Available online at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/cose

Computers & Security

Parallel search over encrypted data under attribute based encryption on the Cloud Computing☆

*Thouraya Bouabana-Tebibel**, *Abdellah Kaci*

Laboratoire de Communication dans les Systèmes Informatiques, Ecole nationale Supérieure d'Informatique, Alger, Algeria

ARTICLE INFO

Article history:
Received 26 December 2014
Received in revised form
20 March 2015
Accepted 18 April 2015
Available online 28 April 2015

Keywords:
Attribute based encryption
Cloud computing
Cryptographic access control
Searchable encryption

ABSTRACT

Data confidentiality in the Cloud Computing is a very challenging task. Encryption is one of the most secure methods ensuring this task, and searchable encryption techniques are used to search on encrypted data without the need for decryption. But, despite this secure measure some leaks may appear when searching over data. In this article, we propose to improve confidentiality of outsourced data. We are particularly interested in reinforcing the access control on the search result, when the search is performed over encrypted data. The property behind this aspect of security is known as ACAS (Access Control Aware Search) principle. We present a hybridization of Searchable Encryption and Attribute Based Encryption techniques in order to satisfy the ACAS property. The proposed model supports a personalized and secure multi-user access to outsourced data, presenting high search performance. It deals with multi-keywords searches and is designed to speed up the search time by taking advantage of High Performance Computing, which is widely used in Cloud Computing. Two Attribute Based Encryption techniques are considered on the side of the Cloud and some conducted experiments show the efficiency of the proposed method.

1. Introduction

The Cloud Computing services offered by large-scale data centers of companies like Amazon, Google, or Microsoft attract every day many new clients. For small and medium enterprises, migration to the Cloud Computing evolves significant economic savings. In fact, the Cloud Computing migration is based on a 'pay per use' pricing model, where users pay according to their resources consumption (Buyya et al., 2010). However, despite the benefits provided by the Cloud migration, new problems and challenges arise like inter-provider data portability, energy conservation, and security (Rong et al., 2013).

With respect to security, the main challenge to deal with is keeping confidentiality of sensitive data. The most powerful mechanism to ensure data confidentiality is encryption. However, the classical cryptographic primitives make data unusable; even authorized users cannot retrieve information from encrypted data. To overcome this problem, Song et al. (2000) proposed a pioneering work dealing with search on encrypted data. Many articles have afterwards been published

☆ This is a revised and extended version of a paper that appeared initially in the proceedings of the 15th IEEE International Conference on Information Reuse and Integration (IEEE IRI 2014) held in San Francisco, CA, USA on Aug 13-15, 2014

* *Corresponding author.*
E-mail addresses: t_tebibel@esi.dz (T. Bouabana-Tebibel), ab_kaci@esi.dz (A. Kaci).

http://dx.doi.org/10.1016/j.cose.2015.04.007
0167-4048/

on this subject (Goh, 2003; Boneh et al., 2004; Curtmola et al., 2006; Tang, 2013).

However, approaches for searching on encrypted data, also known as *Searchable Encryption* methods, present a security breach. They return to the user the identifiers of all requested documents, including those to which he has a denied access. To address this vulnerability, an access control on the searched resources is required. Indeed, in addition to encryption, confidentiality often requires access control mechanisms (Ruj et al., 2012).

In a Cloud Computing environment, the classical access control techniques, such as access control lists (ACLs), show their limits due to the huge number of resources and users. A new encryption approach, based on descriptive attributes, called ABE (Attribute Based Encryption) was introduced. It supports grouped access control facilities rather than only individual. The first publication of ABE appeared in Sahai and Waters (2005), where authors provide a cryptographic access control mechanism based on descriptive attributes. These attributes are used to describe the encrypted data and user's key. Next, the technique is revised in Goyal et al. (2006) and later in Bethencourt et al. (2007) to provide two more elaborate ABE schemes, namely KP-ABE (Key-Policy Attribute Based Encryption) and CP-ABE (Cipher text-Policy Attribute Based Encryption), with enhanced cryptographic access control system. In KP-ABE, descriptive attributes are used to encrypt data, while a user's key contains an access policy defining data to which the user is allowed to access; whereas in CP-ABE, data is encrypted with an access policy based on the users' attributes and the latter serve to construct the user's key. Later, Many other publications on Attribute Based Encryption followed including Ibraimi et al. (2009), Lounis et al. (2013), Yin and Zhang (2011), Yu et al. (2008).

