K-word Proximity Search on Encrypted Data

Mark Gall, Gerd Brost
Fraunhofer AISEC
Garching, Germany
e-mail: {gall, brost}@aisec.fraunhofer.de

***Abstract*—Current Symmetric Searchable Encryption schemes do not fulfill the expectations of users who are used to web search engines. Although users are now able to search for multiple keywords, Boolean retrieval returns all results to a client regardless of how relevant the results are for the user. For searches in large data sets when result sets are also expected to be large, Boolean retrieval is not appropriate for users of current information retrieval systems (which are expecting proximity search). In this paper, we present an SSE scheme that allows ranked retrieval on encrypted data. More specifically, we enhanced highly-scalable Boolean retrieval with k-word proximity ranking. Additionally, we introduce an access control in our search engine, such that clients searching the data set will not learn anything about parts of the data set, for which they are not eligible. To achieve this, we rely on attribute-based authentication. The applicability of our scheme was shown in a prototypical implementation.**

***Index Terms*—searchable encryption; attribute-based encryption; proxy re-encryption; proximity search;**

I. Introduction

Searchable symmetric encryption (SSE) is a cryptographic primitive that enables a client to outsource the storage of data to a server, while keeping the data private from the server and maintaining the possibility to selectively search over the data. A client encrypts his data using an encryption algorithm defined in the searchable encryption (SE) scheme and stores the encrypted data on a server. Later, the client can search the encrypted data through interaction with the server following a search protocol defined in the SE scheme. Such schemes have to balance efficiency of the information retrieval process against security in the form of information leakage to the server.

In the past, most of the SSE schemes have addressed single keyword queries in varying degrees of efficiency and data leakage. Only recently schemes with efficient constructions for multi-keyword queries in a Boolean retrieval model have been presented [1], [2]. But even with these advances SSE schemes cannot fulfill the expectations a user has from a modern information retrieval system. Users of web search engines are used to enter multiple words in a search field and retrieve results that not only contain all the keywords, but are ranked according to their relevance. One possibility to rank results that has proven to be useful in web search engines is adjacency and proximity of query terms [3].

The existing SSE schemes that allow information retrieval with multiple keywords employ the *bag of words* model, which ignores the exact ordering of terms in documents. A *document-level* inverted index is used to indicate whether a search term occurs in the document or not. In order to be able to handle phrase and proximity queries a *word-level* index is required that includes word positions of each term in each document.

We present a symmetric searchable encryption scheme that is also based on the *bag of words* model, but uses a *word-level* index with word positions to allow the handling of phrase and proximity queries. Our search scheme expects *free text queries* from the client and handles them as implicit proximity queries, i.e. the scheme searches for documents matching all query terms and ranks the results according to the proximity of the terms in the documents.

Additionally, our setting differs from the standard SSE setting. We expect multiple clients to search the database, not just a single user. Also, we allow the clients to have differing access rights to the data within the database, i.e. not all documents are accessible by the same set of users. A search scheme should reflect this and return only document ids to a client with matching access rights. In our scheme, we enforce access rights with Ciphertext-Policy ABE (CP-ABE) [4], i.e. the documents are encrypted with CP-ABE and stored on a file server. CP-ABE embeds policies in the cipher texts that only allow decryption if the attributes embedded in the private key of a client fulfill the policy. To retrieve search results, the client must present his attribute set to the server, which verifies them. This is achieved by implementing an attribute-based authentication mechanism.

In section II, we define the notation used throughout the paper and describe our setting. In section III we present our construction with all relevant components. A security analysis is given in section IV. We briefly describe our prototypical implementation of our construction in section V and present related work in section VI.

II. Preliminaries

We assume that readers are familiar with pseudorandom functions (PRF) and symmetric encryption schemes with semantic security. The problem of k-word proximity search for ranking documents has been defined by Sadakane et al. [5]. Proximity search ranks results according to the proximity of the occurrences of the search terms in the result documents. They are used to improve search results on large data sets and are widely used in web search engines, but mostly only implicitly. It is possible to allow users to explicitly form proximity search queries, e.g. *employment /3 place* where $/3$ indicates that the *employment* should occur within three words on either side of *place*. The larger the data set and the larger the expected result set of a search, the more important ranking becomes.

A. Notation

Let $\Delta = \{w_1, \dots, w_d\}$ be a dictionary of d words represented as bit strings, i.e. $w_i \in \{1,0\}^*$. Let $\mathsf{DB} \subseteq 2^\Delta$ be a collection of n documents $\mathsf{DB} = (\mathcal{D}_1, \dots, \mathcal{D}_n)$. We denote by $\mathsf{DB}(w)$ the lexicographically ordered list consisting of the identifiers of all documents in DB that contain the word w. The identifier of a document is also represented as a bit string. Let $[n] = \{1, \dots, n\}$. We denote with $\mathbf{v}$ a vector and with $\mathbf{v}[i]$ the i-th component of $\mathbf{v}$. When we get a value from a dictionary D with key k we write $\mathsf{Get}(D, k)$.

