Ranked Searchable Symmetric Encryption Supporting Conjunctive Queries

Yanjun Shen[1](✉) and Peng Zhang[2]

[1] Faculty of Information Technology, Monash University, Melbourne, VIC 3800, Australia
olivia.shen@monash.edu

[2] College of Information Engineering, Shenzhen University, Shenzhen 518060, China
zhangp@szu.edu.cn

Abstract. Searchable symmetric encryption allows searching over the encrypted data directly without decryption, which is particularly useful in cloud computing as users can outsource the data to cloud servers without privacy leakage while retaining the ability to search. To meet users' specific needs, searching supporting conjunctive queries and ranking is considered. Based on the searchable symmetric encryption protocol OXT of Cash et al., we propose a ranked searchable symmetric encryption scheme by integrating order preserving encryption, so that the matching documents returned to the client are ordered by keyword occurrences. It retains OXT's efficiency and supports ranked search. The security analysis shows that the proposed scheme is secure against chosen plaintext attacks.

Keywords: Searchable symmetric encryption · Conjunctive queries
Order preserving encryption · Ranked search

1 Introduction

With data boosting in modern businesses and trending cloud database solutions, data specialists are constantly facing advanced requirements from aspects including but not limited to data availability and integrity, query processing efficiency, and especially impending security concerns when businesses consider transferring databases and data warehouses from an on-premise server to an untrusted external online server, in other words, outsourcing data storage. Therefore, encryption techniques have never been as challenged as before so as to protect confidential sensitive data from being compromised by malicious or honest-but-curious adversaries.

As an efficient and dominant security technique, data encryption, which encodes data from plaintext into ciphertext that only authorised people with a secret key can read, on the one hand enables parties to communicate messages over a security bridge; however, on the other hand, sacrifices query performance to some extend for security robustness because data has to be decrypted first for further processing.

J. K. Liu and P. Samarati (Eds.): ISPEC 2017, LNCS 10701, pp. 350–360, 2017.
https://doi.org/10.1007/978-3-319-72359-4_20

In order to improve capabilities such like searching or computing over ciphertext to fulfil fundamental business needs, new methods and approaches are proposed to address both data efficiency and security, as a result of which, searchable symmetric encryption (SSE) schemes, firstly introduced by Song et al. [1], have been continuously revised and extended (e.g., [2–5]) to strike a balance between privacy and performance from a technical viewpoint. Meanwhile, with respect to business scenarios, directions for flexible keyword search that supports Boolean queries (both conjunctive and disjunctive), sorting and range queries have been proposed (e.g., [6–10]) to enhance real-world utilisation.

1.1 Our Contributions

In this paper, we follow a searchable symmetric encryption scheme called Oblivious Cross-Tags (OXT) protocol developed by Cash et al. in 2013 [5], and integrate an order preserving encryption scheme into the OXT protocol. As a result, an order preserving searchable encryption scheme in the symmetric key setting is put forward, which allows conjunctive queries over multiple keywords. The proposed scheme returns matching documents ordered by keyword occurrences. In other words, the document with the largest number of occurrences that all keywords appear together is returned first.

As a rigorous extension of Cash et al.'s OXT protocols [5], this proposed scheme retains OXT's performance efficiency, to be precise, optimal server computation and storage as well single round of communication; on the basis of which, it also improves OXT by supporting ranked search to offer a more satisfactory result set.

1.2 The Related Work

In this section, the related studies about searchable symmetric encryption, conjunctive and Boolean queries, and order preserving encryption will be introduced.

Searchable symmetric encryption. Song et al. constructed an SSE solution in 2000 [1] at a price of weakened security guarantee, after which, Goh in 2003 [2] and Chang and Mitzenmacheret in 2005 [3] proposed secure indexes to tackle the above security limitations, whose computation complexity is linear to the number of documents. In 2011, Curtmola et al. [4] provided two solutions whose server computation is linear in the number of documents containing the keyword hence optimal, which was then most secure and efficient and the first sublinear SSE scheme. In addition, they also proposed SSE in the multi-user setting. However, all the above SSE schemes focus on single-keyword search, which results in a large number of matching documents, and a naive solution of conjunctive queries is to run a single-keyword SSE scheme for each keyword thus is not efficient.