More recently, Han et al. (2014) proposed a new encryption scheme, ABEKS (Attribute Based Encryption with Keyword Search), which ensures confidentiality and multi-user access control based on KP-ABE. They encrypt data documents with KP-ABE using the document keywords as attributes and provide KP-ABE with Searchable Encryption semantics. Search is realizable thanks to a sequential decryption of all documents, based on the user's private key which is made of the searched keywords. The pertinence of the idea behind this work appears in the use of the keywords as document attributes and access policy within the user's decryption key. This strategy allows for gathering, in a judicious way, encrypted search and access control under a unique ABE scheme. CP-ABE may also be, at first, applied to encrypt the original documents in clear, before they undergo the KP-ABE encryption. This added protection reinforces access control that they directly attach to the users' attributes. However, even if ABEKS ensures access control on encrypted search, it leaks the number of documents containing the searched keywords. This leakage refers to the Access Control Aware Search (ACAS) property stated by Singh et al. (2009).

ACAS requires that no additional information should be inferred from the result returned by the search mechanism. Authors of Singh et al. (2009) illustrated their idea by invoking a situation where the violation of the ACAS property leads to the divulgation of the total number of files including a given keyword. For example, an attacker could monitor the enterprise's file system to see the number of files containing the word "bankruptcy". A sudden increase in the number of such files could alert him/her to sell off the company stock, practically amounting to insider trading (Singh et al., 2009). This violates the ACAS property, as this information should not be determined by the attacker through the search result.

The purpose of this work is to improve the SE-ACAS technique proposed in Kaci and Bouabana-Tebibel (2014), Kaci et al. (2014) with a view to integrate access control with search on encrypted data, such that users receive only the results they are authorized to access. Contrarily to ABEKS which resorts to the use of a transformative KP-ABE mechanism to assure search on encrypted data, we dedicate KP-ABE to access control and perform search on encrypted data using the SSE-1 technique (Curtmola et al., 2006) which is index-based search. Access control is based on the Key-Policy Attribute Based Encryption technique (KP-ABE) in Kaci and Bouabana-Tebibel (2014) and the Ciphertext-Policy Attribute Based Encryption technique (CP-ABE) in Kaci et al. (2014). Besides ensuring control access, SE-ACAS was also designed to satisfy the ACAS property. However, the proposed scheme for that purpose showed some limits. It necessitated the use of an additional Filter Authority on the side of the user, requiring, itself, trust mechanisms and maintenance. Furthermore, the experimental results provided high search times due to the cryptographic access control functions performed by the Authority. Thus, we propose herein, a new approach, xSE-ACAS, that improves the SE-ACAS scheme by (1) revising the Searchable Encryption technique in a way it takes advantage of High Performance Computing (HPC), which is widely used in the Cloud, to considerably reduce the search time. The revision is concerned with Searchable Encryption parallelization; (2) performing the access control ABE on the side of the Cloud, thus supporting the ACAS property on one hand and reducing the ABE decryption time thanks to the HPC calculation power, on the other hand; (3) supporting multi-keywords search queries in a form of logical expressions; (4) showing the benefits behind the use of the two main access control techniques based on their performance.

The remainder of the article is organized as follows. In section 2, we present works related to ours. Sections 3 and 4 describe, respectively, the search on encrypted data SSE-1 and the access control ABE, which are the mechanisms on which is based the proposed model. This one is developed in sections 5 and 6. Section 5 describes the parallelization process introduced in the search mechanism; whereas section 6 presents the integration of access control into search. Section 7 provides the solution analysis and section 8 shows the experimental results. We conclude the study in section 9.

2. Related works

The pioneers of Searchable Encryption are Song et al. (2000). They proposed the Searchable Symmetric Key Encryption (SSKE) scheme, which enables a user to retrieve the ciphertext bloc containing a word based on a trapdoor including the searched keyword. Goh proposed in Goh (2003) a more efficient scheme that provides O (1) search time per document. Later, an adaptive solution for Symmetric Searchable

Encryption is published in Curtmola et al. (2006) where the queries to the server can be chosen adaptively (by the adversary) during the execution of search. The authors also considered multi-user Searchable Encryption, while, up to then, previous works limited the search only to the data owner.

