryption is similar to the standard IND-CPA security definition with the following two exceptions. A scheme provides privacy only for plaintexts with large min-entropy (could be no privacy at all) and the plaintexts have to be independent from the public key. BBO's encrypt-and-hash construction is proven **PRIV** secure in the **RO** model under the assumption that the underlying scheme is IND-CPA secure. Due to the use of deterministic encryption, BBO directly leaks the index information and the search pattern.

See also: Deterministic encryption in the S/S setting [Amanatidis et al. 2007] (Section 3.1.1.5).

4.2.1.2. Proxy re-encryption. Dong et al. [2008] propose two encryption schemes (DRD-I, DRD-II), where each user has its own unique key to encrypt, search, and decrypt data. Both schemes require a trusted key management server to manage the keys.

The **idea** of **DRD-I** is to use an RSA-based proxy re-encryption scheme. Proxy re-encryption was introduced by Blaze et al. [1998] and can be built on top of different cryptosystems. The proxy re-encryption allows the server to transform an encryption under a user key to an encryption under a different key (e.g., the server's key) without leaking any information on the plaintext. Thus, ciphertexts from different users can be transformed to ciphertexts under the server key, which allow multiple users to create searchable encrypted content. In the same way, the trapdoors are created. A user creates a trapdoor for a keyword by encrypting the keyword with the user's key. The server re-encrypts the trapdoor, which allows him or her to search the encrypted database. Also, the decryption requires a re-encryption step to transform a ciphertext under the server key to a ciphertext under the recipient's key. DRD-I uses a semantically secure symmetric encryption algorithm to encrypt the data, but the searchable part uses only a hash function that makes the data searchable but is not semantically secure. Thus, the authors also present an enhanced version of their scheme.

DRD-II also uses the RSA-based proxy re-encryption scheme for the data. The **idea** is to use optimal asymmetric encryption padding (OAEP) [Bellare and Rogaway 1994]

Table V. Comparison of Different M/S Schemes. The Legend is in Table VI

Scheme/Section	Efficiency: Encrypt	Efficiency: Trapdoor	Efficiency: Search	Security: Definition	Security: Assumption	Security: ROM	Notes
Single Keyword Equality Test							
BCO$^+$ / 4.1.1.1	$nv'(2\mathsf{e}+\mathsf{p}_\mathsf{p}^\mathsf{s})$	$k\mathsf{e}$	$nv'\mathsf{p}_\mathsf{p}^\mathsf{s}$	PK-CKA2	BDH	✓	IBE
BSS-I / 4.1.1.3	$nv'(3\mathsf{e}+\mathsf{p}_\mathsf{p}^\mathsf{s})$	$k\mathsf{e}$	$nv'\mathsf{p}_\mathsf{p}^\mathsf{s}$	PK-CKA2	CDH	✓	ElGamal
CS / 4.1.1.4	$nv'4\mathsf{J}$	$k4l\mathsf{h}$	$n4l\mathsf{J}$	PK-CKA2	QIP	✓	Jacobi symbols
Khader / 4.1.1.5	$5+3nv'\mathsf{e}$	$6k\mathsf{P}$	$4nv'\mathsf{e}$	PK-CKA2	DDH	-	IBE
BSS-II / 4.1.1.6	$nv'(\mathsf{e}+2\mathsf{p}_\mathsf{p}^\mathsf{s})$	$k\mathsf{h}$	$nv'\mathsf{p}_\mathsf{p}^\mathsf{s}$	PK-CKA2	BDH	✓	secure ch. free
RPS$^+$-I / 4.1.1.7	$(2nv')\mathsf{e}+(7+nv')\mathsf{p}_\mathsf{p}^\mathsf{s}$	$k\mathsf{e}$	$nv'(\mathsf{e}+\mathsf{p}_\mathsf{p}^\mathsf{s})$	PK-CKA2	BDH, 1-BDHI	✓	improve 4.1.1.6
LWW / 4.1.1.8	$nv'(\mathsf{e}+\mathsf{p}_\mathsf{p}^\mathsf{s})$	$k\mathsf{e}$	$2nv'\mathsf{p}_\mathsf{p}^\mathsf{s}$	PK-CKA2	BDH	✓	outsource decryption
TC / 4.1.1.9	$nv'(2\mathsf{e}+\mathsf{p}_\mathsf{p}^\mathsf{s})$	$k\mathsf{e}$	$nv'\mathsf{p}_\mathsf{p}^\mathsf{s}$	PK-CKA2	BDH	✓	registered keywords
ZI / 4.1.1.10	$nv'(8\mathsf{e}+2\mathsf{p}_\mathsf{p}^\mathsf{s})$	$k\mathsf{e}$	$nv'(\mathsf{e}+\mathsf{p}_\mathsf{p}^\mathsf{s})$	PK-CKA2	SP	-	AIBE
RPS$^+$-II / 4.1.1.11	$2nv'\mathsf{e}+nv'\mathsf{p}_\mathsf{p}^\mathsf{s}$	$3k\mathsf{e}$	$nv'(2\mathsf{e}+\mathsf{p}_\mathsf{p}^\mathsf{s})$	PK-CKA2	BDH, 1-BDHI	✓	secure trapdoors
INH$^+$ / 4.1.1.12	$2nv'\mathsf{e}$	$k(2\mathsf{e}+2\mathsf{p}_\mathsf{p}^\mathsf{a})$	$nv'(3\mathsf{e}+3\mathsf{p}_\mathsf{p}^\mathsf{a})$	PK-CKA2	SXDH, mCDH	-	ElGamal
Conjunctive Keyword Equality Test							
PKL-I / 4.1.2.1	$nv'\mathsf{p}_\mathsf{p}^\mathsf{s}$	$k\mathsf{e}$	$n\mathsf{p}_\mathsf{p}^\mathsf{s}$	PK-CKA2^t	BDH	✓	security broken
PKL-II / 4.1.2.1	$n\mathsf{e}$	$2k\mathsf{e}$	$n2\mathsf{p}_\mathsf{p}^\mathsf{s}$	PK-CKA2^t	BDHI	✓	proof incomplete
PCL / 4.1.2.2	$n2\mathsf{e}$	$k2\mathsf{e}$	$n2\mathsf{p}_\mathsf{p}^\mathsf{s}$	PK-CCA2^t	q-BDHI	✓	req. non-empty KF
HL / 4.1.2.3	$n(2+2v')\mathsf{e}$	$k3\mathsf{e}$	$n3\mathsf{p}_\mathsf{p}^\mathsf{s}$	PK-CKA2^t	DLIN	✓	
Extended Keyword Tests							
BW / 4.1.3.1	$nv'(5l+3)\mathsf{e}$	$k5(l-w)\mathsf{e}$	$nv'(2l-w+1)\mathsf{p}_{\mathsf{c}2}^\mathsf{s}$	SEL-CKA	C3DH, BDH	-	HVE, subset, range
SBC$^+$ / 4.1.3.2	$n(8DL+2)\mathsf{e}$	$k6DL\mathsf{e}$	$n5D\mathsf{p}_\mathsf{p}^{(\mathsf{s}\text{ or }\mathsf{a})}$	SEL-CKA	BDH, DLIN	-	AIBE, range queries
BCK / 4.1.3.3	$nv'b(2\mathsf{h}+\mathsf{E}+\mathsf{PIS})$	$k2b\mathsf{h}$	$kb(\mathsf{PIR}+\mathsf{D})$	PK-CKA2	SP	-	error-tolerant, interactive
SLN$^+$ / 4.1.3.4	$n((N+1)(l+1)+4)\mathsf{e}$	$k(2l+3)\mathsf{e}$	$nv'(w\mathsf{e}+3\mathsf{p}_\mathsf{p}^\mathsf{s})$	SEL-CKA	DLIN	-	HVE, wildcards

^{t}Security definition conceptually as the one stated, but tailored to a specific setting (see respective section).

Table VI. Legend for M/S Schemes

Amount		Primitive	
n	number of documents	$\mathsf{p}_{\mathsf{p}}^{\mathsf{s}}$	symmetric prime order pairing
v'	number of distinct keywords per document	$\mathsf{p}_{\mathsf{p}}^{\mathsf{a}}$	asymmetric prime order pairing
k	number of keywords per trapdoor	$\mathsf{p}_{\mathsf{c}^2}^{\mathsf{s}}$	composite order pairing of degree 2
l	length of keyword in characters	e	exponentiation
w	number of wildcard characters	h	hash function
N	upper bound on wildcard symbols	s	search
D	number of dimensions	J, P	Jacobi symbol, polynomial(s)
L	depth of a node in a tree	E, D	encryption, decryption
b	number of Bloom filter hash functions	PIS/PIR	private information storage/retrieval

to make the ciphertexts indistinguishable. RSA-OAEP has been proven to be secure under the RSA assumption [Fujisaki et al. 2001]. The main difference lies in the keyword encryption. The proxy re-encryption used for the keywords is deterministic. DRD propose to use a semantically secure noninteractive zero-knowledge proof-style witness instead of the proxy re-encryption scheme to make the keyword ciphertexts indistinguishable and give a concrete construction.