Conjunctive and Boolean queries. An SSE solution with supporting for conjunctive and Boolean queries is of practical essence, at least the scheme should be able to support conjunctive searches that return matching documents containing all queried keywords (e.g.,"female" and "master degree"). Conjunctive searches in the symmetric key setting was first considered by Golle, Staddon and Waters in 2004 [6], which is suited for structured data only and leaks the attributes searched, whose complexity is linear in the number of all documents in the database. Ballard et al. in 2005 [7] and Byun et al. in 2006 [8] furthered efficient conjunctive keyword searches, both of which as before apply to structured data only and are linear in the size of the database. Besides, none of the solutions supports disjunctive Boolean queries.

The OXT protocol built by Cash et al. [5] extends to general Boolean queries not only supporting conjunctions but also disjunctions, negations, and more. It also addresses limitations of previous SSE schemes by applying to arbitrarily-structured data, including both attribute-value data and free text. Furthermore, this solution has significant advantages over previous SSE schemes because it scales to large database at a realistic price of allowing leakage of access patterns (but never searched attributes) that is explicitly and precisely defined, whose complexity is linear in the number of documents that contain the least frequent keyword in the query. This protocol can also be extended to support range queries (e.g., "age < 30") and the multi-user setting.

Order preserving encryption (OPE). OPE was first introduced by Agrawal et al. in 2004 [11], which was formally performed and analysed by Boldyreva et al. in 2009 [12] for the first time. Several other OPE schemes were proposed (e.g., [13–15]) but provided no or little security guarantees while leaking most of the plaintext until Popa et al. [16] proposed an ideal-security OPE scheme that satisfies the minimum security requirement, which is to reveal nothing about the plaintext values other than their relative order. OPE enhances searchable encryption to support range queries containing operators such like greater than ($>$) or less than ($<$) in addition to equal to ($=$), meanwhile enables ranked search over encrypted data. Wang et al. [10,17] proposed ranked searchable symmetric encryption schemes adapted from [12], which supports single-keyword search only. Cao et al. [18] defined a multi-keyword ranked search encryption scheme using "coordinate matching" that ranks documents by as many keyword matches as possible which is not ideal considering varied presence frequency and significance of different keywords. Other feasible multi-keyword ranked search encryption schemes (e.g., [19–21]) have been proposed recently using vector space model.

1.3 Organisation

The rest of this paper is organised as follows. Section 2 introduces the preliminaries, including notations, the OXT protocol, and order preserving encryption. Next, our order preserving SSE is defined and proposed in Sect. 3, after which we analyse the security of the proposed algorithm in Sect. 4. At last, this paper is concluded in Sect. 5.

2 Preliminaries

2.1 Notations

In the rest of the sections, some notations are continuously quoted and are essential denoting concepts and terminologies. Following is a list of fundamental notations involved in this paper.

- $[c]$: a set from 1 to c, which is the same as $\{1, 2, ..., c\}$
- $|t|$: the size or length of t
- $t[i]$: the i-th elements of t
- d: the number of documents in a database DB
- ind_i: the identifier of the i-th document
- W_i: a set of keywords in the i-th document
- $\mathsf{W} = \cup_{i=1}^{d} \mathsf{W}_i$: the keyword set of the entire database
- $\mathsf{DB} = (\mathsf{ind}_i, \mathsf{W}_i)_{i=1}^{d}$: a database is parsed as a list of key (document identifier ind_i) - value (corresponding keyword set W_i) pairs
- $\mathsf{DB}(w)$: a set of identifiers of documents that contain the keyword w
- *sterm*: a term (or keyword) estimated to have the least frequency, which is denoted by w_1 for simplicity, that will return a *smallest* $|\mathsf{DB}(w_1)|$
- *xterm*: other terms (or keywords) that are queried, where x stands for *cross*.
- TSet: a tuple set that presents a list of equal-length data tuples with each keyword in the database
- XSet: a set data structure that contains elements computed from each keyword - document pair

2.2 The OXT Protocol

The Oblivious Cross-Tags (OXT) protocol is proposed in the highly-scalable SSE scheme by Cash et al. [5], which consists of an EDB Setup algorithm and a Search protocol. The output from the protocol are encrypted identifiers, which are used by the client to retrieve encrypted documents. The OXT protocol can support Boolean queries of multiple keywords, achieve better security performance by preventing the server from knowing the identifiers of queried *sterm* (the term or keyword that is evaluated to have the least frequency), and reduce the communication between the client and the server to a single round. In order to present our order preserving SSE in an approachable manner, the syntax of OXT is depicted as follows.