Asymmetric Searchable Encryption was proposed by Boneh et al. (2004) who introduced the Public Key Encryption with keyword Search (PEKS) method, which is based on the Bilinear Diffie-Hellman (BDH) assumption technique. Asymmetric Searchable Encryption schemes are appropriate to any setting where the part searching over data is different from the part that generates it. Its main drawback is a weak security control.

With regard to data access control, a new cryptographic system, called Identity Based Encryption (IBE) was introduced in Shamir et al. (1985). IBE enables any pair of users to communicate securely and to verify each other's signatures without exchanging private or public keys, neither keeping key directories, nor using services of a third party. Later, Sahai and Waters proposed in Sahai and Waters (2005) another type of identity based encryption method called Attribute Based Encryption (ABE). In ABE, an identity is viewed as a set of descriptive attributes. An ABE scheme allows a private key for an identity ω, to decrypt a cipher-text encrypted with an identity ω', if and only if the identities ω and ω' are close to each other as measured by the "set overlap" distance metric.

In Goyal et al. (2006) a more efficient ABE encryption model is presented. It is called Key-Policy Attribute Based Encryption (KP-ABE). In KP-ABE encryption model, the access structure is specified within the user's private key, while the cipher-texts are labeled with a set of descriptive attributes. Each user's key consists on an access structure in the form of a tree, where the leaves represent the cipher-text descriptive attributes. A user is able to decrypt a cipher-text if the latter descriptive attributes fit the access policy defined in his key. A more recent study published in Si et al. (2013), proposes to convert into KP-ABE policy the policies written in XACML (eXtensible Access Control Markup Language), which is a common language used to describe access control policies.

Another ABE scheme, called CP-ABE (Ciphertext-Policy Attribute Based Encryption), which enables complex access control on encrypted data. CP-ABE scheme was introduced in Goyal et al. (2006) but the construction of the scheme was done later in Bethencourt et al. (2007). Unlike previous attribute based encryption systems where the attributes describe encrypted data and users' keys contain data access policies, in this system, the attributes keys are used to describe the users' credentials, and a policy defines who is able to decrypt data. Thus, CP-ABE is conceptually close to Role Based Access Control (RBAC) methods. Later, many other publications on CP-ABE followed including (Cheng et al., 2012; Ibraimi et al., 2009; Lounis et al., 2013; Malek and Miri, 2009).

A more complete study, integrating searchable encryption and access control, was published in Kamara and Lauter (2010). The core of the proposed architecture is composed of a data processor, data verifier, token generator, and credential generator. This architecture even though appropriate for supporting data access control and integrity, it does not guarantee access control on the search results and thus, does not conform to ACAS property.

Lately, Han et al. proposed in Han et al. (2014) a method to transform KP-ABE to ABEKS. To make feasible the transformation, they defined a weak anonymity feature, called attribute privacy, which incurs little computational overhead. CP-ABE may first be used to implement an a priori access control. Next, the underlying KP-ABE scheme is applied to support search on encrypted data. However, ABEKS presents a security breach since it does not satisfy the ACAS property; it leaks the number of documents including the searched words. It is, also, less efficient than techniques relying on index-based search. It performs a full documents decryption to retrieve the requested documents; whereas index-based search techniques decrypt only the identifiers of documents including the searched keywords. The undertaken work is rather theoretical than experimental. It provides the security proof of the proposed method; but no performance measures are given.

3. Search over encrypted data

Many solutions have been proposed, since the first work published in Song et al. 2000), to deal with search on encrypted data. In Tang (2013), Tang proposed a new taxonomy to classify Searchable Encryption schemas according to two aspects: encryption setting and search scope.

In our work, we consider the Searchable Encryption schema SSE-1 proposed by Curtmola et al. (2006). According to the taxonomy presented in Tang (2013), the SSE-1 is a symmetric (encryption setting) index-based searchable encryption scheme (search scope). The index I is generated from the data collection before its encryption. Then, the encrypted data collection C as well as the index I are sent to the Cloud. The symmetric key K is used for both indexation and search mechanisms. It is composed of three sub-keys: s, y, and z.