Efficiency: **DRD-I:** The **index generation** computes $v + 1$ exponentiations, where v is the number of distinct keywords per document. To **search**, the server re-encrypts (one exponentiation) the trapdoor and tests each keyword per index for equality.

DRD-II: The **index generation** computes $4v + 1$ exponentiations, where v is the number of distinct keywords per document. To **search**, the server re-encrypts (one exponentiation) the trapdoor and then has to compute $4v$ exponentiations.

Security: DRD adapt the IND-CKA1 definition to the M/M setting by giving the adversary access to the public parameters. DRD-II is proven secure under their modified **IND-CKA1** definition. DRD-I and the proxy re-encryption (both schemes) are **One-Way (OW) secure** under the **RSA** assumption in the **RO** model. OW guarantees that it is hard for an adversary to invert a ciphertext encrypted under a user's encryption key and to learn the keyword, even if the adversary holds the public parameters and all server-side key pairs, but without knowing the user key pair.

The main concern with proxy re-encryption schemes comes from a collusion attack, which allows an adversary and a user to recover the master keys if the adversary knows all server-side keys.

4.2.1.3. Bilinear maps. Bao et al. [2008] (BDD$^+$) propose a multiuser scheme, where each user has a distinct secret key to insert his or her own encrypted data to the database, while each user is allowed to query the whole database. The **idea** is to use a bilinear map to make sure that users using different query keys still generate the same index for a keyword. The system allows one to **dynamically** add and revoke users without the distribution of new keys. The index generation and data encryption are **interactive** algorithms.

Efficiency: BDD$^+$ uses symmetric bilinear maps of prime order. The **index generation** requires the client to calculate two hashes and two exponentiations. The server has to compute a bilinear map per distinct keyword per document. The server needs to perform only one pairing operation per trapdoor per **search**.

Security: BDD$^+$ is proven **IND-CKA2** secure under the **DDH** and **CDH** assumptions. The construction uses the BLS short signature scheme (BL4S) [Boneh et al. 2001] for query unforgability. The BL4S achieves unforgability in the **RO** model.

See also: There is also a journal version of the paper [Yang et al. 2009].

4.2.2. Conjunctive Keyword Search. See Section 3.1.2 for information on conjunctive keyword searches.

4.2.2.1. Multireceiver public key encryption. Hwang and Lee [2007] (HLm) study the problem of public key encryption with conjunctive keyword search (PECK) as discussed in Section 4.1.2.3. They introduce the concept of multiuser PECK (mPECK) and present the first mPECK scheme. The **idea** is to use multireceiver PKE [Baudron et al. 2000; Bellare et al. 2000, 2007a] and randomness reuse [Kurosawa 2002; Bellare et al. 2003] to improve the computation and communication complexity. HLm does not require a third party.

Efficiency: **Index generation** requires $1 + u + 2v$ exponentiations per document, where u is the number of users and v the number of distinct keywords per document. To **search**, HLm requires three pairing operations per trapdoor.

Security: HLm adapt their PK-CKA2 definition for conjunctive keyword searches to the multiuser setting by giving the adversary access to n user public keys. In addition, the keyword sets are encrypted with these n user public keys. During the trapdoor query phase, the adversary has to specify a user index and receives the trapdoor for this specific user. HLm is secure under the **DLIN** assumption in the **RO** model in their adapted **PK-CKA2** definition.

4.2.2.2. RSA accumulator. Wang et al. [2007b] (WWP-I) are the first to present a searchable encryption scheme in the M/M setting. The **idea** of WWP-I is to use dynamic accumulators [Benaloh and de Mare 1993; Bari and Pfitzmann 1997; Camenisch and Lysyanskaya 2002] (RSA accumulator for membership authentication), Paillier's cryptosystem Paillier [1999], and blind signatures [Chaum 1982] (mask encrypted data). They propose a new conjunctive keyword scheme, called common secure index for conjunctive keyword-based retrieval, to share encrypted documents in a **dynamic** group without re-encrypting the data.

In contrast to other SE schemes, where the trapdoor generation requires a private key, the trapdoors in WWP-I are generated with public keys. WWP-I uses a **group manager** (GM), which manages the group members, group keys, and user private keys.

The search part of WWP-I is **interactive** in the following way. First, every user encrypts his or her documents and creates a common index, both with the group public key. To search, a client sends a trapdoor and an authentication code to the server. After retrieving the matched documents, encrypted under the group key, the client uses his or her blind signature function to encrypt the documents again and sends the encryptions to GM. GM uses the group secret key to re-encrypt the documents under the user's blind signature function and sends the data back to the client, who can now decrypt using its inverse blind signature function.

Efficiency: **Index generation** uses only one pseudo-random function and multiplications and is linear in the number of distinct words. The **search** requires a division and additions and is linear in the number of documents.

Security: WWP-I uses the extended IND1-CKA definition from GSW (cf. Section 3.1.2.1). WWP-I is proven secure under the **coDDH** [Boneh and Franklin 2003; Ballard et al. 2005b] assumption and the ***strong*RSA** assumption [Bari and Pfitzmann 1997; Fujisaki and Okamoto 1997; Cramer and Shoup 2000] in the **RO** model in the adapted **IND1-CKA** definition.

4.2.2.3. Dynamic accumulator. Wang et al. [2008a] (WWP-II) propose the first keyword-field-free conjunctive keyword search (KFF-CKS) scheme in the **RO** model. The **idea** is to combine Wang et al.'s dynamic accumulator Wang et al. [2007a]

(membership authentication), Nyberg's combinatorial accumulator Nyberg [1996] (conjunctive keyword search scheme), and Kiayias et al.'s public key encryption [2007] (data cryptosystem) to a trapdoorless and keyword-field-free scheme. WWP-II is trapdoorless in the sense that no public or private key is required to generate a trapdoor for a list of keywords. They construct a specific three-party cryptosystem (TPC), for the security of the data encryption and decryption, using Kiayias et al.'s public key encryption [2007]. The TPC introduces a **third party**, the group manager (GM). The data retrieval is **interactive** like Wang et al.'s RSA accumulator-based scheme [2007b].

Efficiency: **Index generation** uses a hash and a mapping function and is linear in the upper bound on the number of distinct keywords. The **search** is linear in the number of indexes.

Security: WWP-II uses the same IND1-CKA security definition as in WWP-I (cf. Section 4.2.2.2). WWP-II is proven secure in the **RO** model under the Decisional Composite Residuosity **(DCR)** assumption [Paillier 1999] and the extended strong RSA **(esRSA)** assumption [Wang et al. 2007a] in the adapted **IND1-CKA** definition.

See also: Wang et al. [2008b] (Section 3.1.2.5) present a KFF-CKS scheme in the ST model.

4.2.2.4. Bilinear maps. Wang et al. [2008b] (WWP-IIIm) present the first keyword-field-free conjunctive keyword search scheme in the standard model as discussed in Section 3.1.2.5. The **idea** for their multiuser extension for dynamic groups is to use Boneh and Franklin's IBE system [2001, 2003] for data decryption and bilinear maps for user authentication and search. The extension to a dynamic group includes **three parties**: a server, the users (members of a group), and a group manager. The data retrieval is **interactive** like Wang et al.'s RSA accumulator-based scheme Wang et al. [2007b] and dynamic accumulator scheme [2008a].

Efficiency: WWP-IIIm uses symmetric prime order pairings. The **index generation** constructs an l-degree polynomial per document, where l is the number of distinct keywords contained in the document. The algorithm requires l exponentiations per document. A **search** requires a bilinear map per keyword per document index.

Security: WWP-IIIm uses the same IND1-CKA security definition as in WWP-I (cf. Section 4.2.2.2). WWP-IIIm is proven secure under the **DL** and l**-DDHI** assumptions in the adapted **IND1-CKA** definition.

4.2.2.5. Secret sharing. Wang et al. [2008c] (WWP-IV) introduce the notion of threshold privacy-preserving keyword search (TPPKS) and construct the first TPPKS scheme based on Shamir's Secret Sharing [1979] (SSS) and Boneh and Franklin's ID-based cryptosystem [2001, 2003]. Using secret sharing, the **idea** is to allow only collaborating users to search the database. To search, every user generates a share of the trapdoor using his or her own share of the secret. Then, the collaborating users verify their shares, and if the verification was successful, they combine their shares to create the trapdoor for the target keywords. To decrypt, each user generates a decryption share from his or her secret share. If the decryption shares are valid, the users can compute the plaintext. Due to the use of SSS, WWP-IV is **interactive** and works only for a **fixed group** of users, so adding or removing a user is not possible.

Efficiency: WWP-IV uses symmetric prime order pairings for secret share verification. **Index generation** is linear in the number of keywords per document and requires $v + 2$ exponentiations, where v is the number of keywords. The **search** is linear in the number of keywords and indexes.

Security: WWP-IV uses the extended IND1-CKA definition from GSW (cf. Section 3.1.2.1). The secret share verification is secure under the **DL** and the **CDH** assumption. The search process is secure under the **DDH** assumption in the **RO** model in the adapted **IND1-CKA** definition.