In the EDB Setup algorithm described in the following Algorithm 1, function EDBSetup works by taking DB as input and outputting secret keys K along with EDB. K is given to the client, while EDB is given to the server. At this stage, for each keyword in the database, a list of encrypted document pointers is generated, which is then added to an array $\mathbf{T}$ indexed by all keywords of W. During EDBSetup, TSet is initialised taking $\mathbf{T}$ as an output, assembling EDB together with XSet.

Algorithm 1. Cash - EDB Setup Algorithm

```
Input: DB
Output: EDB, $K$
  function EDBSetup(DB)
    Select key $K_S$ for PRF $F$. Select keys $K_X, K_I, K_Z$ for PRF $F_p$ with range $\mathbb{Z}_p^*$.
    $\mathsf{DB} = (\mathsf{ind}_i, \mathsf{W}_i)_{i=1}^d$
    Initialise $\mathbf{T} \leftarrow \{\}$ indexed by $w \in \mathsf{W}$.
    Set $\mathsf{XSet} \leftarrow \emptyset$
    for $w \in \mathsf{W}$ do
      Initialise $\mathbf{t} \leftarrow \{\}$.
      Set $K_e \leftarrow F(K_S, w)$.
      for $\mathsf{ind} \in \mathsf{DB}(w)$ do
        Set a counter $c \leftarrow 0$
        $e \leftarrow \mathsf{Enc}(K_e, \mathsf{ind})$.
        Compute $\mathsf{xind} \leftarrow F_p(K_I, \mathsf{ind})$, $z \leftarrow F_p(K_Z, w||c)$, and $y \leftarrow \mathsf{xind} z^{-1}$.
        Append $(e, y)$ to $\mathbf{t}$.
        Set $\mathsf{xtag} \leftarrow g^{F_p(K_X, w) \cdot \mathsf{xind}}$ and $\mathsf{XSet} \leftarrow \mathsf{XSet} \cup \{\mathsf{xtag}\}$
        Let $c \leftarrow c + 1$.
      end for
      $\mathbf{T}[w] \leftarrow \mathbf{t}$
    end for
    Set $(\mathsf{TSet}, K_T) \leftarrow \mathsf{TSetSetup}(\mathbf{T})$.
    return $\mathsf{EDB} = (\mathsf{TSet}, \mathsf{XSet})$, $K = (K_S, K_X, K_I, K_Z, K_T)$
  end function
```

For ease of understanding, the Search protocol running between the client and the server is split into two parts: Token Generation algorithm and Search algorithm. The Token Generation algorithm described in Algorithm 2 computes and outputs a tag of *sterm* as well as a set of tokens of *xterms* using K and w_1 at the client, and then the Search algorithm described in Algorithm 3 checks existence of *xterms* in the XSet of *sterms* at the server. In other words, the client takes responsibility to choose *sterm* which is basically the least frequent keyword that exists in the smallest number of satisfied documents, and the server returns encrypted identifiers of all satisfied documents to the client, which is then decrypted at the client to retrieve documents, and then stops the client from sending tokens.

Algorithm 2. Cash - Token Generation Algorithm

```
Input: $\bar{w} = (w_1 \wedge \cdots \wedge w_n), K$
Output: stag, xtoken
  function TokenGeneration($\bar{w}, K$)
    Computes $\mathsf{stag} \leftarrow \mathsf{TSetGetTag}(K_T, w_1)$.
    for $c = 1, 2, \ldots$ until stopped by the server do
      for $i = 2, \ldots, n$ do
        $\mathsf{xtoken}[c, i] \leftarrow g^{F_p(K_Z, w_1||c) \cdot F_p(K_X, w_i)}$
      end for
      $\mathsf{xtoken}[c] \leftarrow (\mathsf{xtoken}[c, 2], \ldots, \mathsf{xtoken}[c, n])$
    end for
    return stag, xtoken
  end function
```

