Boolean Symmetric Searchable Encryption

Tarik Moataz
Telecom Bretagne, Institut Mines-Telecom
Rennes, France
tarik.moataz@telecom-bretagne.eu

Abdullatif Shikfa
Bell Labs Research, Alcatel-Lucent
Nozay, France
abdullatif.shikfa@alcatel-lucent.com

ABSTRACT

In this article we tackle the issue of searchable encryption with a generalized query model. Departing from many previous works that focused on queries consisting of a single keyword, we consider the the case of queries consisting of arbitrary boolean expressions on keywords, that is to say conjunctions and disjunctions of keywords and their complement. Our construction of boolean symmetric searchable encryption BSSE is mainly based on the orthogonalization of the keyword field according to the Gram-Schmidt process. Each document stored in an outsourced server is associated with a label which contains all the keywords corresponding to the document, and searches are performed by way of a simple inner product. Furthermore, the queries in the BSSE scheme are randomized. This randomization hides the search pattern of the user since the search results cannot be associated deterministically to queries. We formally define an adaptive security model for the BSSE scheme. In addition, the search complexity is in $O(n)$ where n is the number of documents stored in the outsourced server.

Categories and Subject Descriptors

[**Security and privacy**]: Cryptography; [**Information systems**]: Data management systems—*Data encryption*; [**Theory of computation**]: Computational complexity and cryptography

General Terms

Security

Keywords

Searchable encryption, Boolean queries, Randomized queries, Outsourced storage

1. INTRODUCTION

Outsourcing is a major trend in computing and it reached its peak with the advent of cloud computing. The issue is that the outsourcing service cannot always be fully trusted. A reasonable assumption is to consider such a server as semi-honest in the sense that it performs operations well but wants to access the data of the user. However, the need to access client's data (e.g., files, email, location) is often at odds with clients' data privacy needs, and in particular the requirement to protect their data (usually through encryption). Reconciling these contradictory requirements, and achieving computation on encrypted data, is an important research and engineering problem.

We focus here on the specific case of indexing and searching on documents. Typically to perform a search on a set of documents, one creates an index of keywords included in the documents and later on performs the search on the index. This does not raise major issues if the querier (the one who requests the search) and the storage provider (the one who stores the documents and the index and who actually performs the search and returns the result) fully trust each other. The difficulties arise when the querier wants to protect the privacy of its data and stores the documents as well as the indexes encrypted, and sends encrypted queries also.

1.1 Problem statement

The problem to solve is therefore to allow performing encrypted searches on a collection of encrypted documents. Solution already exist in the literature and the field of searchable encryption is already mature. However most solutions focus on simple keyword match.

In this article we target a more general model, boolean searchable symmetric encryption, which achieves all basic boolean searches over encrypted data. The basic boolean operations are : the disjunction, the conjunction and the negation. The disjunctive search allows the user to search for encrypted documents containing $word_1$ *or* $word_2$*... or* $word_n$. The conjunctive search allows the user to search for the encrypted documents containing: "$word_1$ *and ... and* $word_n$" and finally the negative search allows the user to search for all encrypted documents which do not contain particular words.

1.2 State of the art

Searchable encryption has been an active research area and many quality works have been published in the beginning of the 2000's, both concerning the symmetric search setting [21, 13, 11, 12, 8, 2] as well as the asymmetric one [5, 4, 10, 1]. Those early schemes focused however only on single keyword query and thus single keyword searches.

Since 2004, several articles focus on extending the search options to enable especially conjunctive search [15, 6, 18,

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
ASIA CCS'13, May 8–10, 2013, Hangzhou, China.
Copyright 2013 ACM 978-1-4503-1767-2/13/05 ...$15.00.

19, 22, 23, 16, 3, 7]. A natural question arises then: why do we need special schemes for performing conjunctive queries? One trivial solution is indeed to perform several queries for one keyword each and then performing set intersection on all the results. However, this simple solution allows the servers to distinguish which documents contain which -encrypted-keyword, and hence leaks information which cannot be tolerated in some cases as mentioned in [15]. Furthermore, this solution comes at the price of higher communication overhead: consider the case where the first query corresponds is matched by 1000 files and the second query by 2000 files while the intersection of the two keywords is only 10 files. In that case the number of transmitted files is several order of magnitudes more than what is required and the situation could be even worse if the query consists on multiple conjunctions. Another solution would be to define a meta-keyword for each possible keywords conjunction; however this solution remains infeasible in practice. Indeed, for each document containing m keywords, the user has to store 2^m other meta-keywords. This solution has thus an exponential storage complexity associated with an increased complexity of search computation on the server.

Golle et al.'s scheme [15] presents the first solution for symmetric conjunctive searchable encryption search. In fact, this scheme is based on bilinear map functions and the Decisional Diffie-Hellman assumption as a security model. We note that there is a similar scheme [18] tackling asymmetric search with similar features as [15]. The main difference between these two schemes besides the fact that they deal with different search setting (symmetric/asymmetric), is that Golle et al. introduce an off-line query. In fact for performance reasons, Golle et al. decided to divide the search query into two parts: an off-line part stored in the outsourced server while storing the whole data, and a common on-line query similar to other schemes. However both these schemes have such a complexity that they cannot be implemented in realistic scenarios due to the exponential storage space that they require. On the other hand, Boneh and Waters [6] present a real improvement in searchable encryption search options. Indeed, they introduce a feasible solution for conjunctive, subset and range queries. They shows that the basic idea is very expensive and needs an exponential data storage, both for the ciphertext or for the trapdoors. Thereby, they present an idea based on Hidden Vector System (HVE) that sharply reduces the ciphetext storage complexity for equality, comparison and subset conjunctions. However, their solution focuses only on conjunctive search queries and did not consider other boolean search options. In addition, Katz et al. [17] present a predicate encryption scheme enabling disjunctive search over encrypted data. Finally, Shi & al. [20] presented an interesting solution to deal with multi-dimensional range query over encrypted data, but as in [6], they only consider conjunctive operations.

To sum up, to the best of our knowledge, there are no previous solutions in the literature supporting boolean searches on encrypted data, i.e. conjunctive, disjunctive and negation search operations at the same time.

1.3 Our contribution

Our construction BSSE is the first scheme that supports generic boolean search over encrypted data. It is based on an original idea of considering keywords as vectors and using the Gram-Schmidt process to orthogonalize and then orthonormalize them. It further makes use of a very efficient operation, the inner product, to perform searches at the server side. The inner product indeed leverages the orthonormalized keywords to efficiently test if a boolean expression query matches the label corresponding to an encrypted document or not. It is an application of mathematical and computing principles to practical security, especially in the searchable encryption scope.

In addition to that, our BSSE scheme presents new features for encrypted documents stored in the server. Indeed, the query sent for retrieving encrypted documents is randomized, which means that the server sees different queries even if the same boolean query is sent several times. Consequently, this feature prevents deterministic association between a query and the corresponding set of documents and makes statistical analysis harder. We introduce the new security notion of *randomized queries* verified by the BSSE scheme. On the other hand, the label that we associate with each encrypted document can be seen as the secure index defined by Goh [13] or the secure scrambled array defined by Curtmola et al. [11]. However, the secure index leaks some meta-information about the number of keywords stored in the Bloom filter, while our secure labels do not leak such information about its content or about the number of keywords that are encrypted before being sent to the server. Our contribution is based on an adversarial model similar to the one defined by Curtmola et al. [11], the differences arise first in the nature of the query sent to the server, and second in the fact that the search pattern of our construction is hidden since we introduce the notion of *randomized query*. Our adversarial model lies in the adaptive setting: this model considers adversaries able to adapt their query in function of previous received queries and search outcomes.

1.4 Outline

The rest of the article is as follows: in section 2 we introduce our notations and some cryptographic reminders while section 3 defines the algorithms of BSSE and the associated security definitions. In section 4 we present the details of our original construction of BSSE and in section 5 we evaluate our construction both from a security and performance perspective. Finally we conclude in section 6.

2. PRELIMINARIES

In this section we start by introducing the notations that will be used in the rest of the article.

2.1 Notations

Let $\mathcal{X}$ be a distribution, we write $x \leftarrow \mathcal{X}$ to denote that x is a sample from the distribution $\mathcal{X}$. $x \xleftarrow{R} \mathcal{S}$ represents a string sampled uniformly at random from the set $\mathcal{S}$. $|x|$ denotes the string length. If x and y are two strings then $x\|y$ designates the string concatenation. If $\mathcal{A}$ is an algorithm then $x \leftarrow \mathcal{A}(...)$ represents the result of applying the algorithm $\mathcal{A}$ to given arguments. k denotes a security parameter. $\delta_{i,j}$ represents the Kronecker symbol such that: if $i = j$ $\delta_{i,j} = 1$ else $\delta_{i,j} = 0$.

Let $D = \{D_1, ..., D_n\}$ be a set of documents, and $W = \{w_1, ..., w_l\}$ be the set of all unique keywords in all documents in D, where $\forall i \in [\![1, l]\!]$ $w_i \in \{0,1\}^*$ (the method for extracting keywords from documents is out of the scope of this work) . We denote the set of positive integers smaller

than l by $I = [\![1, l]\!]$. $M_i \subseteq I$ represents the indexes in W of the keywords corresponding to the document D_i. We consider $M = \{M_i\}_{i \in [\![1,n]\!]}$ the collection of all these index sets.