4.2.3. Synthesis. Research in the M/M architecture was conducted in the years 2007 and 2008. The schemes focus on single and conjunctive keyword searches. All discussed M/M schemes use PKE in combination with some kind of key distribution or user authentication to allow multiple users to read the encrypted data. All but one scheme introduce a trusted third party (TTP). For example, most of Wang et al.'s schemes discussed in this section are for dynamic groups. These schemes introduce a GM as a trusted third party, which has to re-encrypt the query results to allow the client to decrypt. The advantage of this re-encryption is that revoked users have no access to any of the stored data anymore and that new members have access to all data items, even to those items that were previously stored by other users. Only the HLm scheme does not need a TTP.

Only the WWP-IIIm is proven secure in the standard model. All other schemes use random oracles for their security proofs. Half of the schemes base their security on the RSA assumption or a variant of it. The other half of the schemes use bilinear pairings in their constructions and thus base their security on some kind of DH.

Like the M/S schemes, all of the M/M schemes leak the search pattern and the access pattern. The search pattern is leaked either directly by using a deterministic procedure to generate the trapdoors or indirectly by an offline keyword-guessing attack as discussed in Section 4.1.4. If a TTP is used in the scheme, the attack takes place between the TTP and the storage server.

Table VII gives a detailed overview of the computational complexity and the security of the different algorithms of the discussed schemes. The digest of the table can be found in the reading guidelines in Section 1.5 and the legend in Table VIII.

5. RELATED WORK

In theory, searchable encryption can be achieved by using *oblivious RAMs* (oRAM) [Ostrovsky 1990, 1992; Goldreich and Ostrovsky 1996], which hide all information, including the access pattern, from a remote server. The schemes are not efficient in practice because of a high number of communication rounds and large storage costs on the server side. Therefore, more recent searchable encryption schemes try to achieve more efficient solutions by loosening the security requirements and thus leaking some information (e.g., the access pattern).

The work of Ostrovsky and Skeith [2005, 2007] on *private stream searching* (PSS) and the work of Bethencourt et al. [2006, 2009] are related to searches on encrypted data. It allows a client to send an encrypted search query to an untrusted server, which then uses this query to search in a stream of unencrypted data. The server returns the matching documents without learning anything about the query. PSS can be seen as a generalization of PIR in the sense that more general queries are supported and it is applicable for streaming data.

Agrawal et al. [2004] introduce *order-preserving symmetric encryption* (OPE) for allowing efficient range queries on encrypted data. OPE is a symmetric encryption over the integers such that the numerical orders of plaintexts are preserved in the corresponding ciphertexts. OPE was further studied by Boldyreva et al. [2009, 2011] and Yum et al. [2011].

Chase and Kamara [2010] (CK) propose *structured encryption* (STE), which is a generalization of index-based SSE. STE allows private queries on arbitrarily structured data. The authors give concrete constructions for queries on matrix-structured data,

Table VII. Comparison of Different M/M Schemes. The Legend is in Table VIII

Scheme/Section	Efficiency: Encrypt	Efficiency: Trapdoor	Efficiency: Search	Security: Definition	Security: Assumption	Security: ROM	Notes
Single Keyword Equality Test							
BBO / 4.2.1.1	$nv(\mathsf{E}+\mathsf{h})$	$k\mathsf{h}$	$k\mathsf{s}$	deterministic	SP	✓	deterministic hash
DRD-I / 4.2.1.2	$n(v'+1)\mathsf{e}$	$k\mathsf{e}$	$\mathsf{e}+\mathsf{R}+n\mathsf{c}$	OW	RSA	✓	
DRD-II / 4.2.1.2	$n(4v'+1)\mathsf{e}$	$k\mathsf{e}$	$(1+3v)\mathsf{e}+n\mathsf{c}$	IND-CKA1^t	RSA	✓	
BDD$^+$ / 4.2.1.3	$v'n(2\mathsf{e}+\mathsf{p}^{\mathsf{s}}_{\mathsf{p}})$	$k\mathsf{e}$	$k\mathsf{p}^{\mathsf{s}}_{\mathsf{p}}+n\mathsf{c}$	IND-CKA2	DDH, CDH	✓	interactive encryption
Conjunctive Keyword Equality Test							
HLm / 4.2.2.1	$n(1+u+2v')\mathsf{e}$	$u\mathsf{3e}$	$u3n\mathsf{p}^{\mathsf{s}}_{\mathsf{p}}$	PK-CKA2^t	DLIN	✓	no TTP
WWP-I / 4.2.2.2	$v'n\mathsf{f}$	$k(\mathsf{f}+2\mathsf{e})$	$n\mathsf{e}$	IND1-CKA^t	coDDH, sRSA	✓	
WWP-II / 4.2.2.3	$v''n\mathsf{h}$	$k\mathsf{h}$	$n\mathsf{c}$	IND1-CKA^t	DCR, esRSA	✓	keyword-field free
WWP-IIIm /4.2.2.4	$v'n\mathsf{e}$	$kv'2\mathsf{e}$	$v'n\mathsf{p}^{\mathsf{s}}_{\mathsf{p}}$	IND1-CKA^t	DL, l-DDHI	-	keyword-field free
WWP-IV / 4.2.2	$n(2+v')\mathsf{e}$	$u(k+1)2\mathsf{p}^{\mathsf{s}}_{\mathsf{p}}$	$n(k+1)\mathsf{e}$	IND1-CKA^t	DL, CDH, DDH	✓	fixed user group

^{t} Security definition conceptually as the one stated, but tailored to a specific setting (see respective section).
gray Sublinear search time.

Table VIII. Legend for M/M Schemes in Table VII

Amount		Primitive	
n	number of documents	$\mathsf{p}_\mathsf{p}^\mathsf{s}$	symmetric prime order pairing
v	number of keywords per document	e	exponentiation
v'	number of distinct keywords per document	f	pseudo-random function, permutation or generator
v''	max. number of distinct words	h	hash function
k	number of keywords per trapdoor	c	comparison ($=, \leq, <, ...$)
u	number of users	R	re-encryption

labeled data (cf. Section 3.1.1.7), and (web) graphs, including neighbor, adjacency, and subgraph queries. CK also propose the concept of *controlled disclosure*, which reveals as much information as necessary to compute a function.

Tang [2011, 2012a, 2012b] and Yang et al. [2010] proposed the concept of *public key encryption, which supports equality tests* between ciphertexts (PKEET). PKEET schemes allow equality tests of plaintexts that are encrypted under different public keys.

The notion of *predicate encryption* (PE) was first presented by Katz et al. [2008]. PE allows fine-grained access control on encrypted data. It is a generalized notion/concept of encryption that covers various cryptographic primitives such as identity-based encryption (IBE) [Shamir 1985; Boneh and Franklin 2001, 2003; Cocks 2001], hidden-vector encryption (HVE) [Boyen and Waters 2006; Shi et al. 2007], and attribute-based encryption (ABE) [Sahai and Waters 2005]. PE targets more powerful queries, but the complexity of the query comes at a higher computation cost. In PE, secret keys are associated with predicates and ciphertexts are associated with attributes. A user can decrypt the ciphertext if the private key predicate evaluates to 1 when applied to a ciphertext attribute. PE comes in two versions: (1) with a public index and (2) with attribute hiding. Schemes of (1) are not usable for searchable encryption, because they lack the anonymity property by leaking the set of attributes under which the data is encrypted. The schemes of (2) can be used for SE but are often based on bilinear pairings and are thus less efficient than schemes based on simpler primitives.

Inner-product encryption (IPE) or PE with inner-product relations covers/implies anonymous IBE (AIBE) and hidden-vector encryption (HVE). In IPE, predicates and attributes are represented as vectors. If the inner product of these two vectors is 0, the predicate evaluates to 1 (e.g., attributes correspond to a vector $\vec{x}$, each predicate $f_{\vec{v}}$ corresponds to a vector $\vec{y}$, where $f_{\vec{v}}(\vec{x}) = 1$ iff $\vec{x} \cdot \vec{y} = 0$). The dot product enables more complex evaluations on disjunctions, polynomials, and CNF/DNF formulae. Katz et al. [2008] proposed a system over Z_N. Okamoto and Takashima [2009] and Lewko et al. [2010] gave functions over F_p. Then Okamoto and Takashima [2010, 2012] and Park [2011] proposed more advanced schemes.

Anonymous identity-based encryption (AIBE) in its standard form can support only equality tests and works in the multiple writer/single reader (M/S) scenario. Boneh et al. [2004b] were the first who considered searchable encryption in the asymmetric key setting. Their PEKS scheme was enhanced by Baek et al. [2008] and Rhee et al. [2009]. PEKS has a close connection to AIBE, as pointed out by Boneh et al. [2004b]. Abdalla et al. [2008] formalize AIBE and present a generic SE construction by transforming an anonymous identity-based encryption scheme to a searchable encryption scheme. More

improved IBE schemes that are used for searchable encryption were proposed [Boyen and Waters 2006; Gentry 2006; Kiltz 2007; Nishide et al. 2008]. To allow delegation, *hierarchical* identity-based encryption (HIBE) [Horwitz and Lynn 2002; Gentry and Silverberg 2002; Boneh et al. 2005a] was introduced, where the private keys and ciphertexts are associated with ordered lists of identities. Later, *anonymous* HIBE (AHIBE) schemes [Boyen and Waters 2006; Shi et al. 2007; Shi and Waters 2008; Lee and Lee 2010] were proposed. Abdalla et al. [2008] also gave an AHIBE-to-IBE with keyword search (IBEKS) transformation (hibe-2-ibeks).