Let λ be the security parameter. We denote with $\mathsf{PRF}()$ a standard pseudorandom function and with $\mathsf{PRF}_p()$ a pseudorandom function that maps the result to $\mathbb{Z}_p^*$. $\mathsf{Enc}()$ denotes a semantically secure symmetric encryption scheme. We write $K \overset{R}{\leftarrow} \{0,1\}^\lambda$ to denote a key of length λ sampled uniformly at random as a bit string. For $\mathsf{PRF}_p()$ we sample the key from $\mathbb{Z}_p^*$. A negligible function in λ is denoted by $negl(\lambda)$.

B. Setting

Our setting consists of multiple clients (C), a data owner (D), a trusted ranking server (RS) and an untrusted index server (IS). Clients may only search the data store with permission of the data owner. Note that not all clients share the same permissions, i.e. some clients are allowed to search and access information that is inaccessible for others. After gaining permission from the data owner once, clients may process as many search queries as they want. They send their search queries to the ranking server, who transforms the queries and sends them to the index server. The index server processes the search query and sends the results to the ranking server. The RS filters the results such that only results for which the querying client is eligible are send back to the client and ranks the results before sending them to the client according to the proximity of occurrences of the terms. Note that filtering is for the sake of reducing information leakage to clients. The access control itself still is provided by the ABE-encryption of the documents.

C. Security Games

We recall the security definition for *adaptive $\mathcal{L}$-semantically-security* as defined in [1] and adapt them slightly to match our notation and setting. The definition is a parametrized version of the security definition of *adaptive semantic security for SSE* [6]:

Definition: Let $\Pi = (\mathsf{Init}, \mathsf{Search})$ be a SSE-scheme and let $\mathcal{L}$ be a stateful algorithm. For algorithms A and S, we define games $\mathbf{Real}_A^\Pi(\lambda)$ and $\mathbf{Ideal}_{A,S}^\Pi(\lambda)$ as follows:

$\mathbf{Real}_A^\Pi(\lambda)$: $A(1^\lambda)$ chooses DB. The game then runs $(\mathcal{K}, \mathsf{EDB}, D_F) \leftarrow \mathsf{Init}(\mathsf{DB})$ and gives EDB to A. Then A repeatedly chooses a query q. To respond, the game runs the Search protocol with client input $(K_Q \in \mathcal{K}, q)$. The input for RS is D_F and $K, K_X, K_Z, rk_{D_I \to C} \in \mathcal{K}$ and the input for the IS is EDB. The RS gives the transcript of the protocol run and client output to A. Eventually, A returns a bit that the game uses as its own output.

$\mathbf{Ideal}_{A,S}^\Pi(\lambda)$: The game initializes a counter $i = 0$ and an empty list $\mathbf{q}$. $A(1^\lambda)$ chooses DB. The game then runs $EDB \leftarrow S(\mathcal{L}(DB))$ and gives EDB to A. Then A repeatedly chooses a query q. To respond, the game records this as $\mathbf{q}[i]$, increments i, and gives to A the output of $S(\mathcal{L}(\mathsf{DB}, \mathbf{q}))$. Eventually A returns a bit that the game uses as its own output.

We say that Π is *$\mathcal{L}$-secure against adaptive attacks* if for all PPT adversary A there exists an algorithm S such that

$$Pr[\mathbf{Real}_A^\Pi(\lambda) = 1] - Pr[\mathbf{Ideal}_{A,S}^\Pi(\lambda) = 1] \leq negl(\lambda).$$

III. K-word proximity search on encrypted data

A. Overview

A client C creates a search query with encrypted search terms and sends it to RS. Without learning the plaintext of encrypted terms RS creates the search tokens for a conjunctive search of all search terms. RS interacts with IS to retrieve the results of this search and decrypts the positional information from the results. With the positional information RS ranks the results according to the proximity of the occurring search terms. It also filters those documents from the results for which the client C is not eligible, i.e. the client must have at least the attributes the document has been encrypted for. RS returns the ranked and filtered results to the client, who decrypts the document ids and retrieves the respective documents separately from a file server.

B. Index generation

Handling proximity searches requires a word-level inverted index instead of a document-level index. An inverted index defines document references for each word in the index to allow fast text searches. To create a secure word-level index that can be used for processing proximity searches, we extend the document-level index structure used in the construction by Cash et al. [7] with information for the OXT protocol [1] – so that it can deal with multi-keyword queries – and with positional information. The inverted list postings containing the positional information take the form:

$$< Enc(pk_{D_I}, d), Enc(K_2, f_{d,t} || [P_{0,d,t}, \dots, P_{f_{d,t},d,t}]) >$$

The document id d is encrypted with a different key than the frequency f and the positions $P_{i,d,t}$ of term t in document d, because the RS needs access to the latter information for performing the ranking. The document id d – needed to retrieve the document from the file server storing the documents – is information that the RS should remain oblivious of. We omit how positional information gets extracted from the documents and assume that we have access to the list of keyword positions $P_{w,d} = [P_{0,d,t}, \dots, P_{f_{d,t},d,t}]$ for each term t and each document d when we generate the index.