Algorithm 3. Cash - Search Algorithm

```
Input: stag, xtoken, EDB
Output: $e$
  function Search(stag, xtoken, EDB)
      Set $\mathbf{t} \leftarrow$ TSetRetrieve(TSet, stag)
      for $c = 1, \ldots, |t|$ do
          Retrieve $(e, y)$ from the $c$-th tuple in $\mathbf{t}$
          if $\forall i = 2, \ldots, n$ : $\mathsf{xtoken}[c, i]^y \in \mathsf{XSet}$ then
              Send $e$ to the client until the last tuple in $\mathbf{t}$ then sends stop
          end if
      end for
  end function
```

Please refer to [5] to obtain more details about the used parameters and T-set implementation $\Sigma = \{\mathsf{TSetSetup}, \mathsf{TSetGetTag}, \mathsf{TSetRetrieve}\}$.

2.3 Order Preserving Encryption

The order preserving encryption scheme $\Pi = \{KeyGen, Enc, Dec\}$ is a deterministic symmetric-key encryption scheme that maintains order relations of plaintexts. A detailed description of each algorithm follows.

$KeyGen(\lambda) \rightarrow s$. This algorithm inputs the security parameter λ and generates a secret key s.

$Enc(s, x) \rightarrow y$. This algorithm computes a ciphertext y for plaintext x based on the secret key s.

$Dec(s, y) \rightarrow x$. This algorithm computes the plaintext x for ciphertext y based on the secret key s.

We say the scheme Π is correct if $\Pi.Dec(s, \Pi.Enc(s, x)) = x$ for any valid s and x. We say it is order preserving if for any valid s, $x < x' \implies \Pi.Enc(s, x) < \Pi.Enc(s.x')$.

So far, the strongest definition of security of order-preserving encryption is indistinguishability under ordered chosen plaintext attack (IND-OCPA), which is achieved by the encryption schemes of [22].

3 Order Preserving SSE Supporting Conjunctive Queries

In this section, the order preserving extension of SSE, which searches the encrypted database based on a conjunctive query and returns satisfied documents ordered by the number of occurrences of all keywords, will be presented and elaborated in detail. This scheme comprises four algorithms as follows, three of which are altered to introduce keyword(s) occurrences.

EDB Setup Algorithm. As described in Algorithm 4, it takes the database DB as an input and outputs the encrypted database EDB along with keys K by the data owner. For each keyword w of interest, an encrypted identifier e of each of all mapped documents, an order preserving encryption result e' of keyword occurrence o in each mapped document, as well as a word-document pointer y that is a function result of the keyword w, each counter c (which is the number of mapped documents), and the corresponding document identifier ind are all stored in $\mathbf{T}[w]$. The array $\mathbf{T}$ indexed by all keywords is then used by TSetSetup as an input to initialise TSet and outputs K_T. Meanwhile, the function result of each keyword w and associated identifiers ind are stored in XSet. The encrypted database EDB = (TSet, XSet) is given to the server, and all keys K are given to the data owner.

Algorithm 4. EDB Setup Algorithm

```
Input: DB
Output: EDB, $K$
  function EDBSetup(DB)
    • Select key $K_S$ for PRF $F$ and $K_{e'}$ for OPE scheme $\Pi$. Select keys $K_X$, $K_I$, $K_Z$ for PRF
  $F_p$ (with range $\mathbb{Z}_p^*$).
    • Initialise $\mathbf{T}$ to an empty array indexed by keywords from W.
    • Initialise XSet ← {}.
    for $w \in$ W do
      Initialise $\mathbf{t} \leftarrow \{\}$; and let $K_e \leftarrow F(K_S, w)$.
      for ind ∈ DB($w$) do
        Initialise a counter $c \leftarrow 0$.
        Set xind ← $F_p(K_I, \mathsf{ind})$, $z \leftarrow F_p(K_Z, w||c)$, $y \leftarrow \mathsf{xind} \cdot z^{-1}$.
        Compute $e \leftarrow \mathsf{Enc}(K_e, \mathsf{ind})$.
        Compute $e' \leftarrow \Pi.Enc(K_{e'}, o)$.
        Append $(y, e, e')$ to $\mathbf{t}$.
        Set xtag ← $g^{F_p(K_X, w)\cdot \mathsf{xind}}$ and append (xtag, $e'$) to XSet.
        $c \leftarrow c + 1$.
      end for
      $\mathbf{T}[w] \leftarrow \mathbf{t}$
    end for
    • (TSet, $K_T$) ← TSetSetup($\mathbf{T}$).
    • return EDB = (Tset, XSet), $K = (K_S, K_{e'}, K_X, K_I, K_Z, K_T)$
  end function
```