Hidden vector encryption (HVE) is a public key encryption scheme that supports wildcard characters inside a key. This allows a variety of application scenarios. Boneh and Waters [2007] proposed the first HVE scheme for searching in encrypted data in 2007. Their scheme allows conjunctive, subset, and range queries. Katz et al. [2008] extended the list with disjunctions, polynomial equations, and inner products. For more information on HVE schemes, we refer the reader to Iovino and Persiano [2008], Shi and Waters [2008], Nishide et al. [2008], Blundo et al. [2009], Sedghi et al. [2010], Park and Lee [2010], Lee and Lee [2011], Caro et al. [2011], Hattori et al. [2011], and Park [2011]. Delegation in PE, more precisely a primitive called delegateable hidden vector encryption (dHVE), was introduced by Shi and Waters [2008]. Iovino and Persiano Iovino and Persiano [2008] provide a solution based on prime order groups, but the scheme works only on binary symbols. HVE can be seen as an extreme generalization of AIBE [Boneh and Waters 2007]. If the HVE is keyword hiding, the transformed PEKS does not leak any information about the keyword used in the Encrypt algorithm.

Homomorphic encryption (HE) is a special type of encryption that allows one to perform an algebraic operation on ciphertexts without decrypting them. This makes HE an interesting tool for searching over encrypted data, since meaningful computation can be executed on the encrypted data. Most HE schemes support either additions [Paillier 1999] or multiplications [ElGamal 1985] on ciphertexts. The pairing-based HE scheme proposed by Boneh et al. [2005b] is able to perform an arbitrary number of additions and one multiplication. Recently, *fully* homomorphic encryption (FHE) was proposed, which can compute arbitrary functions over encrypted data [Gentry 2009, 2010; van Dijk et al. 2010]. It is generally believed that FHE can solve the problem of querying encrypted data, since *any* meaningful computation can be performed on the encrypted data. However, one issue with FHE is the performance, since current schemes are computationally expensive and have a high storage overhead. Since the first FHE scheme, researchers have tried to make the schemes more efficient, but still no practical construction has been proposed [Naehrig et al. 2011]. For some applications, so-called *somewhat* homomorphic encryption schemes can be used. These schemes are more efficient then FHE but allow only a certain amount of additions and multiplications [Gentry et al. 2010; Brakerski and Vaikuntanathan 2011]. The major issue when using somewhat or fully HE as is is that the resulting search schemes require a search time linear in the length of the dataset. This is too slow for practical applications.

6. CONCLUSIONS AND FUTURE WORK

This section gives a summary of our main results, draws conclusions for the theoretically and the practically oriented community, and gives a discussion on directions for future research.

6.1. Summary

Since the early stages of SE, research in the field has been active in all three research directions: improving the query expressiveness, the efficiency, and the security. One can recognize the tradeoffs among these three directions: (1) security versus efficiency,

Table IX. Overview of Known Research in the Field

Architecture	S/S	S/M	M/S	M/M
Equality	✓	✓	✓	✓
Conjunction	✓	-	✓	✓
Comparison	-	-	✓	-
Subset	(✓)	-	✓	-
Range	(✓)	-	✓	-
Wildcard	-	-	✓	-
Similar/Fuzzy	✓	-	-	-
Inner Product	✓	-	(✓)	-
# of schemes	28	2	19	9
# of implementations	6	1	-	2
Timespan	2000–2013	2009–2011	2004–2011	2007–2008

(2) security versys query expressiveness, and (3) efficiency versus query expressiveness. When a scheme tries to be better in one aspect, usually it has to sacrifice another. A good example demonstrating the tradeoff issue, especially for case (1), is using deterministic encryption. Deterministic encryption makes a scheme more efficient but at the same time leaks more information; that is, the ciphertext itself without any trapdoor leaks information (e.g., document/keyword similarity) and directly leaks the search pattern. In the case of public key deterministic encryption using well-known keywords, the server can start a brute force attack by encrypting all possible keywords with the public key and check the encryptions against the ciphertexts.

Table IX gives an overall view of the field of provably secure searchable encryption. The columns of the table represent the different architectures. In the first eight rows, a check mark denotes that there exists a scheme with the specific query expressiveness. A dash indicates that we are not aware of a provably secure SE scheme with the respective expressiveness captured by that row. The ninth row gives the number of schemes per architecture discussed in this article. The number of implemented schemes that we know of is stated in the 10th row. The last row denotes the timespan in which research in the corresponding architecture was conducted.

In total, we analyzed 58 schemes. As indicated in Table IX, most of the SE schemes proposed so far fall either in the S/S or in the M/S architecture. This is due to the use of symmetric or asymmetric encryption primitives, respectively. Although the S/M architecture has not received much research attention in the past, the two existing schemes that fall into this architecture were proposed in 2011. The S/M architecture as the natural extension of the S/S architecture is used for data sharing, where a single writer shares data with several readers. This is a common scenario in practice. Nevertheless, research with respect to this architecture is lean. The same applies to the M/M architecture, which was intensively researched between 2007 and 2008 but seems to be currently out of interest.

Note that an S/S scheme can be trivially constructed from an M/S scheme by keeping the public key of the PKE in use secret. Since the M/S schemes use PKE, it is likely that those schemes are an order of magnitude less efficient than an S/S scheme that is based on symmetric primitives. S/S schemes can also be trivially constructed from S/M schemes.

Only seven papers (ABC$^+$, BTH$^+$, CJJ$^+$, DRD, KPR, PKL$^+$, and RVB$^+$) provide an implementation of the schemes including performance numbers. Most implementations are not publicly available, which makes it hard to compare the schemes on the same hardware with the same dataset. Moreover, it is hard to provide a direct performance comparison since existing protocols for SE address different scenarios and threat models.

6.2. Conclusions

After more than a decade of research, significant progress in the field of provably secure searchable encryption has been made. Research has taken place in all three research directions, mainly focusing on the improvement of query expressiveness and security. In the following, we present our conclusions, classified based on those three research directions.

Query expressiveness. Existing SE schemes already achieve a variety of different search features that allow the deployment of a wide range of applications. Looking at Table IX, we observe a lack of expressiveness in the $*$/M architectures. The widest variety in query expressiveness can be found in the M/S architecture. This is due to the various public key encryption schemes and primitives used in this area. Since there is a wide spectrum of different techniques for PKE, searchable encryption in the M/S architecture can be realized using different primitives (e.g., IBE [Boneh and Boyen 2004], AIBE and HIBE [Abdalla et al. 2008], and HVE [Boneh and Waters 2007]). SE in the M/S setting moves more and more toward (fine-grained) access control (AC). Using AC, a simple search consists of testing whether a certain trapdoor is allowed to decrypt a ciphertext. A successful decryption indicates a match. Since, in general, IPE can be used for SE but no explicit IPE scheme for SE exists, the checkmark in Table IX is in parentheses. Note that inner products can support conjunctive, disjunctive, subset, and range queries, as well as polynomial evaluations and CNF/DNF formulas.

Efficiency. While research in SE continues in all directions, there is an efficiency issue in the multiuser setting that needs to be solved to allow a widespread use of searchable encryption. Efficient schemes with sublinear or optimal search time that achieve IND/PK-CKA2 security exist only in the S/S setting. Current SE schemes in the S/M, M/S, and M/M settings achieving IND/PK-CKA2 security are inefficient (linear in the number of data items/documents) and do not scale well for large datasets, which makes them impractical for real-life use. The deployment of the proposed schemes will cause high query latency and will allow a database server to serve a limited number of clients.

However, two reasons make it more and more urgent to construct practical schemes nowadays, namely, (1) governments forcing organizations to use encryption and (2) the increasing utilization of cloud services:

(i) A number of governmental regulations, such as the Health Insurance Portability and Accountability Act (HIPAA) and the Sarbanes-Oxley Act (SOX), stipulate that organizations have to encrypt their data to prevent it from being disclosed to unauthorized parties. At the same time, organizations are required to search and process their encrypted data in a secure way. Thus, it becomes more and more important to come up with solutions confronting the needs of real-world scenarios.
(ii) In the past, companies relied solely on a protected environment and strong access control for their databases. The current use of cloud services leaves companies to rely solely upon encryption to secure their data and confront threats such as business espionage.

Security. All discussed schemes achieve provable security. SE schemes need to be proven secure in some sense. If a scheme is not secure, encryption is redundant. Nonetheless, even if a scheme is proven secure, it is hard to assess its security, since most schemes are proven secure in different security models and under different computational hardness assumptions. Thus, comparing existing schemes in terms of security is not always possible. Some schemes base their proofs on standard or well-known assumptions. Others come up with novel security assumptions, which makes the

evaluation of the security difficult. Nevertheless, the IND-CKA2 definition gained widespread acceptance as a strong notion of security in the context of SE.