The OXT protocol operates on an encrypted database EDB that consists of two data structures $TSet$ and $XSet$, i.e. $EDB = (TSet, XSet)$. Both data structures can be realized

```
CreateTSet$(DB, K, K_Q, pk_{D_I})$
1.  Allocate list $L_T$
2.  For each $w \in W$:
3.      $t \leftarrow \mathsf{PRF}(K_Q, w)$
4.      $K_1 \leftarrow \mathsf{PRF}(K, t||1); K_2 \leftarrow \mathsf{PRF}(K, t||2);$
        $K_3 \leftarrow \mathsf{PRF}(K, t||3)$
5.      Initialize counter $c \leftarrow 0$
6.      For each $d \in \mathsf{DB}(w)$:
7.          $l \leftarrow \mathsf{PRF}(K_1, c)$
8.          $a \leftarrow \mathsf{Enc}(K_3, A_d)$
9.          $d' \leftarrow \mathsf{Enc}(pk_{D_I}, d)$
10.         $p \leftarrow \mathsf{Enc}(K_2, c||P_{w,d})$
11.         $x \leftarrow \mathsf{PRF_p}(K_I, d)$
12.         $z \leftarrow \mathsf{PRF_p}(K_Z, t||c)$
13          $y \leftarrow x \cdot z^{-1}$
14.         $c \leftarrow c + 1$
15.         Add $(l, a, d', y, p)$ to $L_T$ (in lexicographic order)
16. Set $D_T \leftarrow \mathbf{Create}(L_T)$
17. Output $D_T$
```

Fig. 1. Generation of TSet for OXT [1] extended with positional information

by dictionaries. The $TSet$ is basically a document-level inverted index for each keyword. The $XSet$ is a construction to enable the efficient search with Boolean queries instead of single-keyword queries. The creation of a $TSet$ extended with positional data is shown in Figure 1. $\mathbf{CreateTSet}()$ takes as input the unencrypted database and three keys K, K_Q, pk_{D_I}. Keys K_1, K_2, K_3 are derived from K to create the pseudorandom label l, to encrypt the positional information $P_{w,d}$ and to encrypt the encoding of the authorized attribute set of document d. This attribute set is derived from the access structure $\mathcal{A}$ of the ABE-encrypted document. When creating an CP-ABE ciphertext, attributes are encoded by their literal description (e.g,. as string values) to a bitstring for efficiency reasons. The key K_Q is used to encrypt the search terms in order to retain a certain level of privacy between the client and the RS. The key pk_{D_I} is used to encrypt the document ids. An entry for the dictionary is comprised of the pseudorandom label l, the encrypted bitstring representing attributes a (which are required to decrypt), the encrypted document id d', the precomputed blinding y from OXT and the encrypted positional information p. Following the approach from Cash et al. [7] we collect all entries in sorted list and create a dictionary from it to achieve history-independence (i.e., the structure of a dictionary does only depend on the elements in the lists, not the order of creation).

The creation of the $XSet$ is shown in Figure 2. $\mathbf{CreateXSet}()$ takes as input the unencrypted database and two keys K and K_Q. Key K_2 is derived from key K to encrypt the positional information. It is important that the cipher texts of the positional information in the $XSet$ and $TSet$ cannot be correlated, otherwise the IS will be able to correlate which terms in the $TSet$ and $XSet$ correspond to each other by using this positional information. To ensure this

```
CreateXSet$(DB, K, K_Q)$
1.  $K_I, K_X, K_Z \overset{R}{\leftarrow} \mathbb{Z}_p^*$
2.  Allocate list $L_X$
3.  For each $w \in W$:
4.      $t \leftarrow \mathsf{PRF}(K_Q, w)$
5.      $K_P \leftarrow \mathsf{PRF}(K, d')$
6.      Initialize counter $c \leftarrow 0$
7.      For each $d \in \mathsf{DB}(w)$:
8.          $p \leftarrow \mathsf{Enc}(K_P, c||P_{w,d})$
9.          $x \leftarrow \mathsf{PRF_p}(K_I, d)$
12.         $c \leftarrow c + 1$
13.         $\mathsf{xt} \leftarrow g^{\mathsf{PRF_p}(K_X, t) \cdot x}$
14.         Add $(\mathsf{xt}, p)$ to $L_X$ (in lexicographic order)
15. Set $D_X \leftarrow \mathbf{Create}(L_X)$
16. Output the keys $K_I, K_X, K_Z$ and $D_X$
```

Fig. 2. Generation of XSet for OXT [1] extended with positional information

we concatenate different counters for the $TSet$ and the $XSet$ to the positional information before encrypting it. The keys K_I, K_X, K_Z for the OXT protocol are generated and used to compute the xtag xt for each entry of the dictionary. The dictionary uses xt as the label and the encrypted positional information p as value. Similar as before we collect all entries in a sorted list and create the dictionary from it to achieve history-independence.

With this preparation the generation of the encrypted positional index shown in Figure 3 works similar to the setup of the encrypted index described by Cash et al. [7]. The key K is generated based on the security parameter λ, the keys pk_{D_I}, sk_{D_I} are generated according to the generation algorithm of the re-encryption scheme [8]. Additionally to the $XSet$ and $TSet$, a dictionary D_F is created that stores the document frequencies of the encrypted search terms. $\mathbf{Init}()$ outputs all created keys as well as the encrypted database EDB and the frequency dictionary D_F.

In our setting the data owner D is supposed to run $\mathbf{Init}()$ and distribute the output as follows: The IS receives EDB, the RS receives D_F and the keys K, K_X, K_Z. The rest of the keys are retained by D.