Boolean expressions: Let $\mathcal{W} = \{w_1, ..., w_r\}$ be a set of keywords. A boolean expression on $\mathcal{W}$ has the form: $\mathcal{B}(\mathcal{W}) = \diamond w_1 \star \diamond w_2 \star ... \star \diamond w_r$, where $\star$ is a binary logical operator $\star \in \{\wedge, \vee\}$ and $\diamond$ is a unitary operator to indicate that the keyword can be complemented ($\diamond \in \{\varnothing, \neg\}$)

All boolean expressions can be expressed in disjunctive normal form and in the following we will only consider this form:

$$\mathcal{B}(\mathcal{W}) = \bigvee_{q=1}^{t} \bigwedge_{j \in I_q} \diamond w_j \ ,$$

where $I_q \subseteq I$ for $1 \leq q \leq t$.

2.2 Cryptographic reminders

In this section we recall some useful and classical definitions in cryptography.

Advantage: Let $\mathcal{A} : \{0,1\}^n \rightarrow \{0,1\}$ be an efficient probabilistic algorithm, and R_1, R_2 two random variables. We define the **distinguishing probability** of $\mathcal{A}$, or the **advantage** of $\mathcal{A}$, for R_1 and R_2 as the ability to distinguish between these two random variables :

$$Adv(\mathcal{A}) = |P_r[\mathcal{A}(R_1) = 1] - P_r[\mathcal{A}(R_2) = 1]|$$

DEFINITION 2.1. ***Symmetric encryption:*** *a symmetric encryption scheme consists of three algorithms $(\mathcal{K}, \mathcal{E}, \mathcal{D})$. The key generation $\mathcal{K}$ takes a security parameter k as input and returns the secret key k_{secret}, we write $k_{secret} \leftarrow \mathcal{K}(k)$. The encryption algorithm $\mathcal{E}$ takes the secret key, the plain text $\mathrm{m} \in \{0,1\}^*$ and a random string r as inputs and returns a ciphertext $\mathcal{C} \in \{0,1\}^*$, we write $\mathcal{C} \leftarrow \mathcal{E}(k_{secret}, m, r)$ or $\mathcal{C} \xleftarrow{R} \mathcal{E}(k_{secret}, m)$. The decryption algorithm $\mathcal{D}$ takes the secret key and the ciphertext as inputs and returns the plaintext m or $\perp$ (in case of error) and we write respectively : $m \leftarrow \mathcal{D}(k_{secret}, \mathcal{E}(k_{secret}, m, r))$ and $\perp \leftarrow \mathcal{D}(k_{secret}, \mathcal{C})$, the symmetric encryption chosen for the following will be semantically secure.*

DEFINITION 2.2. ***Semantic security[14]:*** *a symmetric encryption scheme $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ is semantically secure if for every probabilistic polynomial time algorithm $\mathcal{A}$ there exists a probabilistic polynomial time algorithm $\mathcal{A}'$ such that for every probability ensemble $\{X_k\}_{k \in \mathbb{N}}$ with $|X_k| \leq poly(k)$ where k is the security parameter; every pair of polynomial bounded functions $f, h : \{0,1\}^* \rightarrow \{0,1\}^*$ we have:*

$$\begin{aligned} &\Pr[\mathcal{A}(1^k, \mathcal{E}_K(X_k), h(1^k, X_k) = f(1^k, X_k)] \\ &< \Pr[\mathcal{A}'(1^k, 1^{|X_k|}, h(1^k, X_k) = f(1^k, X_k)] + \epsilon \ , \end{aligned}$$

where $K \leftarrow \mathcal{K}(k)$

The probability in these terms is taken over X_k as well as over the internal coin tosses of either algorithms $\mathcal{K}, \mathcal{E}$ and $\mathcal{A}$ or $\mathcal{A}'$. $\{X_k\}_{k \in \mathbb{N}}$ represents the distribution of plaintexts, f represents the information that the adversary tries to obtain and h represents a priori partial information about the plaintext.

DEFINITION 2.3. ***Pseudo-random functions:*** *Let us consider a function $f : \{0,1\}^s \times \{0,1\}^n \rightarrow \{0,1\}^l$ and the family F of all maps $\{0,1\}^n \rightarrow \{0,1\}^l$. We say that f is a (t, ϵ, q)-pseudo-random function if:*
-f is efficiently computable in polynomial time.
-For any t-time probabilistic polynomial algorithm $\mathcal{A}$:

$$Adv(\mathcal{A}) = |\Pr[\mathcal{A}^{f(k,.)} = 1] - P_r[\mathcal{A}^{r(.)} = 1]| < \epsilon \ ,$$

with $k \xleftarrow{R} \{0,1\}^s$ and $r \xleftarrow{R} F$. When f is bijective (in this case $n = l$) then it is a pseudo-random permutation.

3. BSSE DEFINITIONS

We now focus on definitions specific to our problem. Recall that we are targeting the following scenario: a user wants to store his data D in an outsourced server while protecting data's confidentiality : the documents are thus encrypted. Later on, the user may query the server for some documents. Contrary to previous works we do not target only simple queries consisting of only one keyword, we rather consider an enhanced query model consisting of general boolean expressions on keywords expressed in disjunctive normal form. In the following subsections, we define formally the 4-steps algorithm for encrypting and querying for encrypted data, and then define its security.

3.1 BSSE algorithm definitions

DEFINITION 3.1. *Boolean Symmetric Searchable Encryption (BSSE) over a set of documents D consists of four polynomial-time algorithms BSSE=(Gen,Enc,Query,Test) such that:*

- *$(K_1, K_2) \leftarrow Gen(1^k)$: a key generation algorithm, which takes a security parameter k as input and outputs two secret keys K_1 and K_2.*
- *$(\mathcal{X}, C) \leftarrow Enc(K_1, K_2, W, M, D)$: a probabilistic algorithm, which takes as input the collection of documents D, the keyword set W, the set of indexes M and the secret keys K_1 and K_2, and outputs the collection of encrypted documents $C = \{C_1, ..., C_n\}$ and a set of labels $\mathcal{X} = \{\mathcal{X}_1, ..., \mathcal{X}_n\}$ where each label $\mathcal{X}_i$ is associated with the corresponding encrypted document C_i.*
- *$\mathcal{Q} \leftarrow Query(K_1, \mathcal{B}(\mathcal{W}))$: a probabilistic algorithm, which takes as input a boolean expression $\mathcal{B}(\mathcal{W})$ and the key K_1, and outputs an encoded query.*
- *$\mathcal{L} \leftarrow Test(\mathcal{Q}, \mathcal{X})$: a deterministic algorithm, which takes as input a query $\mathcal{Q}$ and the set of labels $\mathcal{X}$ and outputs the list of ciphertexts $\mathcal{L} \subseteq C$ matching the query.*

3.2 BSSE security definitions

In the following, we define the security requirements of BSSE. We consider the requirements proposed by previous works on symmetric searchable encryption and add two new concepts corresponding to the extended features offered by BSSE.

Before tackling security, our scheme like any cryptographic scheme should be consistent and in our context this means the following:

DEFINITION 3.2. ***Consistency:*** *Let $\mathcal{B}(\mathcal{W})$ be a boolean expression and $\mathcal{L}$ the set of encrypted documents that match*

$\mathcal{B}(\mathcal{W})$. *If $\mathcal{Q}$ is the result of the probabilistic algorithm* ***Query*** *on $\mathcal{B}(\mathcal{W})$, we say that the scheme is consistent if:*

$$Test(\mathcal{Q}, \mathcal{X}) = \mathcal{L}$$

Since the model of adversary is adaptive, we consider security definitions similar to those introduced first by Curtmola et al. in [11] and which take into account the adaptive adversarial model. Some but not all of these definitions are adapted in order to set well for the scheme (we take the same notations and naming as [11]).

DEFINITION 3.3. ***History****: Let D be a set of documents and $\mathsf{W} = \{\mathcal{W}_1, \mathcal{W}_2, ..., \mathcal{W}_q\}$ be a vector of q sets of keywords. A History is a tuple $H = (D, \mathsf{W})$.*

DEFINITION 3.4. ***Access Pattern****: Let D be a set of documents and $\mathsf{W} = \{\mathcal{W}_1, \mathcal{W}_2, ..., \mathcal{W}_q\}$ be a vector of q sets of keywords, and their q corresponding boolean queries $\{\mathcal{B}(\mathcal{W}_1), \mathcal{B}(\mathcal{W}_2), ..., \mathcal{B}(\mathcal{W}_q)\}$. The access pattern of the History $H = (D, \mathsf{W})$ is the tuple*

$$\alpha(H) = (D(\mathcal{B}(\mathcal{W}_1)), D(\mathcal{B}(\mathcal{W}_2)), ..., D(\mathcal{B}(\mathcal{W}_q)))$$

where $D(\mathcal{B}(\mathcal{W}_i))$ represents the set of documents' identifiers which match the boolean query $\mathcal{B}(\mathcal{W}_i)$.