However, using this definition alone is not enough, since the security of SE schemes also needs to take into account the information leakage during or after a search. Recently, Kamara et al. [2012] proposed a framework for describing and comparing the leakage of SE schemes. As a result, the IND-CKA2 definition can be parameterized with additional leakage functions to specify the full information leakage. This is important, since the leaked information can/might be used for statistical analysis (e.g., the search pattern).

Some proposed schemes allow the leakage of the search pattern. Others strive to hide as much information as possible. Revealing the search pattern might not be a problem for certain scenarios, whereas for others it is unacceptable. In a medical database, for example, revealing the search pattern might already leak too much information. This information might be used to correlate it with other (anonymized) public databases. In most situations, it is reasonable that an SE scheme leaks the access pattern. However, for high security applications, the protection of the access pattern might be mandatory. To the best of our knowledge, there is no practical SE scheme that hides the access pattern. However, Boneh et al. [2007] propose a theoretical solution for PKE that allows PIR queries and does not reveal any information with respect to the user's search, not even the access pattern. Their solution is computationally expensive and closer to the concept of PIR than SE, which led us to exclude the aforementioned solution from our analysis.

6.3. Future Work

Future research should focus on improving the query expressiveness and mainly the efficiency/scalability of SE schemes in the S/M, M/S, and M/M settings. Research on query expressiveness needs to move toward closing the gap between existing SE schemes and plaintext searches. This includes but is not limited to functionalities like phrase search, proximity search, or regular expressions. Especially in the $*$/M settings, which represent typical scenarios of data sharing, more query expressiveness is desirable. An interesting research question for $*$/M architectures is whether it is possible to create new schemes that do not rely on a TTP.

An interesting approach for future research is certainly a problem-driven approach; identifying the real-world problems, requirements, and needs first and then trying to address them by means of SE would lead to concrete and useful application scenarios, for example, search in outsourced (personal) databases, secure email routing, search in encrypted emails, and electronic health record (EHR) systems. In order to make an important step toward widespread use of searchable encryption, multiuser schemes need to become more efficient and scalable for large datasets. To assess the real performance of the constructions, more implementations, performance numbers, or at least concrete parameters for the used primitives are necessary.

The main focus of future research in the multiuser setting should be efficiency, since a problem with existing multiuser SE schemes is that they are not practical in real-world applications and do not scale well for large datasets. Only the S/S setting presents reasonable efficient and/or scalable constructions. Consequently, one of the goals of future work should be the reduction of the computational complexity. More efficient provably secure schemes are essential. One possible way to achieve that seems to be the use of different, more efficient primitives or different data representations (e.g., forward index vs. inverted index vs. trees).

Another promising way to address the efficiency/scalability problem might be to explore the possibilities of using two or more collaborating servers to make the search process more efficient. This approach already exists in the context of Secure Multi-Party

Table X. Security Assumptions used in this Article. A * marks assumptions that are generally believed to be well studied

Security Assumption	Reference
Bilinear DH (BDH)*	[Boneh and Franklin 2001; Joux 2002]
Bilinear DH Inversion (BDHI)	[Mitsunari et al. 2002; Boneh and Boyen 2004]
Composite 3-Party DH (C3DH)	[Boneh and Waters 2007]
Computational DH (CDH)*	[Diffie and Hellman 1976]
Co-Diffie-Hellman (coDDH)	[Boneh and Franklin 2003; Ballard et al. 2005b]
External co-DH (coXDH)	[Ryu and Takagi 2007]
Decisional Composite Residuosity (DCR)*	[Paillier 1999]
Decisional DH (DDH)*	[Boneh 1998]
Decisional DH Inversion (DDHI)	[Camenisch et al. 2005]
Discrete Logarithm (DL)*	[Diffie and Hellman 1976]
Decisional Linear DH (DLIN)*	[Boneh et al. 2004a]
Extended Strong RSA (es-RSA)	[Wang et al. 2007a]
Gap DH (GDH)	[Okamoto and Pointcheval 2001]
Modified CDH (mCDH)	[Ibraimi et al. 2011]
Mixed XDH (MXDH)	[Ballard et al. 2005a]
Quadratic Indistinguishability Problem (QIP)	[Goldwasser and Micali 1982] is QRP
RSA (RSA)*	[Rivest et al. 1978]
Strong RSA (s-RSA)*	[Bari and Pfitzmann 1997; Fujisaki and Okamoto 1997] [Cramer and Shoup 2000]
Symmetric External DH (SXDH)	[Ballard et al. 2005a]
External DH (XDH)	[Scott 2002; Boneh et al. 2004a]

Computation [Yao 1982] and Private Information Retrieval [Chor et al. 1995]. Another approach toward improving the efficiency in SE is outsourcing (heavy) computations to third-party entities. One could explore the possibilities of outsourcing (parts of) the computation to (1) dedicated computing utilities and (2) peer-to-peer networks. The field of distributed cryptography could be of help toward this direction. Distributed systems have the additional advantages of autonomy and decentralization, fault tolerance, and scalability. Also, distributed designs can be implemented to be secure against malicious participants and/or allow users to remain anonymous.

In the current research stage, most of the schemes are noninteractive. Deploying interactive protocols can enable the use of simpler primitives and might be computationally more efficient than noninteractive protocols. On the other hand, the communication complexity will most likely increase. This creates a new efficiency tradeoff: computational efficiency versus communication efficiency. However, interactive protocols can achieve a higher level of security [Boneh et al. 2007; Bösch et al. 2012] or efficiency [Cash et al. 2013].

APPENDIX

Table X gives an overview of the security assumptions used by the schemes discussed in this article.

ACKNOWLEDGMENTS

The authors would like to thank Luan Ibraimi, Arjan Jeckmans, Eleftheria Makri, and the anonymous reviewers for their insightful comments.

REFERENCES

Michel Abdalla, Mihir Bellare, Dario Catalano, Eike Kiltz, Tadayoshi Kohno, Tanja Lange, John Malone-Lee, Gregory Neven, Pascal Paillier, and Haixia Shi. 2008. Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions. *Journal of Cryptology* 21, 3 (2008), 350–391.

Masayuki Abe, Rosario Gennaro, Kaoru Kurosawa, and Victor Shoup. 2005. Tag-KEM/DEM: A new framework for hybrid encryption and a new analysis of kurosawa-desmedt KEM. In *EUROCRYPT 2005 (LNCS)*, Vol. 3494. Springer, 128–146.

Michael Adjedj, Julien Bringer, Hervé Chabanne, and Bruno Kindarji. 2009. Biometric identification over encrypted data made feasible. In *ICISS (LNCS)*, Vol. 5905. Springer, 86–100.

Rakesh Agrawal, Jerry Kiernan, Ramakrishnan Srikant, and Yirong Xu. 2004. Order-Preserving encryption for numeric data. In *SIGMOD*. ACM, 563–574.

Georgios Amanatidis, Alexandra Boldyreva, and Adam O'Neill. 2007. Provably-Secure schemes for basic query support in outsourced databases. In *DBSec (LNCS)*, Vol. 4602. Springer, 14–30.

Joonsang Baek, Reihaneh Safavi-Naini, and Willy Susilo. 2006. On the integration of public key data encryption and public key encryption with keyword search. In *ISC (LNCS)*, Vol. 4176. Springer, 217–232.

Joonsang Baek, Reihaneh Safavi-Naini, and Willy Susilo. 2008. Public key encryption with keyword search revisited. In *ICCSA (LNCS)*, Vol. 5072. Springer, 1249–1259.

Lucas Ballard, Matthew Green, Breno de Medeiros, and Fabian Monrose. 2005a. Correlation-Resistant storage via keyword-searchable encryption. *IACR Cryptology ePrint Archive* 2005 (2005), 417.

Lucas Ballard, Seny Kamara, and Fabian Monrose. 2005b. Achieving efficient conjunctive keyword searches over encrypted data. In *ICICS (LNCS)*, Vol. 3783. Springer, 414–426.

Feng Bao, Robert H. Deng, Xuhua Ding, and Yanjiang Yang. 2008. Private query on encrypted data in multi-user settings. In *ISPEC (LNCS)*, Vol. 4991. Springer, 71–85.

Niko Bari and Birgit Pfitzmann. 1997. Collision-Free accumulators and fail-stop signature schemes without trees. In *EUROCRYPT (LNCS)*, Vol. 1233. Springer, 480–494.

Olivier Baudron, David Pointcheval, and Jacques Stern. 2000. Extended notions of security for multicast public key cryptosystems. In *ICALP (LNCS)*, Vol. 1853. Springer, 499–511.

Mihir Bellare, Alexandra Boldyreva, K. Kurosawa, and Jessica Staddon. 2007a. Multirecipient encryption schemes: How to save on bandwidth and computation without sacrificing security. *IEEE Transactions on Information Theory* 53, 11 (2007), 3927–3943.