C. Granting search permission to clients

The first time a client wants to search the data store, it needs to get permission from the data owner. The client creates a key pair (pk_C, sk_C) and sends the data owner his public key pk_C. The data owner sends the key K_Q to the client, creates a re-encryption key $rk_{D_I \to C}$ and sends it to the ranking server. This enables the RS to re-encrypt the document ids such that the client is able to decrypt them. When a client wants to search the data store, it sends search tokens encrypted with K_Q to the ranking server, who processes the search and prepares a list of document ids matching the query as a result. These document ids have been encrypted by the data owner with pk_{D_I} using a uni-directional proxy re-encryption scheme – e.g. one of the schemes presented by Ateniese et al. [8] – and can be re-encrypted by RS using the re-encryption key

```
Init(DB)
1.  $K, K_Q \overset{R}{\leftarrow} \{0,1\}^\lambda$; $K_I, K_X, K_Z \overset{R}{\leftarrow} \mathbb{Z}_p^*$
2.  Generate $(pk_{D_I}, sk_{D_I})$
3.  Allocate list $L_F$
4.  For each $w \in W$:
5.    $t \leftarrow \mathsf{PRF}(K_Q, w)$
6.    Initialize counter $c \leftarrow 0$
7.    For each $d \in \mathsf{DB}(w)$:
8.      $c \leftarrow c + 1$
9.    Add $(t, c)$ to $L_F$
10. Set $D_T \leftarrow \mathbf{CreateTSet}(DB, K, K_Q, pk_{D_I})$
11. Set $D_X \leftarrow \mathbf{CreateXSet}(DB, K, K_Q)$
12. Set $D_F \leftarrow \mathbf{Create}(L_F)$
13. Output keyset $\mathcal{K} = \{K, K_I, K_X, K_Z, K_Q, pk_{D_I}, sk_{D_I}\}$
    and $\mathsf{EDB} = (D_T, D_X)$ and $D_F$
```

Fig. 3. Generation of positional index

$rk_{D_I \to C}$ to a cipher text encrypted with pk_C. The RS sends the re-encrypted ciphertexts to the client, who can decrypt the document ids using his private key sk_C and retrieve the documents from a file server.

D. Attribute based authentication

To minimize information leakage to the client, only document ids of documents that are accessible for the client should be returned so that the client learns nothing about the parts of the database that are inaccessible to him. The client owns a set of attributes that is embedded in his private key. For documents that are accessible to the client, his attribute set fulfills the access policy embedded in the CP-ABE encrypted documents. Thus, the RS needs to verify client attributes, before returning results to a client and check if the client's attributes will fulfill the policy. The RS uses attribute based authentication to achieve this.

CP-ABE offers the following methods:

ABE.$\mathbf{Setup}(\lambda, U)$ that takes a security parameter and the attribute universe to generate public parameters PK and the master key MK.

ABE.$\mathbf{Encrypt}(\mathbf{PK}, \mathbf{M}, \mathcal{A})$ which takes as input the public parameters, the message to encrypted and the access structure $\mathcal{A}$ which defines the policy under which the message is encrypted and generates the cipher text CT.

ABE.$\mathbf{KeyGen}(\mathbf{MK}, \mathbf{S})$ takes the master key and the set of attributes a key should be generated for. It outputs a private key SK.

ABE.$\mathbf{Decrypt}(\mathbf{PK}, \mathbf{CT}, \mathbf{SK})$ is called to decrypt a message. It takes as input the public parameters, a cipher text containing an access structure $\mathcal{A}$ and a private key SK. If the attributes embedded in the private key satisfy $\mathcal{A}$, it outputs the message M.

It is easy to see that the generated MK needs to be kept private by the key server and serves as the *master secret*.

A simple approach to realize attribute based authentication is shown in Figure III-D. The RS can use the CP-ABE scheme

1	C	$\rightarrow$	RS	send attribute set A_C
				calculate $nonce_{ABE(A_C)}$
2	C	$\leftarrow$	RS	send $nonce_{ABE(A_C)}$
3	C	$\rightarrow$	RS	send decrypted $nonce$
				RS compares nonce values
				and verifies A_C

Fig. 4. Access control for ABE tokens; C: Client, RS: Ranking Server, C holds Attribute Set A_C

to verify client attributes. The attributes of the client are embedded in his private key, which it must keep secret. The server encrypts a nonce under the attributes, the client claims to possess. If the client can decrypt the nonce, the client's attributes verified and the client is authenticated under that attribute set.

Once the RS has verified that the client can decrypt a nonce encrypted for the claimed attributes, it can filter the result set for results with a matching access structure and return the filtered results to the client. Since in our ABE scheme every participant can encrypt a cipher text under an attribute set, no extra key-exchange is needed.

E. Searching

When a client wants to search the data store, it initiates the search protocol given in Figure 5. We will discuss important steps of that protocol: The client forms a search query by encrypting all query keywords q_i that it wants to search for separately using the key K_Q. It received K_Q from the data owner, when it was granted permission to search.