DEFINITION 3.5. ***Search Pattern****: Let D be a set of documents and $\mathsf{W} = \{\mathcal{W}_1, \mathcal{W}_2, ..., \mathcal{W}_q\}$ be a vector of q sets of keywords, and their q corresponding boolean queries $\{\mathcal{B}(\mathcal{W}_1), \mathcal{B}(\mathcal{W}_2), ..., \mathcal{B}(\mathcal{W}_q)\}$. The search pattern of the History $H = (D, \mathsf{W})$ is a binary symmetric matrix $(\sigma(H))_{1 \leq i,j \leq q}$ such that: $\sigma(H)_{i,j} = 1$ if $\mathcal{B}(\mathcal{W}_i) = \mathcal{B}(\mathcal{W}_j)$, and 0 otherwise.*

The *access pattern* represents the results of queries sent to the server, and more specifically the document identifiers corresponding to each query, on the other hand the *search pattern* represents a transcript of queries searched for, the matrix σ records the iteration of the search, a special case of this matrix is the identity matrix if all keyword' sets are different from each others. In addition, the server knows the documents' size. Hence, we will capture all of these information leakage in the *trace* notion.

DEFINITION 3.6. ***Trace****: Let D be a set of documents and $\mathsf{W} = \{\mathcal{W}_1, \mathcal{W}_2, ..., \mathcal{W}_q\}$ be a vector of q sets of keywords, and their q corresponding boolean queries $\{\mathcal{B}(\mathcal{W}_1), \mathcal{B}(\mathcal{W}_2), ..., \mathcal{B}(\mathcal{W}_q)\}$. The trace induced by the History $H = (D, \mathsf{W})$ is the tuple $\gamma(H) = (|D_1|, |D_2|, ..., |D_n|, \alpha(H), \sigma(H))$ where $\alpha(H)$ and $\sigma(H))$ are respictively the access and the search pattern induced by the history H.*

Our adversarial model is based on a simulation-based definition (adversary and simulator), this notion as illustrated in [11], compared to *oracle* access, does not weaken our adversarial definition since we deal with a set of documents instead of individual ones. Adaptive security means that the adversary generates the history adaptively, that means that he takes into account the results of previous queries (search and access pattern) each time before sending the next query from a chosen set of documents generated initially. The only additional requirement is that the view of the adversary (labels, cyphertexts, generated queries) generated from an adversarial and adaptively chosen history should be simulatable given only the trace. We are going to detail the simulation-based experiments in the following.

DEFINITION 3.7. ***Adaptive semantic security*** *: Let $BSSE = (Gen, Enc, Query, Test)$ be the boolean symmetric searchable scheme, k be the security parameter, $\mathcal{A} = (\mathcal{A}_0, ..., \mathcal{A}_q)$ be an adversary such that $q \in \mathbb{N}$ and $\mathcal{S} = (\mathcal{S}_0, ..., \mathcal{S}_q)$ be a simulator, we consider the following probabilistic experiments $\textbf{\textit{Real}}_{BSSE,\mathcal{A}}(k)$ and $\textbf{\textit{Sim}}_{BSSE,\mathcal{A},\mathcal{S}}(k)$:*

$\textbf{\textit{Real}}_{BSSE,\mathcal{A}}(k)$
$(K_1, K_2) \leftarrow Gen(1^k)$
$(D, st_{\mathcal{A}}) \leftarrow \mathcal{A}_0(1^k)$
$(\mathcal{X}, C) \leftarrow Enc(K_1, K_2, W, M, D)$
$(\mathcal{B}(\mathcal{W}_1), st_{\mathcal{A}}) \leftarrow \mathcal{A}_1(st_{\mathcal{A}}, \mathcal{X}, C)$
$\mathcal{Q}_1 \leftarrow Query(K_1, \mathcal{B}(\mathcal{W}_1))$
for $2 \leq i \leq q$:
  $(\mathcal{B}(\mathcal{W}_i), st_{\mathcal{A}}) \leftarrow \mathcal{A}_i(st_{\mathcal{A}}, \mathcal{X}, C)$
  $\mathcal{Q}_i \leftarrow Query(K_1, \mathcal{B}(\mathcal{W}_i))$
let $\mathcal{Q} = \{\mathcal{Q}_1, ..., \mathcal{Q}_q\}$
output $o=(\mathcal{X}, C, \mathcal{Q})$ *and* $st_{\mathcal{A}}$

$\textbf{\textit{Sim}}_{BSSE,\mathcal{A},\mathcal{S}}(k)$
$(D, st_{\mathcal{A}}) \leftarrow \mathcal{A}_0(1^k)$
$(\mathcal{X}, C, st_{\mathcal{S}}) \leftarrow \mathcal{S}_0(\gamma(D))$
$(\mathcal{B}(\mathcal{W}_1), st_{\mathcal{A}}) \leftarrow \mathcal{A}_1(st_{\mathcal{A}}, \mathcal{X}, C)$
$(\mathcal{Q}_1, st_{\mathcal{S}}) \leftarrow \mathcal{S}_1(st_{\mathcal{S}}, \gamma(D, \mathcal{B}(\mathcal{W}_1)))$
for $2 \leq i \leq q$:
  $(\mathcal{B}(\mathcal{W}_i), st_{\mathcal{A}}) \leftarrow \mathcal{A}_i(st_{\mathcal{A}}, \mathcal{X}, C, \mathcal{Q}_1, ..., \mathcal{Q}_{i-1})$
  $(\mathcal{Q}_i, st_{\mathcal{S}}) \leftarrow \mathcal{S}_i(st_{\mathcal{S}}, \gamma(D, \mathcal{B}(\mathcal{W}_1), ..., \mathcal{B}(\mathcal{W}_i)))$
output $o=(\mathcal{X}, C, \mathcal{Q})$ *and* $st_{\mathcal{A}}$

where $st_{\mathcal{A}}$ and $st_{\mathcal{S}}$ are respectively the state of the adversary and the simulator.
We say that BSSE is adaptively semantically secure if for all polynomial time adversaries $\mathcal{A} = (\mathcal{A}_0, ..., \mathcal{A}_q)$ such that q is polynomial in k, there exists a non-uniform polynomial-size simulator $\mathcal{S} = (\mathcal{S}_0, ..., \mathcal{S}_q)$ such that for all polynomial size $\mathcal{H}$ we have:

$$Adv(\mathcal{H}) = \Pr[\mathcal{H}(o, st_{\mathcal{A}}) = 1] - \Pr[\mathcal{H}(o', st'_{\mathcal{A}}) = 1] \leq \epsilon$$

where $(o, st_{\mathcal{A}}) \leftarrow Real_{BSSE,\mathcal{A}}(k)$, $(o', st'_{\mathcal{A}}) \leftarrow Sim_{BSSE,\mathcal{A},\mathcal{S}}(k)$ and the probabilities are over the coins of Gen, Enc and Query.

As defined previously, our scheme introduces a randomized query that enables the user to hide his search pattern. In fact, queries are different even if they are generated for the same boolean query. We formalize this notion as follows:

DEFINITION 3.8. ***Randomized query*** *Let $(\mathcal{Q})_{1 \leq i \leq l}$ be l queries generated with $Query(K_1, .)$ with the same key K_1 and possibly for the same boolean expression $\mathcal{B}(\mathcal{W})$. We say that the scheme has (l, ϵ)-**randomized query** if:*

$$\forall\ i, j \in \{1, l\}\ \ \Pr(\mathcal{Q}_i = \mathcal{Q}_j) < \epsilon$$

This completes our presentation of BSSE security requirement, and we now present our BSSE construction.

4. BSSE CONSTRUCTION

Our overall approach is to consider the keywords as vectors and to pad them so that they form a family of independent vectors. From such a family we construct an orthonormal basis of the resulting vector space, which means that each keyword is associated with one vector of this orthonormal base. Then to each document we associate the vector resulting of the addition of all vectors. The queries are then

also expressed as a vector in the same vector space using the same basis and we roughly use scalar product between the vector of the query and the vector of a document to decide whether a document satisfies a query or not. We thus first present the setup phase of constructing the orthonormal basis and then present the construction of the four algorithm of BSSE.

4.1 BSSE pre-construction

The BSSE algorithm is based on a pre-construction over the unique keywords set $W = \{w_1, ..., w_l\}$ (see section 2 for notations). The aim of this construction is to create from W an orthonormal keyword family. This last is obtained by applying the Gram-Schmidt algorithm to a free family derived from W.

To be more precise, we consider all keywords to be of equal length m (this can be achieved by padding all keywords so that they have the same length).

Let consider a pseudo-random permutation $\pi : \{0,1\}^s \times \{0,1\}^m \to \{0,1\}^m$ and a key K_1 generated from the $Gen(1^k)$ where k is security parameter, for each keyword w_i in W, we apply π_{K_1} in order to create a permuted set of keywords such that $W = \{\pi_{K_1}(w_1), ..., \pi_{K_1}(w_l)\}$.