Mihir Bellare, Alexandra Boldyreva, and Silvio Micali. 2000. Public-Key encryption in a multi-user setting: Security proofs and improvements. In *EUROCRYPT (LNCS)*, Vol. 1807. Springer, 259–274.

Mihir Bellare, Alexandra Boldyreva, and Adam O'Neill. 2007b. Deterministic and efficiently searchable encryption. In *CRYPTO (LNCS)*, Vol. 4622. Springer, 535–552.

Mihir Bellare, Alexandra Boldyreva, and Jessica Staddon. 2003. Randomness re-use in multi-recipient encryption schemeas. In *PKC (LNCS)*, Vol. 2567. Springer, 85–99.

Mihir Bellare and Phillip Rogaway. 1993. Random oracles are practical: A paradigm for designing efficient protocols. In *CCS*. 62–73.

Mihir Bellare and Phillip Rogaway. 1994. Optimal asymmetric encryption. In *EUROCRYPT (LNCS)*, Vol. 950. Springer, 92–111.

Steven M. Bellovin and William R. Cheswick. 2004. Privacy-enhanced searches using encrypted bloom filters. *IACR Cryptology ePrint Archive* 2004 (2004), 22.

Josh Cohen Benaloh and Michael de Mare. 1993. One-Way accumulators: A decentralized alternative to digital sinatures (extended abstract). In *EUROCRYPT (LNCS)*, Vol. 765. 274–285.

John Bethencourt, Dawn Xiaodong Song, and Brent Waters. 2006. New constructions and practical applications for private stream searching (extended abstract). In *S&P*. IEEE Computer Society, 132–139.

John Bethencourt, Dawn Xiaodong Song, and Brent Waters. 2009. New techniques for private stream searching. *ACM Transactions on Information and System Security.* 12, 3 (Jan. 2009), Article 16, 32 pages. DOI:http://dx.doi.org/10.1145/1455526.1455529

Matt Blaze, Gerrit Bleumer, and Martin Strauss. 1998. Divertible protocols and atomic proxy cryptography. In *EUROCRYPT (LNCS)*, Vol. 1403. Springer, 127–144.

Burton H. Bloom. 1970. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM* 13, 7 (1970), 422–426.

Carlo Blundo, Vincenzo Iovino, and Giuseppe Persiano. 2009. Private-key hidden vector encryption with key confidentiality. In *CANS (LNCS)*, Vol. 5888. 259–277.

Alexandra Boldyreva, Nathan Chenette, Younho Lee, and Adam O'Neill. 2009. Order-preserving symmetric encryption. In *EUROCRYPT (LNCS)*, Vol. 5479. Springer, 224–241.

Alexandra Boldyreva, Nathan Chenette, and Adam O'Neill. 2011. Order-preserving encryption revisited: Improved security analysis and alternative solutions. In *CRYPTO (LNCS)*, Vol. 6841. Springer, 578–595.

Dan Boneh. 1998. The decision diffie-hellman problem. In *ANTS (LNCS)*, Vol. 1423. Springer, 48–63.

Dan Boneh and Xavier Boyen. 2004. Efficient selective-ID secure identity-based encryption without random oracles. In *EUROCRYPT (LNCS)*, Vol. 3027. Springer, 223–238.

Dan Boneh, Xavier Boyen, and Eu-Jin Goh. 2005a. Hierarchical identity based encryption with constant size ciphertext. In *EUROCRYPT (LNCS)*, Vol. 3494. Springer, 440–456.

Dan Boneh, Xavier Boyen, and Hovav Shacham. 2004a. Short group signatures. In *CRYPTO (LNCS)*, Vol. 3152. Springer, 41–55.

Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. 2004b. Public key encryption with keyword search. In *EUROCRYPT (LNCS)*, Vol. 3027. 506–522.

Dan Boneh and Matthew K. Franklin. 2001. Identity-based encryption from the weil pairing. In *CRYPTO (LNCS)*, Vol. 2139. Springer, 213–229.

Dan Boneh and Matthew K. Franklin. 2003. Identity-based encryption from the weil pairing. *SIAM J. Comput.* 32, 3 (2003), 586–615.

Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. 2003. Aggregate and verifiably encrypted signatures from bilinear maps. In *EUROCRYPT (LNCS)*, Vol. 2656. Springer, 416–432.

Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. 2005b. Evaluating 2-DNF formulas on ciphertexts. In *TCC (LNCS)*, Vol. 3378. Springer, 325–341.

Dan Boneh, Eyal Kushilevitz, Rafail Ostrovsky, and William E. Skeith III. 2007. Public key encryption that allows PIR queries. In *CRYPTO (LNCS)*, Vol. 4622. Springer, 50–67.

Dan Boneh, Ben Lynn, and Hovav Shacham. 2001. Short signatures from the weil pairing. In *ASIACRYPT (LNCS)*, Vol. 2248. Springer, 514–532.

Dan Boneh and Brent Waters. 2007. Conjunctive, subset, and range queries on encrypted data. In *TCC (LNCS)*, Vol. 4392. Springer, 535–554.

Christoph Bösch, Qiang Tang, Pieter Hartel, and Willem Jonker. 2012. Selective document retrieval from encrypted database. In *ISC (LNCS)*, Vol. 7483. Springer, 224–241.

Xavier Boyen and Brent Waters. 2006. Anonymous hierarchical identity-based encryption (without random oracles). In *CRYPTO (LNCS)*, Vol. 4117. Springer, 290–307.

Zvika Brakerski and Vinod Vaikuntanathan. 2011. Fully homomorphic encryption from ring-LWE and security for key dependent messages. In *CRYPTO (LNCS)*, Vol. 6841. 505–524.

Julien Bringer, Hervé Chabanne, and Bruno Kindarji. 2009. Error-Tolerant searchable encryption. In *ICC*. 1–6.

Richard Brinkman, Ling Feng, Jeroen Doumen, Pieter H. Hartel, and Willem Jonker. 2004. Efficient tree search in encrypted data. *Information Systems Security* 13, 3 (2004), 14–21.

Jin Wook Byun, Dong Hoon Lee, and Jongin Lim. 2006a. Efficient conjunctive keyword search on encrypted data storage system. In *EuroPKI (LNCS)*, Vol. 4043. Springer, 184–196.

Jin Wook Byun, Hyun Suk Rhee, Hyun-A Park, and Dong Hoon Lee. 2006b. Off-Line keyword guessing attacks on recent keyword search schemes over encrypted data. In *SDM (LNCS)*, Vol. 4165. Springer, 75–83.

Jan Camenisch, Susan Hohenberger, and Anna Lysyanskaya. 2005. Compact e-cash. In *EUROCRYPT (LNCS)*, Vol. 3494. Springer, 302–321.

Jan Camenisch and Anna Lysyanskaya. 2002. Dynamic accumulators and application to efficient revocation of anonymous credentials. In *CRYPTO (LNCS)*, Vol. 2442. Springer, 61–76.

Ran Canetti, Shai Halevi, and Jonathan Katz. 2003. A forward-secure public-key encryption scheme. In *EUROCRYPT (LNCS)*, Vol. 2656. Springer, 255–271.

Angelo De Caro, Vincenzo Iovino, and Giuseppe Persiano. 2011. Hidden vector encryption fully secure against unrestricted queries. *IACR Cryptology ePrint Archive* 2011 (2011), 546.

David Cash, Stanislaw Jarecki, Charanjit S. Jutla, Hugo Krawczyk, Marcel Rosu, and Michael Steiner. 2013. Highly-Scalable searchable symmetric encryption with support for Boolean queries. *IACR* 2013 (2013), 169.

Yan-Cheng Chang and Michael Mitzenmacher. 2005. Privacy preserving keyword searches on remote encrypted data. In *(ACNS)*. 442–455.

Melissa Chase and Seny Kamara. 2010. Structured encryption and controlled disclosure. In *ASIACRYPT (LNCS)*, Vol. 6477. Springer, 577–594. Available at `http://dblp.uni-trier.de/db/conf/asiacrypt/asiacrypt2010.html#ChaseK10`.

Sanjit Chatterjee and Alfred Menezes. 2009. On cryptographic protocols employing asymmetric pairings - The role of psi revisited. *IACR Cryptology ePrint Archive* 2009 (2009), 480.

David Chaum. 1982. Blind signatures for untraceable payments. In *CRYPTO*. Plenum Press, New York, 199–203.

Liqun Chen and Zhaohui Cheng. 2005. Security proof of sakai-kasahara's identity-based encryption scheme. In *IMA Int. Conf. (LNCS)*, Vol. 3796. Springer, 442–459.

Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. 1995. Private information retrieval. In *Proceedings of the 36th Annuual IEEE Symposium on Foundations of Computer Science*. 41–50.

Clifford Cocks. 2001. An identity based encryption scheme based on quadratic residues. In *IMA Int. Conf. (LNCS)*, Vol. 2260. Springer, 360–363.

Ronald Cramer and Victor Shoup. 2000. Signature schemes based on the strong RSA assumption. *ACM Trans. Inf. Syst. Secur.* 3, 3 (2000), 161–185.