SSE-1 is composed of four algorithms: *KeyGen*, *BuildIndex*, *Trapdoor*, and *Search*. The *Keygen* algorithm produces a random symmetric key K. The *BuildIndex* algorithm generates the search index I from the user's data collection using the symmetric key K. The *Trapdoor* algorithm generates a trapdoor Tw corresponding to a word w using the symmetric key K. The *Search* algorithm takes as inputs the trapdoor Tw and the search index I, and returns the list of the identifiers of documents containing the word w.

3.1. *Structure of SSE-1 index*

The index I is built from the data collection. For each keyword w_i from the collection, the list of the identifiers of documents containing w_i is built. We note $D(w_i)$ this list. The index I is used in the Cloud to search and return the identifiers of the documents containing a keyword w_i. It is built in a way that the Cloud service performs the search without any knowledge of the underlying structure of the index.

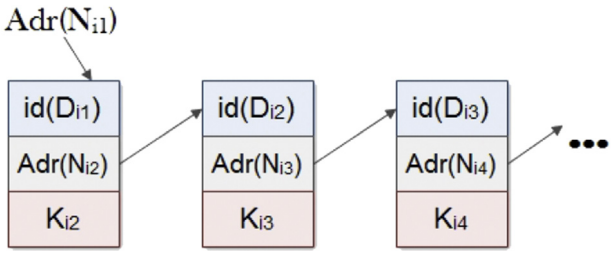

Fig. 1 – Structure of the array A.

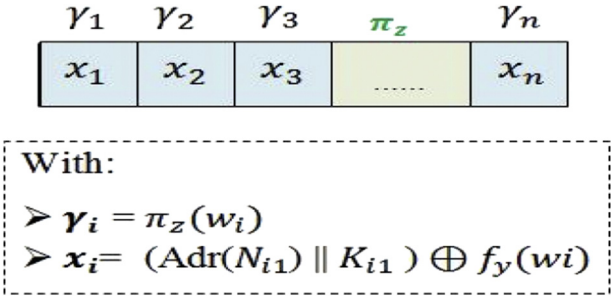

Fig. 2 – Structure of the lookup table T.

The index I is composed of two data structures: an array A and a lookup table T. The array A stores in an encrypted form and random order the lists $D(w_i)$. The lookup table T contains the information necessary to build $D(w_i)$.

Within the array A, each $D(w_i)$ is represented by a chained list where each node N_{ij} is composed of: the identifier of the jth document containing w_i, noted $id(D_{ij})$, the address of the next node of the list, noted $Adr(N_{i(j+1)})$, and the decryption key of that node, noted $K_{i(j+1)}$, (see Fig. 1). Each node N_{ij} is encrypted using a key K_{ij} and stored in a random position in the array A using a pseudo-random permutation Ψ with the sub-key s of the searchable encryption key K. This hides the structure of the index from the Cloud service provider, when allowing it to execute the search requests.

The lookup table T is used to localize and decrypt the heads of the lists $D(w_i)$. Each of its elements provides the position $Adr(N_{i1})$, in the array A, of the first node of the list $D(w_i)$ concatenated to its decryption key K_{i1}. This element, noted x_i in Fig. 2, is scrambled with a pseudo-random function f applied to the keyword w_i using the sub-key y of the searchable encryption key K. The position of x_i in the table T is noted γ_i in Fig. 2. To shuffle the positions of the entries of T, the pseudo-random permutation π is applied to the code of the keyword w_i using the sub-key z of the searchable encryption key K. The array A and the lookup table T are constructed in that order.

3.2. *Trapdoor generation*

In Searchable Encryption, when a user wants to get the documents containing a keyword w_i, he launches his request towards the Cloud with a trapdoor Tw_i calculated using w_i. The trapdoor is composed of a pair $(\pi_z(w_i), f_y(w_i))$, where $\pi_z(w_i)$ is used to locate the entry corresponding to w_i in the lookup table T, whereas $f_y(w_i)$ decrypts $\pi_z(w_i)$ to get the address $Adr(N_{i1})$, on the array A, of the first element of the list $D(w_i)$ and its decryption key.