Let us consider the canonical base of size l $\{e_1, ..., e_l\}$ such that $\forall i \in \{1, l\}$ $|e_i| = l$, and $e_i = (\delta_{i,1}, \ldots, \delta_{i,j}, \ldots, \delta_{i,l})$. In order to create a free family $(f_i)_{1 \leq i \leq l}$ from W, we concatenate each permuted keyword $\pi_{K_1}(w_i)$ with e_i such that $f_i = \pi_{K_1}(w_i) \| e_i$ and $|f_i| = m + l$.

We consider the inner scalar product ϕ :

$$\begin{array}{rccc} \phi & : & \{0,1\}^{m+l} \times \{0,1\}^{m+l} & \to & [\![0, m+l]\!] \\ & & x, y & \mapsto & \sum_{k=1}^{m+l} x_k y_k \end{array}$$

THEOREM GRAM-SCHMIDT [9] 4.1. *If $(f_i)_{i \in \{0,l\}}$ is a free family of a pre-Hilbert space, there exists an orthogonal family $(o_i)_{i \in \{0,l\}}$ such that:*

- $\phi(o_i, o_j) = \phi(o_i, o_i)\delta_{i,j}$,
- $Vect(f_1, ..., f_l) = Vect(o_1, ..., o_l)$,
- *The scalar product $\phi(f_i, o_i)$ is strictly positive.*

DEFINITION 4.1. *The orthogonal projection p on a vector line led by u is defined as:*

$$p_u(v) = \frac{\phi(v,u)}{\phi(u,u)} u$$

Orthogonal Keyword Construction: Let us consider $(f_i)_{1 \leq i \leq l}$ a free family. According to the Theorem 4.1, there is an orthogonal family $(o_i)_{i \in \{0,l\}}$. We can construct this family following the Gram-Schmidt process:

$$o_i = f_i - \sum_{k=1}^{i-1} p_{o_k}(f_i) \qquad for\ 1 \leq i \leq l$$

$(o_i)_{i \in \{0,l\}}$ forms the new keyword orthogonal family that verifies:

$$\forall i, j \in I,\ \phi(o_i, o_j) = \phi(o_i, o_i).\ \delta_{i,j} \tag{1}$$

In addition, we can normalize the vectors to obtain an orthonormal family:

$$\forall k \in I\ u_k = \frac{1}{\sqrt{\phi(o_k, o_k)}} . o_k$$

At the end of this phase, we should point out that each orthonormal keyword u_i corresponds to a keyword w_i. In addition, if we take into account the order of the free family $(f_i)_{1 \leq i \leq l}$ then the construction according to the Gram Schmidt algorithm produces a unique orthonormal family. If we change the order of keywords, the algorithm outputs a different orthonormal family. The number of possible families generated by this construction is $l!$, and the order can hence be considered as a parameter that can enhance security if kept secret.

We should point out that the keyword set W will be stored on the client side in a table $\mathcal{T}$, and that optionally this table could also contain the orthonormal keywords corresponding to them $\{u_k\}_{1 \leq k \leq l}$. Here ends the setup phase, in the following, we will detail the 4-steps BSSE algorithm.

4.2 Construction of BSSE algorithms

We first detail the two steps of the BSSE algorithm namely *Gen* and *Enc* steps generically, and then for the *Query* and *Test* steps, we present them gradually depending on the operations supported.

4.2.1 Key generation step

Gen(k): The user generates randomly two secret keys of size s where s depends on the security parameter k that is to say $(K_1, K_2) \xleftarrow{R} \{0,1\}^{2s}$.

4.2.2 Encryption step

Enc(K_1, K_2, W, M, D): From the entire set of unique keywords $(w_i)_{i \in \{0,l\}}$, according to the BSSE pre-construction we produce the orthonormal keyword family $(u_i)_{i \in \{0,l\}}$. These orthonormal families can either be retrieved or recomputed depending on whether they were stored in table $\mathcal{T}$ or not.

Each document D_i in the set D contains a set M_i of unique keywords $(u_k)_{k \in M_i}$ where M_i is the set of keywords indexes of D_i. The output of this step is the encrypted document $\mathcal{E}_{K_2}(D_i)$, and the associated label $\mathcal{X}_i = \sum_{k \in M_i} u_k$ for each $i \in \{1, n\}$, where $\mathcal{E}$ is the encryption algorithm of a semantically secure symmetric encryption scheme.
Once all documents are encrypted, they will be stored in the outsourced server with their associated labels, see table 1.

$\mathcal{E}_{K_2}(D_1)$	$\mathcal{X}_1$
...	...
$\mathcal{E}_{K_2}(D_i)$	$\mathcal{X}_i$
...	...
$\mathcal{E}_{K_2}(D_n)$	$\mathcal{X}_n$

Table 1: Encrypted documents in the outsourced server

4.2.3 Search and Test phases

We now explain the search phase. This phase is generic as well but we will explain it in three steps for the sake of clarity: first how to proceed with conjunctive queries, then how to take into account disjunctions as well and finally how to deal with negations and hence any boolean expression in disjunctive normal form defined earlier in section 2.

Conjunctive Search: We consider a user who wants to search for all documents that match the query:
"w_{i_1} *and* w_{i_2}*...and* w_{i_r}" in the outsourced server. First of all,

these keywords are converted in the orthonormal base either by regenerating this base or by retrieving the corresponding vectors in table $\mathcal{T}$ if available. The user only selects the orthonormal keywords $(u_i)_{i\in I_q}$ corresponding to keywords existing in the boolean expression where I_q is the set of keyword indexes in the boolean expression.

Afterwards the user picks at random $|I_q|$ strictly positive integers $\{a_k\}_{1\leq k\leq |I_q|}$ such that $a_k \xleftarrow{R} \{1,..,2^r\}$ where r is a sufficiently large positive number. The query is then constructed as follows:

$$\mathcal{Q} = Query(K_1, \bigwedge_{k\in I_q} w_k) = \frac{1}{\sum_{k\in I_q} a_k}.\sum_{k\in I_q} a_k u_k$$

The user sends then to the server the query $\mathcal{Q}$.

On the server side, for each document D_i, the server will perform the following verification:

$$Test(\mathcal{Q}, \mathcal{X}_i) = \phi(\mathcal{Q}, \mathcal{X}_i) = \mathcal{Q}.^t\mathcal{X}_i$$

$$\begin{aligned} Test(\mathcal{Q}, \mathcal{X}_i) &= \frac{1}{\sum_{k\in I_q} a_k}\sum_{k\in I_q}\sum_{l\in M_i} a_k\phi(u_k, u_l) \\ &= \frac{1}{\sum_{k\in I_q} a_k}\sum_{k\in I_q\bigcap M_i} a_k \end{aligned}$$

- If all keywords exist in the document C_i i.e. $I_q \bigcap M_i = I_q$ then: $Test(\mathcal{Q}, \mathcal{X}_i) = 1$ and we add C_i to $\mathcal{L}$.
- Else the document C_i does not contain all keywords i.e. $I_q \bigcap M_i \subsetneq I_q$ and then : $Test(\mathcal{Q}, \mathcal{X}_i) \neq 1$.

Conjunctive and disjunctive search: In this paragraph, we consider a more general case where the user wants to search for the following query:

$$\bigvee_{k=1}^{t}\bigwedge_{j\in I_{q_k}} w_j$$

Again the user translates the keywords in the corresponding orthonormal keywords.

For each $k \in [\![1,t]\!]$, the user picks at random $|I_{q_k}|$ integers $\{a_{k,i}\}_{1\leq i\leq |I_{q_k}|}$ such that $a_{k,i} \xleftarrow{R} \{1,..,2^r\}$ where r is a sufficiently large positive number. We then define:

$$\mathcal{S}_k = \frac{1}{\sum_{j\in I_{q_k}} a_{k,j}}\sum_{j\in I_{q_k}} a_{k,j}u_j$$

The user sends to the server the query in the form of the following matrix:

$$\mathcal{Q} = \mathcal{Q}uery(K_1, \bigvee_{k=1}^{t}\bigwedge_{j\in I_{q_k}} w_j) = \begin{pmatrix}\mathcal{S}_1\\ \vdots \\ \mathcal{S}_t\end{pmatrix}$$

For each document C_i, the server performs the following verification:

$$\begin{aligned} Test(\mathcal{Q}, \mathcal{X}_i) &= \phi(\mathcal{Q}, \mathcal{X}_i) = \mathcal{Q}.^t\mathcal{X}_i = \begin{pmatrix}\phi(\mathcal{S}_1, \mathcal{X}_i)\\ \vdots \\ \phi(\mathcal{S}_t, \mathcal{X}_i)\end{pmatrix} \\ &= \begin{pmatrix}\frac{1}{\sum_{j\in I_{q_1}} a_{1,j}}\sum_{j\in I_{q_1}\bigcap M_i} a_{1,j}\\ \vdots \\ \frac{1}{\sum_{j\in I_{q_t}} a_{t,j}}\sum_{j\in I_{q_t}\bigcap M_i} a_{t,j}\end{pmatrix} \end{aligned}$$

The result of this operation is interpreted as follows:

- If $\exists k \in [\![1,t]\!]$ such that $(Test(\mathcal{Q}, \mathcal{X}_i))_k = 1$ then the document C_i exactly match the query and we add C_i to $\mathcal{L}$, as it means that at least one conjunction was satisfied, hence the disjunction of these disjunctions is satisfied.
- Else $\forall k \in [\![1,t]\!]$, $(Test(\mathcal{Q}, \mathcal{X}_i))_k \neq 1$ then the document does not match the query, because no conjunction was satisfied so the whole disjunction is not satisfied either.