Giovanni Di Crescenzo and Vishal Saraswat. 2007. Public key encryption with searchable keywords based on jacobi symbols. In *INDOCRYPT (LNCS)*, Vol. 4859. Springer, 282–296.

Reza Curtmola, Juan Garay, Seny Kamara, and Rafail Ostrovsky. 2006. Searchable symmetric encryption: Improved definitions and efficient constructions. In *CCS*. ACM, New York, NY, 79–88. DOI:http://dx.doi.org/10.1145/1180405.1180417

Whitfield Diffie and Martin E. Hellman. 1976. New directions in cryptography. *IEEE Transactions on Information Theory* 22, 6 (November 1976), 644–654.

Changyu Dong, Giovanni Russello, and Naranker Dulay. 2008. Shared and searchable encrypted data for untrusted servers. In *DBSec*. Springer-Verlag, Berlin, Heidelberg, 127–143. DOI:http://dx.doi.org/10.1007/978-3-540-70567-3_10

Taher ElGamal. 1985. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory* 31, 4 (1985), 469–472.

Amos Fiat and Moni Naor. 1994. Broadcast encryption. In *CRYPTO (LNCS)*, Vol. 773. Springer, 480–491.

Michael L. Fredman, János Komlós, and Endre Szemerédi. 1984. Storing a sparse table with 0(1) worst case access time. *J. ACM* 31, 3 (1984), 538–544.

Eiichiro Fujisaki and Tatsuaki Okamoto. 1997. Statistical zero knowledge protocols to prove modular polynomial relations. In *CRYPTO (LNCS)*, Vol. 1294. Springer, 16–30.

Eiichiro Fujisaki, Tatsuaki Okamoto, David Pointcheval, and Jacques Stern. 2001. RSA-OAEP is secure under the RSA assumption. In *CRYPTO (LNCS)*, Vol. 2139. Springer, 260–274.

Craig Gentry. 2006. Practical identity-based encryption without random oracles. In *EUROCRYPT (LNCS)*, Vol. 4004. Springer, 445–464.

Craig Gentry. 2009. Fully homomorphic encryption using ideal lattices. In *STOC*. ACM, 169–178.

Craig Gentry. 2010. Computing arbitrary functions of encrypted data. *Commun. ACM* 53, 3 (2010), 97–105.

Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. 2010. A simple BGN-type cryptosystem from LWE. In *EUROCRYPT (LNCS)*, Vol. 6110. Springer, 506–522.

Craig Gentry and Alice Silverberg. 2002. Hierarchical ID-based cryptography. In *ASIACRYPT (LNCS)*, Vol. 2501. Springer, 548–566.

Eu-Jin Goh. 2003. Secure Indexes. Cryptology ePrint Archive, Report 2003/216. (2003). http://eprint.iacr.org/2003/216/.

Oded Goldreich. 2004. *The Foundations of Cryptography - Volume 2, Basic Applications*. Cambridge University Press.

Oded Goldreich and Rafail Ostrovsky. 1996. Software protection and simulation on oblivious RAMs. *Journal of the ACM* 43, 3 (1996), 431–473.

Shafi Goldwasser and Silvio Micali. 1982. Probabilistic encryption and how to play mental poker keeping secret all partial information. In *STOC*. ACM, 365–377.

Philippe Golle, Jessica Staddon, and Brent Waters. 2004. Secure conjunctive keyword search over encrypted data. In *ACNS*. LNCS 3089, 31–45.

Mitsuhiro Hattori, Takato Hirano, Takashi Ito, Nori Matsuda, Takumi Mori, Yusuke Sakai, and Kazuo Ohta. 2011. Ciphertext-Policy delegatable hidden vector encryption and its application to searchable encryption in multi-user setting. In *IMACC (LNCS)*, Vol. 7089. Springer, 190–209.

Swee-Huay Heng and Kaoru Kurosawa. 2004. k-Resilient identity-based encryption in the standard model. In *CT-RSA (LNCS)*, Vol. 2964. Springer, 67–80.

Swee-Huay Heng and Kaoru Kurosawa. 2006. k-Resilient identity-based encryption in the standard model. *IEICE Transactions* 89-A, 1 (2006), 39–46.

Jeremy Horwitz and Ben Lynn. 2002. Toward hierarchical identity-based encryption. In *EUROCRYPT (LNCS)*, Vol. 2332. Springer, 466–481.

Yong Ho Hwang and Pil Joong Lee. 2007. Public key encryption with conjunctive keyword search and its extension to a multi-user system. In *Pairing (LNCS)*, Vol. 4575. Springer, 2–22.

Luan Ibraimi, Svetla Nikova, Pieter H. Hartel, and Willem Jonker. 2011. Public-key encryption with delegated search. In *ACNS (LNCS)*, Vol. 6715. 532–549.

Vincenzo Iovino and Giuseppe Persiano. 2008. Hidden-vector encryption with groups of prime order. In *Pairing (LNCS)*, Vol. 5209. Springer, 75–88.

Antoine Joux. 2002. The Weil and Tate pairings as building blocks for public key cryptosystems. In *ANTS (LNCS)*, Vol. 2369. Springer, 20–32.

Seny Kamara and Charalampos Papamanthou. 2013. Parallel and dynamic searchable symmetric encryption. In *FC*.

Seny Kamara, Charalampos Papamanthou, and Tom Roeder. 2012. Dynamic searchable symmetric encryption. In *CCS*. ACM, 965–976.

Jonathan Katz, Amit Sahai, and Brent Waters. 2008. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *EUROCRYPT (LNCS)*, Vol. 4965. 146–162.

Dalia Khader. 2007. Public key encryption with keyword search based on k-resilient IBE. In *ICCSA (LNCS)*, Vol. 4707. Springer, 1086–1095.

Aggelos Kiayias, Yiannis Tsiounis, and Moti Yung. 2007. Group encryption. In *ASIACRYPT (LNCS)*, Vol. 4833. Springer, 181–199.

Eike Kiltz. 2007. From Selective-ID to Full Security: The Case of the Inversion-Based Boneh-Boyen IBE Scheme. Cryptology ePrint Archive, Report 2007/033. Available at `http://eprint.iacr.org/`.

Eike Kiltz and David Galindo. 2006. Direct chosen-ciphertext secure identity-based key encapsulation without random oracles. In *ACISP (LNCS)*, Vol. 4058. Springer, 336–347.

Kaoru Kurosawa. 2002. Multi-recipient public-key encryption with shortened ciphertext. In *PKC (LNCS)*, Vol. 2274. Springer, 48–63.

Kaoru Kurosawa and Yvo Desmedt. 2004. A new paradigm of hybrid encryption scheme. In *CRYPTO (LNCS)*, Vol. 3152. Springer, 426–442.

Kaoru Kurosawa and Yasuhiro Ohtaki. 2012. UC-Secure searchable symmetric encryption. In *FC (LNCS)*, Vol. 7397. Springer, 285–298.

Mehmet Kuzu, Mohammad Saiful Islam, and Murat Kantarcioglu. 2012. Efficient similarity search over encrypted data. In *ICDE*. IEEE Computer Society, 1156–1167.

Kwangsu Lee and Dong Hoon Lee. 2010. New techniques for anonymous HIBE with short ciphertexts in prime order groups. *TIIS* 4, 5 (2010), 968–988.

Kwangsu Lee and Dong Hoon Lee. 2011. Improved hidden vector encryption with short ciphertexts and tokens. *Des. Codes Cryptography* 58, 3 (2011), 297–319.

Allison B. Lewko, Tatsuaki Okamoto, Amit Sahai, Katsuyuki Takashima, and Brent Waters. 2010. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In *EUROCRYPT (LNCS)*, Vol. 6110. Springer, 62–91.

Jin Li, Qian Wang, Cong Wang, Ning Cao, Kui Ren, and Wenjing Lou. 2010. Fuzzy keyword search over encrypted data in cloud computing. In *INFOCOM 2010*. IEEE, 441–445.

Qin Liu, Guojun Wang, and Jie Wu. 2009. An efficient privacy preserving keyword search scheme in cloud computing. In *CSE (2)*. IEEE Computer Society, 715–720.

S. Mitsunari, R. Sakai, and M. Kasahara. 2002. A new traitor tracing. *IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences* E85-A (2002), 481–484.

Michael Naehrig, Kristin Lauter, and Vinod Vaikuntanathan. 2011. Can homomorphic encryption be practical? In *CCSW*. ACM, 113–124.

Takashi Nishide, Kazuki Yoneyama, and Kazuo Ohta. 2008. Attribute-based encryption with partially hidden encryptor-specified access structures. In *ACNS (LNCS)*, Vol. 5037. 111–129.

Takashi Nishide, Kazuki Yoneyama, and Kazuo Ohta. 2009. Attribute-based encryption with partially hidden ciphertext policies. *IEICE Transactions* 92-A, 1 (2009), 22–32.

Kaisa Nyberg. 1996. Fast accumulated hashing. In *FSE (LNCS)*, Vol. 1039. 83–87.

Tatsuaki Okamoto and David Pointcheval. 2001. The gap-problems: A new class of problems for the security of cryptographic schemes. In *PKC (LNCS)*, Vol. 1992. Springer, 104–118.