The client sends the encrypted query to the RS, who checks if it has a re-encryption key $rk_{D_I \to C}$ for this client. If the data owner has granted permission beforehand to the client, this will be the case. Searching with the OXT protocol is most efficient when choosing the main search term st (which is used to generate the sterm for OXT) to be the one with the lowest document frequency df. Therefore the RS queries the document frequency dictionary D_F with all terms q_i'. Let q_k' be this term (the one with the lowest frequency), then RS uses K to generate the intermediate key $K_1 = PRF(K, q_k||1)$. K_1 is used to create the sterm st. The xtags xt are created per document id. This is in contrast to the original OXT protocol [1], in which only a single st is created together with the same amount of xtags as in our protocol. There the $TSet$ was comprised of lists of document ids that can be retrieved by a single sterm. Depending on the implementing data structure this can leak the document frequency of each term to the server. Following the single keyword approach in [7] this can be avoided and document frequency is only leaked for terms that have been searched for. Our protocol extends this approach to multi-keyword searches with the consequence that df sterms are needed instead of one.

The RS sends the search tokens t to the index server. The index server uses the search tokens to perform a conjunctive search, following our modified OXT protocol. The IS retrieves a data entry from D_T with the sterm st and calculates for each

```
Search
C:  On input $(K_Q,\, q = (q_0, \ldots, q_n))$
    For $1 \leq i \leq n$ :
        $q'_i \leftarrow \mathsf{PRF}(K_Q, q_i)$
    Send $q' = (q'_0, \ldots, q'_n)$ to RS

RS: On input $(D_F, q', K, K_X, K_Z)$
    Set index $k$ of least frequent term $q'_k$ :
        $df \leftarrow \mathsf{Get}(D_F, q'_k) = \min_{1 \leq i \leq n} (\{\mathsf{Get}(D_F, q'_i)\})$
    Set $K_1 \leftarrow \mathsf{PRF}(K, q'_k||1)$
    For $c = 1, \ldots, df$ :
        Set sterm $st[c] \leftarrow \mathsf{PRF}(K_1, c)$
        For $q'_j$ with $1 \leq j \leq n$ and $j \neq k$ :
            $xt[c,j] \leftarrow g^{\mathsf{PRF}_p(K_Z, q'_k||c) \cdot \mathsf{PRF}_p(K_X, q'_j)}$
        Set xterm $xt[c] \leftarrow (xt[c,1], \ldots, xt[c, n-1])$
    Send $t = ((st[1], xt[1]), \ldots, (st[df], xt[df]))$ to IS

IS: On input $(EDB = (D_T, D_X),\, t)$
    Allocate list $L$
    For $c = 1, \ldots, df$ :
        $(d'[c], a[c], y[c], p_T[c]) \leftarrow \mathsf{Get}(D_T, st[c])$
        For $1 \leq i < n$
            Calculate $xtag[c,i] \leftarrow xt[c,i]^{y[c]}$
            $p_X[c,i] \leftarrow \mathsf{Get}(D_X, xtag[c,i])$
        If $p_X[c,i] \neq \perp$ for all $i$ :
            $p_X[c] \leftarrow (p_X[c,1], \ldots, p_X[c, n-1])$
            Add $(d'[c], a[c], p_T[c], p_X[c])$ to list $L$
    Send $L$ to RS

RS: On input $(L, rk_{D_I \to C}, A_C, K)$
    Allocate $n$ lists $L_i$ and list $L_R$
    For each $(d', a, p_T, p_X) \in L$ :
        Add $(d, \mathsf{Dec}(a), \mathsf{Dec}(p_T))$ to list $L_1$
        Add $(d, \mathsf{Dec}(p_X[i]))$ to list $L_i$ for $1 < i \leq n$
    $L_R \leftarrow \mathbf{KWordProximityRanking}(L_i)$
    $L_R \leftarrow \mathbf{FilterAccessRights}(A_C, L_R)$
    $L_R \leftarrow \mathbf{ReEncrypt}(rk_{D_I \to C}, L_R)$
    Output $L_R$
```

Fig. 5. Search protocol

xt the corresponding $xtag$. It tries to retrieve the positional information p from D_X. If D_X returns $\perp$ for any $xtag$ then the data entry is discarded. Otherwise the IS combines the data entry from D_T with the positional information retrieved from D_X and stores it in a result list. After the IS has processed all search tokens it returns the result list to the RS.

The RS uses K_2 to decrypt the positional information of the search results and forms posting lists L_i by splitting the information of the elements of the input list L. The set of attributes – encoded as a bit string – is required to decrypt the document and is added to L_1 only. Thus the postings of L_1 slightly differ from those of L_i for $i > 1$, which take the form:

$$< Enc(pk_{D_I}, d), f_{d,t}||[P_{0,d,t}, \ldots, P_{f_{d,t},d,t}] >$$

At this point the RS has all relevant posting lists for the current search and can then run standard k-word proximity algorithms, such as the plane-sweep algorithm of Sakadane et al. [5] and perform a k-word proximity ranking. After the proximity ranking the RS matches the attributes of the client with the attributes required to decrypt the data and filters out the results for which the client is not eligible. Wether a client is eligible depends on the set of attributes it possesses and which were authenticated before. To match filter by attribute sets, for every item in L_R, $a[c]$ is matched with A_C (also encoded as a bit string) and only valid results are returned. The method $\mathbf{FilterAccessRights}(\mathbf{L_R}, \mathbf{A_C})$ takes the arguments L_R (the result list, which also contains the access structure for each document) and A_C, the attribute set of the client. Only results matching A_C are returned.

Finally, the RS uses the re-encryption key $rk_{D_I \to C}$ to re-encrypt the document ids d' in the result list L_R to d'', so that d'' is a cipher text that can be decrypted with the private key sk_C of the client and outputs L_R.