So far, we have seen general queries dealing with any number of conjunctions or disjunctions in the same query. However, to complete our scheme, we need to take into account the negation as well. In the following paragraph, we detail our full solution which is the first solution that supports general boolean searches over encrypted data.

Boolean search: We now consider the case of a general boolean expression search taken in the disjunctive normal form as follows:

$$\bigvee_{k=1}^{t}\bigwedge_{j\in I_{q_k}} \diamond w_j$$

For this construction, we introduce a new keyword which we consider as belonging to all documents by default. To this extent, during the u_i construction, we add to all documents an arbitrary keyword (a word which does not exist in reality). After the Gram-Schmidt process, we thus have one orthonormal keyword J existing in all documents D_i and at the same time $((u_i)_{i\in I}, J)$ represents an orthonormal family.

Let us now consider each conjunction $L_k = \bigwedge_{j\in I_{q_k}} \diamond w_j$ separately. I_{q_k} represents the indexes of keywords in each disjunction L_k. We split I_{q_k} into two partitions such that: $I_{q_k} = \{P_{q_k}, N_{q_k}\}$ where P_{q_k} and N_{q_k} represent respectively the indexes of affirmative and negative keywords in the boolean expression. For each $k \in [\![1,t]\!]$, we pick at random $|P_{q_k}|$ integers $\{a_{k,i}\}_{1\leq i\leq |P_{q_k}|}$, $|N_{q_k}|$ negative numbers $\{c_{k,i}\}_{1\leq i\leq |N_{q_k}|}$ and one random integer b_k such that $a_{k,i} \xleftarrow{R} \{1,..,2^r\}$, $c_{k,i} \xleftarrow{R} \{-2^r,...,-1\}$ and $b_k \xleftarrow{R} \{1,..,2^r\}$ where r is a sufficiently large positive number. We then define:

$$\mathcal{S}_k = \frac{1}{b_k + \sum_{j\in P_{q_k}} a_{k,j}}\Big(\sum_{j\in N_{q_k}} c_{k,j}u_j + \sum_{j\in P_{q_k}} a_{k,j}u_j + b_k J\Big)$$

The user sends to the server the following query:

$$\mathcal{Q} = \mathcal{Q}uery(K_1, \bigvee_{k=1}^{t}\bigwedge_{j\in I_{q_k}} \diamond w_j) = \begin{pmatrix}\mathcal{S}_1\\ \vdots \\ \mathcal{S}_t\end{pmatrix}$$

For each document C_i, the server performs the following verification:

$$\begin{aligned} Test(\mathcal{Q}, \mathcal{X}_i) &= \phi(\mathcal{Q}, \mathcal{X}_i) = \mathcal{Q}.^t\mathcal{X}_i = \begin{pmatrix}\phi(\mathcal{S}_1, \mathcal{X}_i)\\ \vdots \\ \phi(\mathcal{S}_t, \mathcal{X}_i)\end{pmatrix} \\ &= \begin{pmatrix}\frac{1}{b_1 + \sum_{j\in P_{q_1}} a_{1,j}}\big(\sum_{j\in N_{q_1}\bigcap M_i} c_{1,j} + \sum_{j\in P_{q_1}\bigcap M_i} a_{1,j} + b_1\big)\\ \vdots \\ \frac{1}{b_t + \sum_{j\in P_{q_t}} a_{t,j}}\big(\sum_{j\in N_{q_t}\bigcap M_i} c_{t,j} + \sum_{j\in P_{q_t}\bigcap M_i} a_{t,j} + b_t\big)\end{pmatrix} \end{aligned}$$

The result of this operation is interpreted as follows:

- If $\exists k \in [\![1,t]\!]$ *such that* $(Test(\mathcal{Q},\mathcal{X}_i))_k = 1$ then the document C_i exactly match the query and we add C_i to $\mathcal{L}$. Indeed it means that in the corresponding disjunctions all affirmative keywords are in the document and non negative keyword is present (otherwise the result would be smaller than one).

- Else $\forall k \in [\![1,t]\!]$, $(Test(\mathcal{Q},\mathcal{X}_i))_k \neq 1$ then the document doesn't match the query.

After performing the $Test$ algorithm, the server sends to the user the set of all encrypted documents $\mathcal{L}$ that match the query.

Note: The reason why we have included a joker J in all documents is to deal with some particular cases. For instance, the case where we have only negative keywords in the disjunction L_k. Then $|P_{q_k}| = 0$. Without a joker, the query $\mathcal{S}_k$ would have to be divided by 0. By adding the joker, we prevent this case from occurring and the division is at least by a.

5. EVALUATION

In this section we evaluate the security and performance of our solution.

We first note that our scheme is consistent in the sense of definition 3.2. We already hinted to that during the explanation of our construction. Basically we consider each conjunction separately and if at least one of them is satisfied then the whole disjunction is satisfied as well. The test operation at each conjunction level is basically a scalar product between the query and the labels which are both vectors of a vector space expressed with an orthonormal base. Hence if a keyword is present in both the query and the label the result is one otherwise it is zero.The overall result is divided by the weight of positive keywords in the conjunction. If a negative keyword is both in the query and the keyword it will produce a negative component in the sum, hence the sum in the inner product will be strictly less than the divisor.

5.1 Security

THEOREM 5.1. *BSSE probabilistic Query algorithm has randomized queries.*

PROOF. We are going to consider two situations, the first case is when we have the same boolean query $\mathcal{B}(\mathcal{W})$, the second one will be more general since we are going to take queries generated for different boolean queries. We are aiming to prove that:

$$\forall\ i,j \in \{1,l\}\ \ \Pr(\mathcal{Q}_i = \mathcal{Q}_j) < \epsilon\ ,$$

where ϵ tends to zero.

The first case: let us consider the following boolean query $\mathcal{B}(\mathcal{W})$, we apply the probabilistic algorithm Query(K_1,.) (with the same key K_1) q times such that $\{\mathcal{Q}_1, \mathcal{Q}_2, ..., \mathcal{Q}_q\}$ represents the resulting queries. Having the same boolean query implies that, for all queries, we will have the same affirmative and negative keywords in all disjunctions. Let us consider the i^{th} query:

$$\mathcal{Q}_i = \begin{pmatrix} \mathcal{S}_{i,1} \\ \vdots \\ \mathcal{S}_{i,t} \end{pmatrix}$$

where

$$\mathcal{S}_{i,k} = \frac{1}{b_k^{(i)} + \sum_{j\in P_{q_k}^{(i)}} a_{k,j}^{(i)}} \Big(\sum_{j\in N_{q_k}^{(i)}} c_{k,j}^{(i)} u_j + \sum_{j\in P_{q_k}^{(i)}} a_{k,j}^{(i)} u_j + b_k^{(i)} J \Big)$$

We note $\alpha_k^{(i)} = \frac{1}{b_k^{(i)} + \sum_{j\in P_{q_k}} a_{k,j}^{(i)}}$.

Let us take the l^{th} query, since we have the same boolean query we simplify the notation in this case to $N_{q_k}^{(i)} = N_{q_k}^{(l)} = N_{q_k}$ and $P_{q_k}^{(i)} = P_{q_k}^{(l)} = P_{q_k}$. We have then for the first row of the queries:

$$\Pr(\mathcal{S}_{i,1} - \mathcal{S}_{l,1} = 0) = \Pr\Big(\sum_{j\in N_{q_1}} (\alpha_1^{(i)} c_{1,j}^{(i)} - \alpha_1^{(l)} c_{1,j}^{(l)}) u_j + \\ + \sum_{j\in P_{q_1}} (\alpha_1^{(i)} a_{1,j}^{(i)} - \alpha_1^{(l)} a_{1,j}^{(l)}) u_j + (\alpha_1^{(i)} b_1^{(i)} - \alpha_1^{(l)} b_1^{(l)}) J = 0\Big)$$

Since we have that $P_{q_1} \bigcap N_{q_1} \bigcap \{J\} = \emptyset$ (each keyword in the combination is unique) and orthonormalized keywords which represent a free family, we then have for a fixed i and l:

$$\Pr(\mathcal{S}_{i,1} - \mathcal{S}_{l,1} = 0) = \prod_{j\in P_{q_1}} \Pr\Big(a_{1,j}^{(i)} = \frac{\alpha_1^{(l)}}{\alpha_1^{(i)}} a_{1,j}^{(l)}\Big) \times \\ \prod_{j\in N_{q_1}} \Pr\Big(c_{1,j}^{(i)} = \frac{\alpha_1^{(l)}}{\alpha_1^{(i)}} c_{1,j}^{(l)}\Big) \times \Pr\Big(b_1^{(i)} = \frac{\alpha_1^{(l)}}{\alpha_1^{(i)}} b_1^{(l)}\Big)$$