Token Generation Algorithm. This algorithm runs at the client side by generating *stag* and *xtoken*, using keywords $\bar{w}$ from a query and keys K from the data owner. *stag* is computed against keyword w_1 that is assumed to result in a relatively small number of mapped documents among all keywords in the query, and *xtoken* is the function result of w_1, counter c, and each other keyword w_i. *stag* and *xtoken* are sent to the server. The details are described in the following Algorithm 5.

Algorithm 5. Token Generation Algorithm

```
Input: $K$, $\bar{w} = (w_1 \wedge \cdots \wedge w_n)$
Output: stag, xtoken
  function TokenGeneration($K$, $\bar{w}$)
    • Computes stag ← TSetGetTag($K_T, w_1$).
    for $c = 1, 2, \ldots$ until the server stops do
      for $i = 2, \ldots, n$ do
        xtoken[$c, i$] ← $g^{F_p(K_Z, w_1||c)\cdot F_p(K_X, w_i)}$
      end for
      Set xtoken[$c$] ← (xtoken[$c, 2$], ..., xtoken[$c, n$])
    end for
    • return stag, xtoken
  end function
```

Search Algorithm. This algorithm described in Algorithm 6 runs at the server, which first retrieves $\mathbf{T}[w_1]$ using *stag* from the client and TSet from the data owner, further retrieves the encrypted identifier e of each document that contains w_1 and the corresponding number of occurrences e'_1. It then uses *xtoken* to verify whether other keywords exist in XSet for each satisfying document and get their encrypted number of occurrences e'_i accordingly. Following that, the smallest of e' denoted as e'_{min} is chosen to be the overall number of occurrences that all keywords appear together. In the end, encrypted identifiers e of all satisfying documents along with their number of occurrences e'_{min} are sent to the client sorted by e'_{min}.

Algorithm 6. Search Algorithm

```
Input: stag, xtoken, EDB
Output: ERS
  function Search(TSet, XSet, stag, xtoken)
      • Initialise ERS ← {}
      • Set $\mathbf{t} = \mathbf{T}[w_1]$ ← TSetRetrieve(TSet, stag)
      for $c = 1, \ldots, |t|$ do
          Retrieve $(y, e, e'_1)$ from the $c$-th tuple in $\mathbf{t}$
          if $\forall i = 2, \ldots, n$ : xtoken$[c, i]^y \in$ XSet then
              Retrieve $e'_2, \ldots, e'_n$.
              $e'_{min} \leftarrow \min(e'_1, e'_2, \ldots, e'_n)$ and append $(e, e'_{min})$ to ERS
          end if
      end for
      • $\forall e$ in ERS, order $e$ by $e'_{min}$.
      • return ERS
  end function
```

Retrieve Algorithm. This algorithm described in Algorithm 7 runs at the client side after receiving ERS from the server, which is decrypted using keys K_S and $K_{e'}$ to get document identifiers and corresponding number of occurrences accordingly. Encrypted documents are retrieved and returned in order (i.e., the document with largest keyword occurrences is returned first).

Algorithm 7. Retrieve Algorithm

```
Input: $K$, ERS
Output: ind, $o$
  function Retrieve(ERS)
      • $K_e \leftarrow F(K_S, w_1)$
      for $(e, e'_{min}) \in$ ERS do
          Compute ind ← Dec$(K_e, e)$, $o \leftarrow \Pi$.Dec$(K_{e'}, e'_{min})$
      end for
      • return (ind, $o$) sorted by $o$.
  end function
```

4 Security Analyses

In this section we describe the scheme leakage profile $\mathcal{L}$ and analyse its security. We use the definition of semantic security for SSE as Cash et al.

Definition. Let $\varUpsilon$= {EDBSetup, TokenGeneration, Search, Retrieve} be an SSE scheme and let $\mathcal{L}$ be an algorithm. For efficient algorithms A and S, we define experiments $Real_A^{\varUpsilon}(\lambda)$ and $Ideal_{A,S}^{\varUpsilon}(\lambda)$ as follows.