IV. Security Analysis

In this section, we will describe the leakage function and use this construct to analyze what information can be learned by and adversary.

The first step in the security analysis of our protocol is the definition of the leakage function $\mathcal{L}$ that parametrizes the security definition. We note that for the analysis only the leakage towards the IS is important. The RS is a trusted server in terms of SE. This does not affect cryptographic access control of the ABE-encrypted documents, however. The data owner takes no part in the search protocol, only in the initialization (i.e. the creation of the encrypted database). In our setting clients have differing access rights to the data set, i.e. clients are only allowed to search and access those data items for which they are eligible. As such a client must not learn that there exist documents that match his query, but are not accessible for him. The RS filters all information about such documents from the results, so that no such information is leaked.

As shown in Figure 5 the IS interacts directly only with the RS, when the search protocol is run. From the perspective of the IS, the interaction between the RS and the IS constitutes a conjunctive search according to a modified version of the OXT protocol [1]. This is due to the fact that our construction is based on the basic construction from [7] and uses an adapted version of the OXT protocol for the search. As a consequence, we can infer the leakage function $\mathcal{L}$ of our protocol from the leakage function of the original OXT.

The leakage function $\mathcal{L}$ is a stateful algorithm that describes what an adversary – the IS – is allowed to learn about the data and queries when interacting with a secure protocol. $\mathcal{L}$ gets as initial input DB and outputs $N = \sum_{w \in W} |\mathsf{DB}(w)|$. Thereafter the input for the leakage function is DB and $\mathbf{t}$, where $\mathbf{t}$ consists of all previous queries in addition to the latest query $t = ((st[1], xt[1]), \ldots, (st[df], xt[df]))$ for each search that has been initiated by a client. The output of $\mathcal{L}$ consists of the same components as the leakage function of the original OXT, i.e. $(N, \bar{s}, \mathsf{SP}, \mathsf{XP}, \mathsf{RP}, \mathsf{IP})$. Due to our modifications of the protocol some of components of the output had to be modified as well. For our modified protocol the components of the output for the i-query is:

- $N = \sum_{w \in W} |\mathsf{DB}(w)|$, the total number of entries in the encrypted database.
- $\bar{s} \in [m]^i$: The *equality pattern* $\bar{s}$ includes all previous $i-1$ queries between RS and IS and is formed by assigning each sterm an integer in $[m]$ determined by the order of appearance. Though in our protocol each query t contains multiple sterms, i.e. $st[j]$, each $st[j]$ is a deterministic label and all of them represent the same keyword – just like the single sterm in the original OXT – i.e. the IS will know that a search was for the same keyword whenever a client searches with a same $st[j]$.
- $\mathsf{SP} \in [df_{max}]^i$: Let df_{max} be the maximum document frequency of all terms in the encrypted database. The *size pattern* of the queries is the document frequency df of the first search term, i.e. the sterm which was chosen to be the one with the lowest document frequency. In our scheme this information is obvious to the server because $t = ((st[1], xt[1]), \ldots, (st[df], xt[df]))$
- $\mathsf{XP} \in \mathbb{N}^i$: The vector XP consists of the number of xterms per query
- RP : The *results pattern* contains the results of the queries matching the conjunctions, i.e. for query i it contains $\mathsf{RP}[i] = \bigcup_{i=1}^{df} \mathsf{DB}(st[i]) \cap \mathsf{DB}(xt[i])$
- IP : Formally, the *conditional intersection pattern* for the i-th query is a $i \times i$ table and the entries are defined as:

$$\mathsf{IP}[i,j,k,l] = \begin{cases} \cup_{i=1}^{df_i} \mathsf{DB}(st[i]) \bigcap \cup_{j=1}^{df_j} \mathsf{DB}(st[j]) \\ \qquad \text{if } i \neq j \wedge k \neq l \wedge xt_k[i] \neq xt_l[j] \\ \emptyset \quad \text{otherwise} \end{cases}$$

The parameter N is leaked, since to achieve reasonable performance, data structures are created as dictionaries that will leak the number of entry. This could be reduced by introducing fake entries, but this would require more space on the IS and would still lead to an upper bound, defined by a factor in N.

$\bar{s}$ will leak repetitions in s-terms, since our scheme is deterministic and repeating s-terms can identified. So an adversary is able to create a search history, identifying repeating patterns. As in all variants of the OXT-protocol, multiple T-Sets could be created for a certain keyword and clients could choose one of these instantiations to disguise the query term. For larger data sets, this would mean a considerable increase in database size.

SP is caused by OXT's optimized search, i.e. the separation of $TSet$ and $XSet$. As with the original OXT, artificially extending the size of the $TSet$ in our adapted version could mitigate this by providing upper bounds of results instead of the exact result count. However, storage and communication cost would increase.

RP leaks the number of results of a query. Since we use a deterministic scheme, we cannot disguise the results. However, only repetition patterns can be detected. No semantic information is leaked, since all query terms and all result details are encrypted.

IP is a subtle form of leakage. If two queries share same xterms but have different sterms and the result sets are not disjoint, then information is leaked. The leakage consists of the fact that there is a set of documents matching both sterms. This leakage is also inherited by using OXT for conjunctive searches. Choosing the terms with lowest document frequency somewhat mitigates this leakage, but not entirely.