For all $j \in N_{q_1} \bigcup P_{q_1}$ we have $c_{1,j}^{(l)}$, $c_{1,j}^{(i)}$ and $b_1^{(l)}$, $b_1^{(i)}$, $a_{1,j}^{(l)}$, $a_{1,j}^{(i)}$ are respectively sampled uniformly at random from $\{-2^r, ..., -1\}$ and $\{1, .., 2^r\}$ where r is a sufficiently large positive number, we then have:

$$\begin{aligned} \Pr(\mathcal{S}_{i,1} - \mathcal{S}_{l,1} = 0) &= \tfrac{1}{2^n}^{|P_{q_1}|} \times \tfrac{1}{2^n}^{|N_{q_1}|} \times \tfrac{1}{2^n} \\ &= \big(\tfrac{1}{2^n}\big)^{|P_{q_1}|+|N_{q_1}|+1} \end{aligned}$$

We should point out that all rows are independent. Consequently for all rows we will obtain:

$$\begin{aligned} \Pr(\mathcal{Q}_i - \mathcal{Q}_l = 0) &= \prod_{k=1}^{t} Pr(\mathcal{S}_{i,k} - \mathcal{S}_{l,k} = 0) \\ &= \big(\tfrac{1}{2^n}\big)^{\sum_{k=1}^{t}(|P_{q_k}|+|N_{q_k}|+1)} \end{aligned}$$

We say that the scheme has $(l, (\frac{1}{2^n})^{\sum_{k=1}^{t}(|P_{q_k}|+|N_{q_k}|+1)})$-randomized query.

The second case, we consider different boolean query for each time the user sends a query to the server. Consequently, we will have different affirmative and negative keywords for each query $\mathcal{Q}_i$.

We have then for the first row of queries $\mathcal{Q}_i$ and $\mathcal{Q}_l$:

$$\Pr(\mathcal{S}_{i,1} - \mathcal{S}_{l,1} = 0) = \Pr\Big(\sum_{j\in N_{q_1}^{(i)}} \alpha_1^{(i)} c_{1,j}^{(i)} u_j + \sum_{j\in P_{q_1}^{(i)}} \alpha_1^{(i)} a_{1,j}^{(i)} u_j + \\ \alpha_1^{(i)} b_1^{(i)} J - \sum_{j\in N_{q_1}^{(l)}} \alpha_1^{(l)} c_{1,j}^{(l)} u_j - \sum_{j\in P_{q_1}^{(l)}} \alpha_1^{(l)} a_{1,j}^{(l)} u_j - \alpha_1^{(l)} b_1^{(l)} J = 0\Big)$$

Since we can have negative keywords in the first query existing in the affirmative keywords sets of the second query and vice versa, we should take into account all the possible combinations such that:

- $\Gamma_1^{(1)} = N_{q_1}^{(i)} \bigcap (N_{q_1}^{(l)} \bigcup P_{q_1}^{(l)})$
- $\Gamma_2^{(1)} = P_{q_1}^{(i)} \bigcap (N_{q_1}^{(l)} \bigcup P_{q_1}^{(l)})$
- $\Gamma_3^{(1)} = N_{q_1}^{(i)} \setminus (N_{q_1}^{(l)} \bigcup P_{q_1}^{(l)})$
- $\Gamma_4^{(1)} = P_{q_1}^{(i)} \setminus (N_{q_1}^{(l)} \bigcup P_{q_1}^{(l)})$
- $\Gamma_5^{(1)} = N_{q_1}^{(l)} \setminus (N_{q_1}^{(i)} \bigcup P_{q_1}^{(i)})$
- $\Gamma_6^{(1)} = P_{q_1}^{(l)} \setminus (N_{q_1}^{(i)} \bigcup P_{q_1}^{(i)})$

The feature owned by this repartition is the creation of six partitions such that:

$$\Gamma_1^{(1)} \bigcap \Gamma_2^{(1)} \bigcap \Gamma_3^{(1)} \bigcap \Gamma_4^{(1)} \bigcap \Gamma_5^{(1)} \bigcap \Gamma_6^{(1)} \bigcap \{J\} = \emptyset$$

Consequently, the probability of distinguishing two rows of different queries is:

$\Pr(\mathcal{S}_{i,1} - \mathcal{S}_{l,1} = 0) = \Pr(\sum_{j \in \Gamma_1^{(1)}} (\alpha_1^{(i)} c_{1,j}^{(i)} - \alpha_1^{(l)} (a_{1,j}^{(l)} + c_{1,j}^{(l)})) u_j + \sum_{j \in \Gamma_2^{(1)}} (\alpha_1^{(i)} a_{1,j}^{(i)} - \alpha_1^{(l)} (a_{1,j}^{(l)} + c_{1,j}^{(l)})) u_j + \sum_{j \in \Gamma_3^{(1)}} \alpha_1^{(i)} c_{1,j}^{(i)} u_j + \sum_{j \in \Gamma_4^{(1)}} \alpha_1^{(i)} a_{1,j}^{(i)} u_j - \sum_{j \in \Gamma_5^{(1)}} \alpha_1^{(l)} c_{1,j}^{(l)} u_j - \sum_{j \in \Gamma_6^{(1)}} \alpha_1^{(l)} a_{1,j}^{(l)} u_j + (\alpha_1^{(i)} b_1^{(i)} - \alpha_1^{(l)} b_1^{(l)}) J = 0)$

Since we are dealing with orthonormal keywords we have than:

$\Pr(\mathcal{S}_{i,1} - \mathcal{S}_{l,1} = 0) = \prod_{j \in \Gamma_1^{(1)}} \Pr(c_{1,j}^{(i)} = \frac{\alpha_1^{(l)}}{\alpha_1^{(i)}} (a_{1,j}^{(l)} + c_{1,j}^{(l)})) \times \prod_{j \in \Gamma_2^{(1)}} \Pr(a_{1,j}^{(i)} = \frac{\alpha_1^{(l)}}{\alpha_1^{(i)}} (a_{1,j}^{(l)} + c_{1,j}^{(l)})) \times \prod_{j \in \Gamma_3^{(1)}} \Pr(c_{1,j}^{(i)} = 0) \times \prod_{j \in \Gamma_4^{(1)}} \Pr(a_{1,j}^{(i)} = 0) \times \prod_{j \in \Gamma_5^{(1)}} \Pr(c_{1,j}^{(l)} = 0) \times \prod_{j \in \Gamma_6^{(1)}} \Pr(a_{1,j}^{(l)} = 0) \times \Pr(b_1^{(i)} = \frac{\alpha_1^{(l)}}{\alpha_1^{(i)}} b_1^{(l)})$

All sample here are independent from each others, this will imply that:

$$\Pr(c_{1,j}^{(i)} = \frac{\alpha_1^{(l)}}{\alpha_1^{(i)}} (a_{1,j}^{(l)} + c_{1,j}^{(l)})) = \Pr(c_{1,j}^{(l)}) + \Pr(a_{1,j}^{(l)}) - \Pr(c_{1,j}^{(l)}).\Pr(a_{1,j}^{(l)})$$

$$\Pr(c_{1,j}^{(i)} = \frac{\alpha_1^{(l)}}{\alpha_1^{(i)}} (a_{1,j}^{(l)} + c_{1,j}^{(l)})) = \frac{1}{2^{n-1}} - \frac{1}{2^{2n}} \leq \frac{1}{2^{n-1}}$$

Consequently:

$\Pr(\mathcal{S}_{i,1} - \mathcal{S}_{l,1} = 0) \leq \frac{1}{2^{n-1}}^{|\Gamma_1^{(1)}| + |\Gamma_2^{(1)}|} \times \frac{1}{2^n}^{|\Gamma_3^{(1)}| + |\Gamma_4^{(1)}| + |\Gamma_5^{(1)}| + |\Gamma_6^{(1)}| + 1}$

Knowing that all rows are independent. Consequently for all rows we will obtain:

$$\Pr(\mathcal{Q}_i - \mathcal{Q}_l = 0) = \prod_{k=1}^{t} Pr(\mathcal{S}_{i,k} - \mathcal{S}_{l,k} = 0)$$

$\Pr(\mathcal{Q}_i - \mathcal{Q}_l = 0) \leq \frac{1}{2^{n-1}}^{\sum_{k=1}^{t} (|\Gamma_1^{(k)}| + |\Gamma_2^{(k)}|)} \times \frac{1}{2^n}^{\sum_{k=1}^{t} (|\Gamma_3^{(k)}| + |\Gamma_4^{(k)}| + |\Gamma_5^{(k)}| + |\Gamma_6^{(k)}| + 1)}$ □

We have shown that the queries generated are randomized which makes it harder for the server to differentiate them. However the server can know if an orthonormalized keyword exists or not in the query, it can be done by performing an inner product between a randomized generated sequence and the query, the probability to find it out is equal to $\frac{|P_{q_k}| + |N_{q_K}|}{2^{m+l}}$ where m+l is the size of the orthonormalized encrypted keywords, we fix the lower bound of keyword size to m=100 , consequently when $W = w_1$ (which is the most favorable case for the attacker, i.e. there is only one keyword stored in labels), the probability to find it is equal to $\frac{|P_{q_k}| + |N_{q_K}|}{2^{100}}$, on the other hand if we are storing for example 1000 keywords in labels, the size of keywords will then be equal to m+l=1100 and the probability will be equal then to $\frac{|P_{q_k}| + |N_{q_K}|}{2^{1100}}$. Hence it makes it difficult to check if a given query contains a particular keyword, further more, even if the server knows that a query contains a particular keyword, he can not find out if it is an affirmative or negative one with a probability equal to $\frac{1}{2}$. That means that the server can not distinguish if the user search for documents containing this keyword or not. Even if keywords are already secured by using a pseudo-random permutation which make them indistinguishable from a random sample of strings, the aim of our randomized queries is to hide the search pattern of the user, making queries not deterministic.