Tatsuaki Okamoto and Katsuyuki Takashima. 2009. Hierarchical predicate encryption for inner-products. In *ASIACRYPT (LNCS)*, Vol. 5912. Springer, 214–231.

Tatsuaki Okamoto and Katsuyuki Takashima. 2010. Fully secure functional encryption with general relations from the decisional linear assumption. In *CRYPTO (LNCS)*, Vol. 6223. 191–208.

Tatsuaki Okamoto and Katsuyuki Takashima. 2012. Adaptively attribute-hiding (hierarchical) inner product encryption. In *EUROCRYPT (LNCS)*, Vol. 7237. 591–608.

Rafail Ostrovsky. 1990. Efficient computation on oblivious RAMs. In *STOC*. ACM, 514–523.

Rafail Ostrovsky. 1992. *Software Protection and Simulations on Oblivious RAMs*. Ph.D. Dissertation. MIT.

Rafail Ostrovsky and William E. Skeith III. 2005. Private searching on streaming data. In *CRYPTO (LNCS)*, Vol. 3621. Springer, 223–240.

Rafail Ostrovsky and William E. Skeith. 2007. Private searching on streaming data. *Journal of Cryptology* 20, 4 (October 2007), 397–430.

Pascal Paillier. 1999. Public-Key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT (LNCS)*, Vol. 1592. Springer, 223–238.

Dong Jin Park, Juyoung Cha, and Pil Joong Lee. 2005. Searchable keyword-based encryption. *IACR Cryptology ePrint Archive* 2005 (2005), 367.

Dong Jin Park, Kihyun Kim, and Pil Joong Lee. 2004. Public key encryption with conjunctive field keyword search. In *WISA (LNCS)*, Vol. 3325. Springer, 73–86.

Hyun-A Park, Bum Han Kim, Dong Hoon Lee, Yon Dohn Chung, and Justin Zhan. 2007. Secure similarity search. In *GrC*. IEEE, 598–604.

Jong Hwan Park. 2011. Inner-product encryption under standard assumptions. *Designs, Codes and Cryptography* 58, 3 (2011), 235–257.

Jong Hwan Park and Dong Hoon Lee. 2010. A hidden vector encryption scheme with constant-size tokens and pairing computations. *IEICE Transactions* 93-A, 9 (2010), 1620–1631.

Raluca A. Popa, Catherine M. S. Redfield, Nickolai Zeldovich, and Hari Balakrishnan. 2011. CryptDB: Protecting confidentiality with encrypted query processing. In *SOSP*. ACM, 85–100.

Mariana Raykova, Binh Vo, Steven M. Bellovin, and Tal Malkin. 2009. Secure anonymous database search. In *CCSW*. ACM, 115–126.

Hyun Sook Rhee, Jong Hwan Park, Willy Susilo, and Dong Hoon Lee. 2009. Improved searchable public key encryption with designated tester. In *ASIACCS*. ACM, 376–379.

Hyun Sook Rhee, Jong Hwan Park, Willy Susilo, and Dong Hoon Lee. 2010. Trapdoor security in a searchable public-key encryption scheme with a designated tester. *Journal of Systems and Software* 83, 5 (2010), 763–771.

Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. 1978. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM* 21, 2 (1978), 120–126.

Eun-Kyung Ryu and Tsuyoshi Takagi. 2007. Efficient conjunctive keyword-searchable encryption. In *AINAW*. IEEE Computer Society, Washington, DC, 409–414. DOI:http://dx.doi.org/10.1109/AINAW.2007.166

Amit Sahai and Brent Waters. 2005. Fuzzy identity-based encryption. In *Proceeding of EUROCRYPT 2005*, Vol. 3494. Springer, 457–473.

Mike Scott. 2002. Authenticated ID-based Key Exchange and remote log-in with simple token and PIN number. *IACR Cryptology ePrint Archive* 2002 (2002), 164.

Saeed Sedghi, Peter van Liesdonk, Svetla Nikova, Pieter H. Hartel, and Willem Jonker. 2010. Searching keywords with wildcards on encrypted data. In *SCN (LNCS)*, Vol. 6280. Springer, 138–153.

Adi Shamir. 1979. How to share a secret. *Commun. ACM* 22, 11 (November 1979), 612–613.

Adi Shamir. 1985. Identity-Based cryptosystems and signature schemes. In *CRYPTO (LNCS)*, Vol. 196. Springer, 47–53.

Emily Shen, Elaine Shi, and Brent Waters. 2009. Predicate privacy in encryption systems. In *TCC (LNCS)*, Vol. 5444. Springer, 457–473.

Elaine Shi, John Bethencourt, Hubert T.-H. Chan, Dawn Xiaodong Song, and Adrian Perrig. 2007. Multi-Dimensional range query over encrypted data. In *S&P*. IEEE Computer Society, 350–364.

Elaine Shi and Brent Waters. 2008. Delegating capabilities in predicate encryption systems. In *ICALP (LNCS)*, Vol. 5126. Springer, 560–578.

V. Shoup. 2004. ISO 18033-2: An Emerging Standard for Public-Key Encryption (committee draft). Available at http://shoup.net/iso/.

Dawn Xiaodong Song, David Wagner, and Adrian Perrig. 2000. Practical techniques for searches on encrypted data. In *IEEE Symposium on Security and Privacy*. IEEE Computer Society, Washington, DC, 44–55.

Qiang Tang. 2011. Towards public key encryption scheme supporting equality test with fine-grained authorization. In *ACISP (LNCS)*, Vol. 6812. Springer, 389–406.

Qiang Tang. 2012a. Public key encryption schemes supporting equality test with authorisation of different granularity. *IJACT* 2, 4 (2012), 304–321.

Qiang Tang. 2012b. Public key encryption supporting plaintext equality test and user-specified authorization. *Security and Communication Networks* 5, 12 (2012), 1351–1362.

Qiang Tang and Liqun Chen. 2009. Public-Key encryption with registered keyword search. In *EuroPKI (LNCS)*, Vol. 6391. Springer, 163–178.

Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. 2010. Fully homomorphic encryption over the integers. In *EUROCRYPT (LNCS)*, Vol. 6110. Springer, 24–43.

Peter van Liesdonk, Saeed Sedghi, Jeroen Doumen, Pieter H. Hartel, and Willem Jonker. 2010. Computationally efficient searchable symmetric encryption. In *SDM (LNCS)*, Vol. 6358. Springer, 87–100.

Peishun Wang, Huaxiong Wang, and Josef Pieprzyk. 2007a. A new dynamic accumulator for batch updates. In *ICICS (LNCS)*, Vol. 4861. Springer, 98–112.

Peishun Wang, Huaxiong Wang, and Josef Pieprzyk. 2007b. Common secure index for conjunctive keyword-based retrieval over encrypted data. In *SDM (LNCS)*, Vol. 4721. Springer, 108–123.

Peishun Wang, Huaxiong Wang, and Josef Pieprzyk. 2008a. An efficient scheme of common secure indices for conjunctive keyword-based retrieval on encrypted data. In *WISA (LNCS)*, Vol. 5379. Springer, 145–159.

Peishun Wang, Huaxiong Wang, and Josef Pieprzyk. 2008b. Keyword field-free conjunctive keyword searches on encrypted data and extension for dynamic groups. In *CANS (LNCS)*, Vol. 5339. Springer, 178–195.

Peishun Wang, Huaxiong Wang, and Josef Pieprzyk. 2008c. Threshold privacy preserving keyword searches. In *SOFSEM (LNCS)*, Vol. 4910. Springer, 646–658.

Guomin Yang, Chik How Tan, Qiong Huang, and Duncan S. Wong. 2010. Probabilistic public key encryption with equality test. In *CT-RSA (LNCS)*, Vol. 5985. Springer, 119–131.

Yanjiang Yang, Feng Bao, Xuhua Ding, and Robert H. Deng. 2009. Multiuser private queries over encrypted databases. *IJACT* 1, 4 (2009), 309–319.

Yanjiang Yang, Haibing Lu, and Jian Weng. 2011. Multi-User private keyword search for cloud computing. In *CloudCom*. IEEE, 264–271.

Andrew Chi-Chih Yao. 1982. Protocols for secure computations (extended abstract). In *IEEE Symposium on Foundations of Computer Science (FOCS '82)*. IEEE Computer Society, 160–164.

Wei-Chuen Yau, Swee-Huay Heng, and Bok-Min Goi. 2008. Off-Line keyword guessing attacks on recent public key encryption with keyword search schemes. In *ATC (LNCS)*, Vol. 5060. Springer, 100–105.

Dae Hyun Yum, Duk Soo Kim, Jin Seok Kim, Pil Joong Lee, and Sung Je Hong. 2011. Order-Preserving encryption for non-uniformly distributed plaintexts. In *WISA (LNCS)*, Vol. 7115. Springer, 84–97.

Rui Zhang and Hideki Imai. 2009. Combining public key encryption with keyword search and public key encryption. *IEICE Transactions* 92-D, 5 (2009), 888–896.

Received December 2012; revised May 2014; accepted June 2014