$Real_A^{\varUpsilon}(\lambda)$: Chooses DB and a list of queries q. The experiment then runs $(\mathsf{EDB}, K) \leftarrow \mathsf{EDBSetup}(\mathsf{DB})$. For each $i \in |q|$, it runs $\mathsf{TokenGeneration}(K, q[i])$ by the client and $\mathsf{Search}(\mathsf{EDB}, stag, xtoken)$ by the server. Finally the game gives EDB and $\mathbf{t}$ to A, which returns a bit that the game uses as its own output.

$Ideal_{A,S}^{\varUpsilon}(\lambda)$: Chooses DB and a list of queries q. The experiment runs $S(\mathcal{L}(\mathsf{DB}, q))$ and gives its output to A, which returns a bit that the game used as its own output.

We say that is $\mathcal{L}$-semantically-secure against non-adaptive attacks if for all efficient adversaries A there exists an algorithm S such that $\Pr[\mathrm{R}eal_A^{\varUpsilon}(\lambda) = 1] - \Pr[Ideal_{A,S}^{\varUpsilon}(\lambda)] \leq neg(\lambda)$.

Theorem. $\varUpsilon$ is $\mathcal{L}$-semantically-secure against non-adaptive attacks where $\mathcal{L}$ is defined as above, assuming that the DDH assumption holds in G, that F and F_p are secure PRFs, that (Enc, Dec) is an IND-CPA secure symmetric encryption scheme, that $\varSigma$ is a $\mathcal{L}$-secure and computationally correct T-set implementation, and that $\varPi$ is an IND-OCPA secure order preserving encryption scheme.

Proof. This proof is same with the one in [5] except a new game G_N is added after all games G_{Cash}.

Game G_N. During $\mathsf{Initialize}$ mentioned in [5], the ciphertext e' is generated with an encryption of keyword occurrence o. We define there exists an efficient adversary B_N such that

$Pr[G_N = 1] - Pr[G_{Cash} = 1] \leq \mathbf{Adv}_{\varPi, B_N}^{IND-OCPA}(\lambda)$

So we say the proposed $\varUpsilon$ is $\mathcal{L}$-semantically-secure against non-adaptive attacks.

5 Conclusion

Based on the work in [5], by setting keyword-occurrence parameters and integrating an order preserving encryption scheme, an order preserving searchable symmetric encryption supporting conjunctive queries is proposed. Not only all matching documents with keywords are returned to clients, the documents are also ordered by keyword occurrences where the document with the largest number of occurrences is returned first, which may be the most desirable answer for users. The security analysis indicates that the proposed scheme is IND-CPA secure based on the DDH assumptions, secure PRFs, secure symmetric encryption scheme, secure T-set implementation, and secure order preserving encryption scheme.

There are several directions of improvement we can make in the future following the current work. The most promising one is to extend the solution to support Boolean queries, which not only supports conjunctive keyword search but also disjunctions, negations, and more. However, the current relevance score of a matching document is defined as the number of occurrences of the least frequently occurring keyword of all queried keywords contained, which needs further investigation to cooperate with more general Boolean search.

Acknowledgments. This work was supported by the Science and Technology Plan of Shenzhen, China (JCYJ20160307150216309, JCYJ20170302151321095), and Tencent Rhinoceros Birds - Scientific Research Foundation for Young Teachers of Shenzhen University, China.