THEOREM 5.2. *If $\mathcal{E}$ is a semantically secure symmetric encryption scheme and π is a pseudo-random permutation, then BSSE scheme is adaptively semantically secure.*

PROOF. We are going to describe a polynomial-size simulator $\mathcal{S} = \{\mathcal{S}_0, ..., \mathcal{S}_q\}$ such that for all polynomial-size adversaries $\mathcal{A} = \{\mathcal{A}_0, ..., \mathcal{A}_q\}$ the outputs $(o, st_{\mathcal{A}})$ of $Real_{BSSE,\mathcal{A}}(k)$ and $(o', st'_{\mathcal{A}})$ of $Sim_{BSSE,\mathcal{A},\mathcal{S}}(k)$ are computationally indistinguishable. We are going to detail each steps of the simulator in the experiment $Sim_{BSSE,\mathcal{A},\mathcal{S}}(k)$.

$\mathcal{S}_0(1^k, \gamma(D))$: in this phase the simulator based on the history that he knows at this step (i.e. $\gamma(D)$)(which means only the number and the size of documents from the trace), $\mathcal{S}_0$ will generate at random for each $i \in \{1, ..., n\}$ $C'_i \xleftarrow{R} \{0,1\}^{|D_i|}$ and $\mathcal{X}'_i \xleftarrow{R} \{0,1\}^{m+l}$. S_0 then outputs $o = (\mathcal{X}', \mathcal{C}', st_{\mathcal{S}})$. Since $st_{\mathcal{A}}$ does not include K_1, $\mathcal{X}'$ is indistinguishable from a real label, otherwise we can distinguish between the output of π and random values of size m+l which is not true in accordance to the pseudo-random permutation definition. Likewise, since $st_{\mathcal{A}}$ does not include K_2, the output C' of the simulator $\mathcal{S}_0$ is indistinguishable from the real cyphertexts, otherwise we can distinguish between the output of a semantically secure symmetric encryption scheme from a random values of size $|D_i|$.

$\mathcal{S}_1(st_{\mathcal{S}}, \gamma(D, \mathcal{B}(\mathcal{W}_1)))$: At this phase the simulator has the knowledge of all document' identifiers corresponding to the boolean query, but the simulator $\mathcal{S}_1$ has no information about the structure of the boolean query (i.e. the number of keywords in each conjunction, how many negative or affirmative keywords exist in the boolean query and also the number of disjunctions in the structure). Let consider $D(B(\mathcal{W}_1))$ the set of all these identifiers. For all $1 \leq j \leq |D(B(\mathcal{W}_1))|$, $\mathcal{S}_1$ chooses at first step an association between each document identifier and a generated label at the previous phase such that $(D(B(\mathcal{W}_1))_i, \mathcal{X}'_i)$ are pairwise distinct (because each document has one and only one associated label). $\mathcal{S}_1$ then creates $\mathcal{Q}'_1 \xleftarrow{R} \{0,1\}^{m+l} \times \{0,1\}^a$ (where a representes the simulator guess about the number of disjunctions) such that for all $1 \leq j \leq |D(B(\mathcal{W}_1))|$, there exists at least one $1 \leq k \leq a$ such that the inner product $\phi(\mathcal{Q}'_{1,k}, \mathcal{X}'_j) = 1$. $\mathcal{S}_1$ stores the association between $\mathcal{Q}'_1$ and $\mathcal{B}(\mathcal{W}_1)$ in $st_{\mathcal{S}}$, then outputs $(\mathcal{Q}'_1, st_{\mathcal{S}})$. Since $st_{\mathcal{A}}$ does not include K_1, the output $\mathcal{Q}'_1$ is indistinguishable from a real generated query $\mathcal{Q}_1$, otherwise we can distinguish between the output of a pseudo-random permutation from a random sequence of bits of size $\{0,1\}^{m+l} \times \{0,1\}^a$.

$\mathcal{S}_i(st_{\mathcal{S}}, \gamma(D, \mathcal{B}(\mathcal{W}_1), ..., \mathcal{B}(\mathcal{W}_i)))$ for $2 \leq i \leq q$: $\mathcal{S}_i$ check first if the boolean query $\mathcal{B}(\mathcal{W}_i)$ was queried before. The simulator has only to verify in the trace if there exists a $1 \leq j \leq i-1$ such that $\sigma_{i,j} = 1$. If $\sigma_{i,j} = 0$, the boolean query has not previously queried for, the simulator $\mathcal{S}_i$ then follows the same steps as $\mathcal{S}_0$, however he has to make sure that any label associated to document' identifiers are pairwise distinct. On the other hand, if the boolean query has appeared before, the simulator $\mathcal{S}_i$ takes the same generated query. Finally it outputs $(\mathcal{Q}'_i, st_{\mathcal{S}})$ where the output $\mathcal{Q}'_i$ is indistinguishable from a real generated query $\mathcal{Q}_i$, otherwise we can distinguish between the output of a pseudo-

random permutation from a random sequence of bits of size $\{0,1\}^{m+l} \times \{0,1\}^{a}$.

At the end the simulution the output $\mathcal{Q}' = (\mathcal{Q}'_1, ..., \mathcal{Q}'_q)$ is indistinguishable from a real generated query $\mathcal{Q} = (\mathcal{Q}_1, ..., \mathcal{Q}_q)$ and consequently the output of the experiments $Sim_{BSSE,\mathcal{A},\mathcal{S}}(k)$ and $Real_{BSSE,\mathcal{A}}(k)$ are indistinguishable. □

5.2 Performance

From a performance perspective, the most critical step of searchable encryption is the search phase as it is performed by the server for potentially many users. The search complexity of our construction is $O(n)$ inner products where n is the number of documents stored in the server. This is as good as the solution provided by Goh in [13], except that our scheme can handle much more complex queries. The search phase is illustrated in figure 1, in this example, we show that the search is linear in the number of documents, we take also three different cases where documents stored in the server are associated to labels containing 1000, 2000 or 4000 keywords (for all searches we have taken the same boolean query, we did not take into account the network transmission delay or the query construction time). We emphasize that the increase of labels size implies a longer computation phase for each document.

The main novelty in our scheme is the pre-computation phase with the Gram-Schmidt process. As we mentioned there are two options: either performing this computation once and storing the result in a table resulting in $O(l)$ storage complexity or trading storage for computation and performing the Gram-Schmidt process for each query. Intuitively, since storage is cheap the first solution would be more advantageous. However we implemented the keyword orthonormalization construction which include the Gram-Schmidt and the permutation processes and found out that this computation phase is still reasonable although quadratic in the number of keywords as can be seen in figure 2.

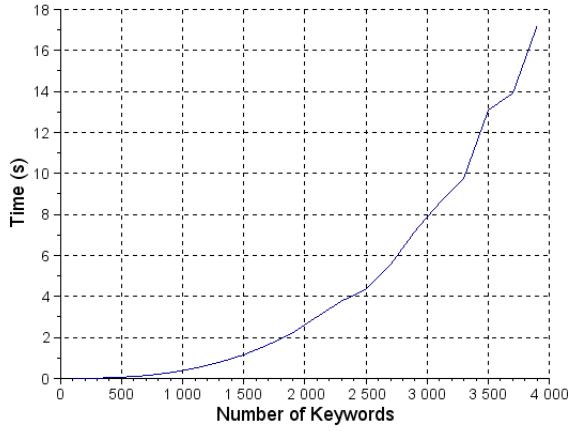

Figure 2: Orthogonal keywords construction

In this figure, the *number of keywords* denotes the size of the free family that we have constructed in the first section. As can be seen on the graph, the construction of one thousand orthogonal keywords takes less 0.5 second which is acceptable given that these computations are performed on a client side. The algorithm used was not optimized in any way, and it simply proves that the solution of recomputing the base at each time is feasible (although we believe that the storage alternative is more efficient). In addition to this implementation, we have tested the performance of the offline construction of labels in the client side, see figure 3 (we do not take into account the extraction of keywords from documents).