V. Implementation

We implemented a prototype of the k-word proximity search protocol in Java. The prototype uses AES in CTR mode for the encryption and HMAC for PRFs. The proxy re-encryption is realized with an implementation of the third scheme of Ateniese et el. [8] and for the k-word proximity ranking we use a modified version of the plane-sweep algorithm of Sakadane et al. [5]. For the encryption of the documents, we implemented the CP-ABE scheme of Waters [9]. Both re-encryption and CP-APE are based on pairing based cryptography and we used the Java variant of the Pairing-Based Crypto Library (JPBC) [10] for the implementation.

VI. Related Work

Song et al. proposed the first Searchable Encryption scheme in [11]. This scheme is limited to fixed-size words and cannot be used for more complex data structures than plaintext, since it does not contain an index with a link to arbitrary data items. Goh added a secure index to each document to bypass these limitations in [12]. In [13], Chang and Mitzenmacher present a solution with one index per document with prebuilt dictionaries, causing few communication cost. Curtmola et al. [6] add an inverted index which is generated for each word instead of each document, improving search time since only matching documents affect this. This technique is also good practice in common information retrieval. Van Liesdonk et al. [14] present a scheme with the same search efficiency, but also efficient attribute updates. Chase and Kamara showed a scheme based on Curtmola et al., but with adaptive security in [15], also with optimal search time. This was later extended with dynamic index updates in [16]. A dynamic SE scheme with improved performance, designed for large scale databases was presented by Cash et al. in [7].

The mentioned schemes allow for single keyword searches only. A first scheme supporting conjunctive keyword search is presented by Golle et al. in [17], but this scheme is also causing high communication cost linear to the number of stored items and is based on dedicated keyword fields in the data structure. This is also true for several following schemes. Wang et al. presented in [18] the first conjunctive search scheme without the need for keyword fields. The first search scheme with sublinear search speed and support for – almost – arbitrary Boolean queries was presented by Cash et al. in [1].

Other approaches to the handling arbitrary Boolean queries over encrypted data were presented by Pappas et al. [2] and Moataz et al. [19].

Zittrower et al. [20] proposed a search scheme similar to ours that is capable of handling phrase queries. But their scheme is based on a single keyword searchable encryption scheme and as such leaks to the server much more information than our scheme when searching. Another approach to ranked search over encrypted data was presented by Cao et al. [21]. Their scheme is capable of searching for multiple keywords, but as they generate one index per document their search is less efficient than ours. Also they do not include positional information in their index structure as their ranking only includes the number of occurrences of the keywords.

The term attribute-based encryption (ABE) has first been coined by Sahai and Waters in [22], where they introduce an error tolerance in IBE, allowing a decryption if k of n attributes match in biometrics. This is partially based on the work of Yao et al. in [23], where they show an IBE-system that is able to encrypt to multiple hierarchical identities and already show how to avoid collusion of users. The first separate ABE-scheme for fine-grained access control of cryptographic data was introduced by Goyal, Pandey, Sahai and Waters in [24], where they present a key-policy attribute-based encryption scheme (KP-ABE) that is much more expressive, using trees of Boolean expressions as access structures that are used to generate the key. The first scheme to embed the policy into the cipher text was introduced by Bethencourt, Sahai and Waters in [4]. This scheme was defined under the generic group model and then later transformed by Waters into a scheme under the standard model in [9].

VII. Conclusion and Future Work

We presented a proximity search scheme on encrypted data for multiple clients which supports proximity search over conjunctive keyword queries and incorporates attribute based encryption and authentication. Our scheme tries to move searchable encryption closer to the state of the art in full text search. Since our documents are encrypted not for specific users but for bearers of attribute sets, we incorporated a mechanism for attribute based authentication and filtering results according to attributes. The scheme was implemented in a prototype to show its feasibility.

In our setting, the ranking server is trusted and ranks the results according to the proximity of the search terms in the documents. The ranking server also filters document ids from the result set for which the client does not have the necessary attributes to decrypt the documents.

In general *word-level* indexes are bigger than *document-level* indexes, because they store much more information. To reduce the size of the *word-level* index in the future, we will evaluate the use of more other storage encodings for the positional information, e.g. d-gaps. Also, the current scheme stores the positional information of all documents twice and getting rid of the doubling would result in massive storage gains. The doubling is due to the optimized search of OXT and the fact that one part of the result is taken from the $TSet$, while the other part is taken from the $XSet$. A naive solution of using pointers in both sets to the same positional data item leaks information to the server, it will know which item in the $TSet$ corresponds to which item in the $XSet$ and vice versa.

The retrieval of ABE-encrypted ciphertexts leaks at least the access structure and thus the embedded policy that needs to be fulfilled to decrypt the document. In future work, we will relax the security requirements of RS to enable arbitrary deployment. Currently, whole documents are encrypted via ABE. For future settings, we depend on atomic ABE to efficiently encrypt parts of documents for different attribute sets. We will also examine Predicate Encryption as a way to hide the access policy. Also, to support mobile devices and even less powerful devices, e.g. IoT devices, we will utilize decentralized encryption techniques to save on required computing power.