References

1. Song, D.X., Wagner, D., Perrig, A.: Practical techniques for searches on encrypted data. In: IEEE Symposium on Security & Privacy, pp. 44–55 (2000)
2. Goh, E.: Secure indexes. Cryptology ePrint Archive, Report 2003/216 (2003). http://eprint.iacr.org/2003/216
3. Chang, Y.-C., Mitzenmacher, M.: Privacy preserving keyword searches on remote encrypted data. In: Ioannidis, J., Keromytis, A., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 442–455. Springer, Heidelberg (2005). https://doi.org/10.1007/11496137_30
4. Curtmola, R., Garay, J., Kamara, S., Ostrovsky, R.: Searchable symmetric encryption: improved definitions and efficient constructions. J. Comput. Secur. **19**(5), 895–934 (2011)
5. Cash, D., Jarecki, S., Jutla, C., Krawczyk, H., Roşu, M.-C., Steiner, M.: Highly-scalable searchable symmetric encryption with support for boolean queries. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8042, pp. 353–373. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40041-4_20
6. Golle, P., Staddon, J., Waters, B.: Secure conjunctive keyword search over encrypted data. In: Jakobsson, M., Yung, M., Zhou, J. (eds.) ACNS 2004. LNCS, vol. 3089, pp. 31–45. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24852-1_3
7. Ballard, L., Kamara, S., Monrose, F.: Achieving efficient conjunctive keyword searches over encrypted data. In: Qing, S., Mao, W., López, J., Wang, G. (eds.) ICICS 2005. LNCS, vol. 3783, pp. 414–426. Springer, Heidelberg (2005). https://doi.org/10.1007/11602897_35
8. Byun, J.W., Lee, D.H., Lim, J.: Efficient conjunctive keyword search on encrypted data storage system. In: Atzeni, A.S., Lioy, A. (eds.) EuroPKI 2006. LNCS, vol. 4043, pp. 184–196. Springer, Heidelberg (2006). https://doi.org/10.1007/11774716_15
9. Shi, E., Bethencourt, J., Chan, T.H.H., Song, D., Perrig, A.: Multi-dimensional range query over encrypted data. In: IEEE Symposium on Security & Privacy, pp. 350–364 (2007)
10. Wang, C., Cao, N., Li, J., Ren, K., Lou, W.: Secure ranked keyword search over encrypted cloud data. In: IEEE 30th International Conference on Distributed Computing Systems, pp. 253–262 (2010)
11. Agrawal, R., Kiernan, J., Srikant, R., Xu, Y.: Order preserving encryption for numeric data. In: The 2004 ACM SIGMOD International Conference on Management of Data, pp. 563–574 (2004)

12. Boldyreva, A., Chenette, N., Lee, Y., O'Neill, A.: Order-preserving symmetric encryption. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 224–241. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-01001-9_13
13. Lee, S., Park, T.J., Lee, D., Nam, T., Kim, S.: Chaotic order preserving encryption for efficient and secure queries on databases. IEICE Trans. Inf. Syst. **92**(11), 2207–2217 (2009)
14. Yum, D.H., Kim, D.S., Kim, J.S., Lee, P.J., Hong, S.J.: Order-preserving encryption for non-uniformly distributed plaintexts. In: Jung, S., Yung, M. (eds.) WISA 2011. LNCS, vol. 7115, pp. 84–97. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-27890-7_7
15. Boldyreva, A., Chenette, N., O'Neill, A.: Order-preserving encryption revisited: improved security analysis and alternative solutions. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 578–595. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22792-9_33
16. Popa, R.A., Li, F.H., Zeldovich, N.: An ideal-security protocol for order-preserving encoding. In: IEEE Symposium on Security & Privacy, pp. 463–477 (2013)
17. Wang, C., Cao, N., Ren, K., Lou, W.: Enabling secure and efficient ranked keyword search over outsourced cloud data. IEEE Trans. Parallel Distrib. Syst. **23**(8), 1467–1479 (2012)
18. Cao, N., Wang, C., Li, M., Ren, K., Lou, W.: Privacy-preserving multi-keyword ranked search over encrypted cloud data. IEEE Trans. Parallel Distrib. Syst. **25**(1), 222–233 (2014)
19. Sun, W., Wang, B., Cao, N., Li, M., Lou, W., Hou, Y.T., Li, H.: Privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking. In: Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security, pp. 71–82. ACM (2013)
20. Fu, Z., Sun, X., Liu, Q., Zhou, L., Shu, J.: Achieving efficient cloud search services: multi-keyword ranked search over encrypted cloud data supporting parallel computing. IEICE Trans. Commun. **98**(1), 190–200 (2015)
21. Xia, Z., Wang, X., Sun, X., Wang, Q.: A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data. IEEE Trans. Parallel Distrib. Syst. **27**(2), 340–352 (2016)
22. Kerschbaum, F., Schroepfer, A.: Optimal average-complexity ideal-security order-preserving encryption. In: CCS, pp. 275–286 (2014)