6. CONCLUSION

Searchable encryption is a mechanism which enables to perform secure searches over encrypted data stored on outsourced servers. Our work focused on the enhancement of search options over encrypted data. In particular we presented BSSE, the first efficient scheme which supports any boolean expression query, that is to say that our scheme to perform queries composed of conjunction, disjunction and negation of keywords. Our scheme is based on an original approach making use of the Gram-Schmidt orthogonalization process to encode keywords that are further used in labels and queries and inner products to perform the search and leveraging on the orthonormal basis properties. We have proven our scheme secure in an adaptive adversarial model by means of a simulation based formal proof. The efficiency of the scheme is also supported by the fact that the search complexity is in $O(n)$ where n is the number of documents stored in the outsourced server, while the orthogonalization process complexity is quadratic but can be performed offline and can benefit from pre-computations.

As future work, we aim to further enhance the search option as boolean operations only represent a first step towards supporting queries in the form of generic regular expressions over encrypted data. On the other hand, it would be also interesting to create a similar boolean scheme in an asymmetric setting.

7. REFERENCES

[1] M. Abdalla, M. Bellare, D. Catalano, E. Kiltz, T. Kohno, T. Lange, J. Malone-Lee, G. Neven, P. Paillier, and H. Shi. Searchable encryption revisited: Consistency properties, relation to anonymous ibe, and extensions. *J. Cryptology*, 21(3):350–391, 2008.

[2] G. Amanatidis, A. Boldyreva, and A. O'Neill. Provably-secure schemes for basic query support in outsourced databases. *DBSec*, pages 14–30, 2007.

[3] L. Ballard, S. Kamara, and F. Monrose. Achieving efficient conjunctive keyword searches over encrypted data. *ICICS*, pages 414–426, 2005.

[4] M. Bellare, A. Boldyreva, and A. O'Neill. Deterministic and efficiently searchable encryption. *CRYPTO*, pages 535–552, 2007.

[5] D. Boneh, G. D. Crescenzo, R. Ostrovsky, and G. Persiano. Public key encryption with keyword search. *EUROCRYPT*, pages 506–522, 2004.

[6] D. Boneh and B. Waters. Conjunctive, subset, and range queries on encrypted data. *TCC*, pages 535–554, 2007.

[7] J. W. Byun, D. H. Lee, and J. Lim. Efficient conjunctive keyword search on encrypted data storage system. *EuroPKI*, pages 184–196, 2006.

[8] Y.-C. Chang and M. Mitzenmacher. Privacy

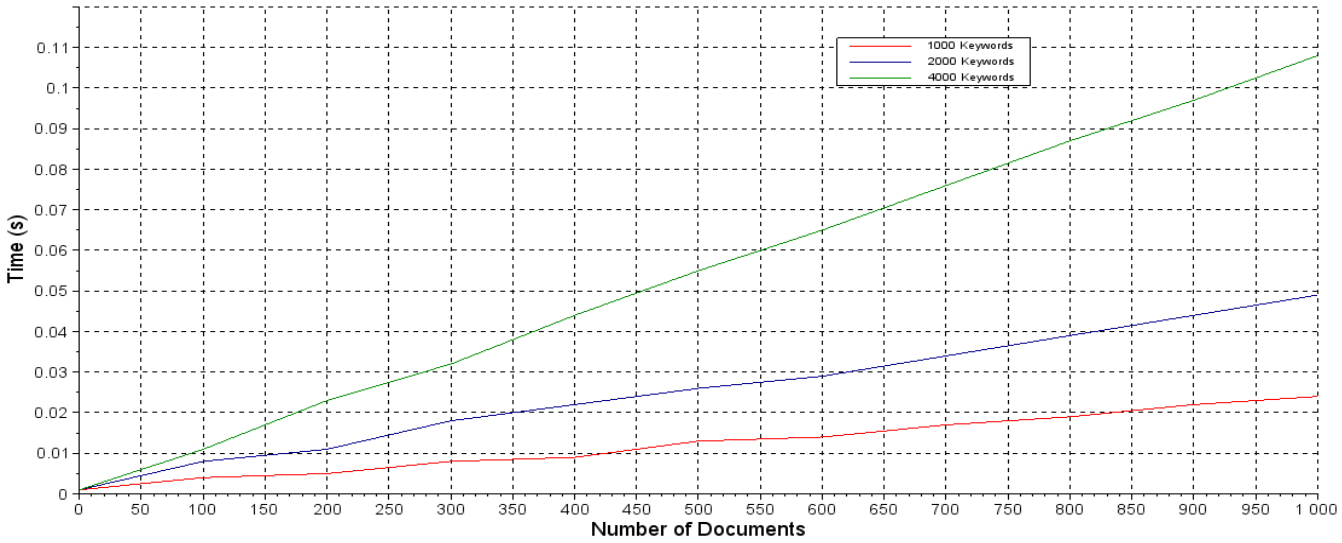

Figure 1: Search phase

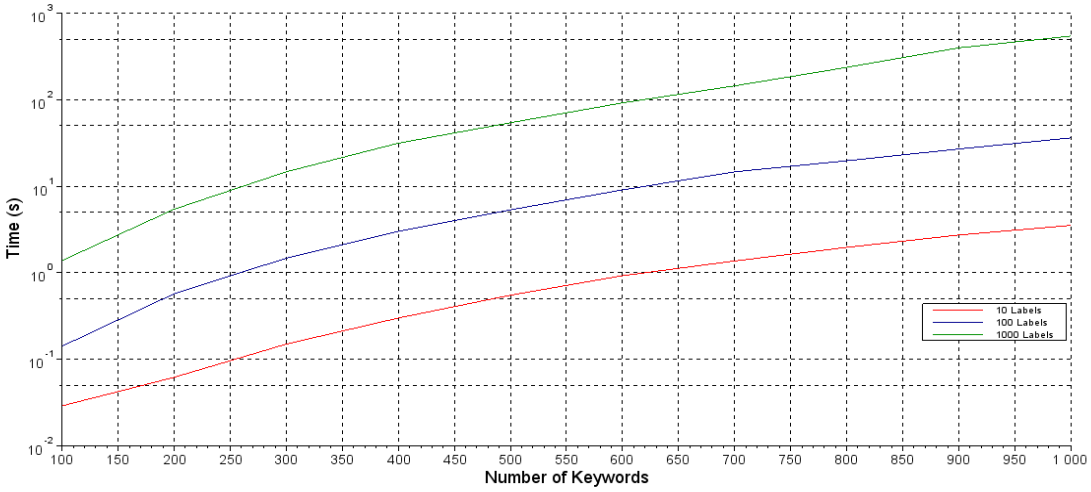

Figure 3: Labels creation

preserving keyword searches on remote encrypted data. *ACNS*, pages 442–455, 2005.

[9] W. Cheney and D. R. Kincaid. Linear algebra: Theory and applications. pages 544–558.

[10] G. D. Crescenzo and V. Saraswat. Public key encryption with searchable keywords based on jacobi symbols. *Progress in Cryptology - INDOCRYPT 2007, 8th International Conference on Cryptology in India, Chennai, India, December 9-13, 2007, Proceedings*, 4859:282–296, 2007.

[11] R. Curtmola, J. A. Garay, S. Kamara, and R. Ostrovsky. Searchable symmetric encryption: improved definitions and efficient constructions. *ACM Conference on Computer and Communications Security*, pages 79–88, 2006.

[12] R. Curtmola, J. A. Garay, S. Kamara, and R. Ostrovsky. Searchable symmetric encryption: Improved definitions and efficient constructions. *Journal of Computer Security*, 19(5):895–934, 2011.

[13] E.-J. Goh. Secure indexes. *IACR Cryptology ePrint Archive*, 2003:216, 2003.

[14] O. Goldreich. Foundations of cryptography volume 2 basic applications. 2004.

[15] P. Golle, J. Staddon, and B. R. Waters. Secure conjunctive keyword search over encrypted data. *ACNS*, pages 31–45, 2004.

[16] Y. H. Hwang and P. J. Lee. Public key encryption with conjunctive keyword search and its extension to a multi-user system. *Pairing*, pages 2–22, 2007.

[17] J. Katz, A. Sahai, and B. Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. pages 146–162, 2008.

[18] D. J. Park, K. Kim, and P. J. Lee. Public key encryption with conjunctive field keyword search. *WISA*, pages 73–86, 2004.

[19] E.-K. Ryu and T. Takagi. Efficient conjunctive keyword-searchable encryption. *AINA Workshops (1)*, pages 409–414, 2007.

[20] E. Shi, J. Bethencourt, H. T.-H. Chan, D. X. Song, and A. Perrig. Multi-dimensional range query over encrypted data. *IEEE Symposium on Security and Privacy*, pages 350–364, 2007.

[21] D. X. Song, D. Wagner, and A. Perrig. Practical techniques for searches on encrypted data. *IEEE Symposium on Security and Privacy*, pages 44–55, 2000.

[22] P. Wang, H. Wang, and J. Pieprzyk. An efficient scheme of common secure indices for conjunctive keyword-based retrieval on encrypted data. *WISA*, pages 145–159, 2008.

[23] P. Wang, H. Wang, and J. Pieprzyk. Keyword field-free conjunctive keyword searches on encrypted data and extension for dynamic groups. *CANS*, pages 178–195, 2008.