Acknowledgment

Parts of this work were funded from the ECOSSIAN project. The ECOSSIAN project has received funding from the European Union's Seventh Framework Programme ([FP7/2007-2013]) under grant agreement number SEC-607577.

References

[1] D. Cash, S. Jarecki, C. Jutla, H. Krawczyk, M.-C. Roşu, and M. Steiner, "Highly-scalable searchable symmetric encryption with support for boolean queries," in *Advances in Cryptology–CRYPTO 2013*. Springer, 2013, pp. 353–373.

[2] V. Pappas, F. Krell, B. Vo, V. Kolesnikov, T. Malkin, S. G. Choi, W. George, A. Keromytis, and S. Bellovin, "Blind seer: A scalable private dbms," in *2014 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2014, pp. 359–374.

[3] J. Zobel and A. Moffat, "Inverted files for text search engines," *ACM computing surveys (CSUR)*, vol. 38, no. 2, p. 6, 2006.

[4] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *IEEE Symposium on Security and Privacy, 2007. SP'07.* IEEE, 2007, pp. 321–334.

[5] K. Sadakane, "Fast algorithms for k-word proximity search," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 84, no. 9, pp. 2311–2318, 2001.

[6] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: improved definitions and efficient constructions," in *Proceedings of the 13th ACM conference on Computer and communications security*. ACM, 2006, pp. 79–88.

[7] D. Cash, J. Jaeger, S. Jarecki, C. Jutla, H. Krawczyk, M.-C. Rosu, and M. Steiner, "Dynamic searchable encryption in very-large databases: Data structures and implementation," in *Network and Distributed System Security Symposium (NDSS'14)*, 2014.

[8] G. Ateniese, K. Fu, M. Green, and S. Hohenberger, "Improved proxy re-encryption schemes with applications to secure distributed storage," *ACM Transactions on Information and System Security (TISSEC)*, vol. 9, no. 1, pp. 1–30, 2006.

[9] B. Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization," in *Public Key Cryptography–PKC 2011*. Springer, 2011, pp. 53–70.

[10] A. De Caro and V. Iovino, "jpbc: Java pairing based cryptography," in *Proceedings of the 16th IEEE Symposium on Computers and Communications, ISCC 2011*. Kerkyra, Corfu, Greece, June 28 - July 1: IEEE, 2011, pp. 850–855. [Online]. Available: http://gas.dia.unisa.it/projects/jpbc/

[11] D. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Security and Privacy, 2000. S&P 2000. Proceedings. 2000 IEEE Symposium on*. IEEE, 2000, pp. 44–55.

[12] E.-J. Goh, “Secure indexes,” Cryptology ePrint Archive, Report 2003/216, 2003, http://eprint.iacr.org/2003/216/.

[13] Y. Chang and M. Mitzenmacher, “Privacy preserving keyword searches on remote encrypted data,” in *Applied Cryptography and Network Security*. Springer, 2005, pp. 391–421.

[14] P. Van Liesdonk, S. Sedghi, J. Doumen, P. Hartel, and W. Jonker, “Computationally efficient searchable symmetric encryption,” in *Secure data management*. Springer, 2010, pp. 87–100.

[15] M. Chase and S. Kamara, “Structured encryption and controlled disclosure,” *Advances in Cryptology-ASIACRYPT 2010*, pp. 577–594, 2010.

[16] S. Kamara, C. Papamanthou, and T. Roeder, “Dynamic searchable symmetric encryption,” in *Proceedings of the 2012 ACM conference on Computer and communications security*. ACM, 2012, pp. 965–976.

[17] P. Golle, J. Staddon, and B. Waters, “Secure conjunctive keyword search over encrypted data,” in *Applied Cryptography and Network Security*. Springer, 2004, pp. 31–45.

[18] P. Wang, H. Wang, and J. Pieprzyk, “Keyword field-free conjunctive keyword searches on encrypted data and extension for dynamic groups,” in *Cryptology and Network Security*, ser. Lecture Notes in Computer Science, M. Franklin, L. Hui, and D. Wong, Eds. Springer Berlin Heidelberg, 2008, vol. 5339, pp. 178–195.

[19] T. Moataz and A. Shikfa, “Boolean symmetric searchable encryption,” in *Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security*. ACM, 2013, pp. 265–276.

[20] S. Zittrower and C. C. Zou, “Encrypted phrase searching in the cloud,” in *Global Communications Conference (GLOBECOM), 2012 IEEE*. IEEE, 2012, pp. 764–770.

[21] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, “Privacy-preserving multi-keyword ranked search over encrypted cloud data,” in *INFOCOM, 2011 Proceedings IEEE*. IEEE, 2011, pp. 829–837.

[22] A. Sahai and B. Waters, “Fuzzy identity-based encryption,” *Advances in Cryptology–EUROCRYPT 2005*, pp. 557–557, 2005.

[23] D. Yao, N. Fazio, Y. Dodis, and A. Lysyanskaya, “Id-based encryption for complex hierarchies with applications to forward security and broadcast encryption,” in *Proceedings of the 11th ACM conference on Computer and communications security*. ACM, 2004, pp. 354–363.

[24] V. Goyal, O. Pandey, A. Sahai, and B. Waters, “Attribute-based encryption for fine-grained access control of encrypted data,” in *Proceedings of the 13th ACM conference on Computer and communications security*. ACM, 2006, pp. 89–98.