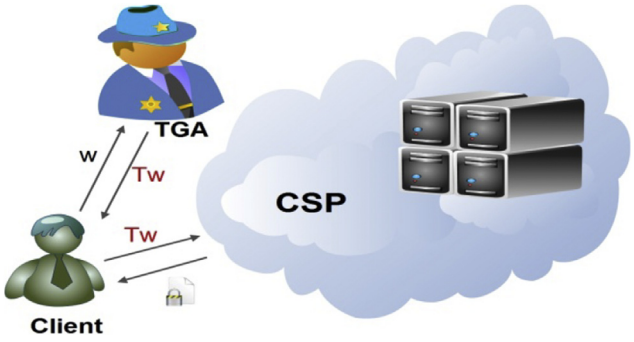

Fig. 3 – Search on the Cloud.

3.3. *Search on the Cloud*

The searchable encryption scheme SSE-1 can be used to implement a search service on encrypted data as shown in Fig. 3. Both the encrypted data collection C and the index I are located on the Cloud. The index serves to search and return the identifiers of the documents containing a keyword w.

When a user wants to search a keyword w, he asks the Trapdoor Generation Authority (TGA) to provide him with the trapdoor Tw corresponding to the searched keyword w. The TGA uses the algorithm *Trapdoor* to generate Tw. Next, the user sends Tw to the search service. The later uses Tw to search in the index I and returns the identifiers of documents containing the keyword w.

However, the search service described above presents a security breach. It does not ensure an access control on the search result, as it returns to the user all identifiers of documents containing the keyword w including those not authorized for him.

4. Attribute based access control

The attribute based access ABE technique is widely used to ensure cryptographic access control in the Cloud Computing. Two main ABE schemes exist in the literature, namely: The Key Policy Attribute Based Encryption (KP-ABE) scheme and the Cipher-text Policy Attribute Based Encryption (CP-ABE) scheme.

4.1. *Key-Policy Attribute Based Encryption*

The Key-Policy Attribute Based Encryption (KP-ABE) is an ABE scheme where the access structure is embedded in the users' private keys, while the cipher-texts are labeled with attributes. A user is able to decrypt a cipher-text if the latter attributes satisfy the access structure of the key.

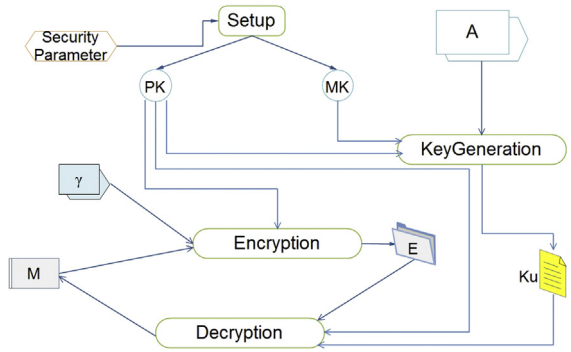

Fig. 4 – KP-ABE scheme.

KP-ABE scheme consists of four algorithms: Setup, Encryption, KeyGeneration, and Decryption (see Fig. 4).

The *Setup* algorithm is a randomized algorithm that takes as input the implicit security parameter. It outputs the public parameters PK and a master key MK. The *Encryption* algorithm is a randomized algorithm that takes as input a message m, a set of attributes γ, and the public parameters PK. It outputs the cipher text E. The *KeyGeneration* algorithm is a randomized algorithm that takes as input an access structure A, the master key MK and the public parameters PK. It outputs a decryption key Ku. Finally, the *Decryption* algorithm takes as input the cipher text E that was encrypted under the set γ of attributes, the decryption key Ku corresponding to the access control structure A and the public parameters PK. It outputs the message M if γ satisfies A.

4.2. *Ciphertext-policy attribute-based encryption*

The Ciphertext-Policy Attribute-Based Encryption (CP-ABE) differs from KP-ABE in that the keys are used to describe the users' attributes and the policy defining who is able to decrypt data is embedded in the ciphertext.

CP-ABE scheme is composed of four algorithms: Setup, Encrypt, KeyGen, Decrypt, and optionally a fifth algorithm Delegate (see Fig. 5). Setup algorithm generates the public parameters PK and the master key MK. The encryption algorithm Encrypt produces the encryption CT of the original message M, which can be decrypted only by users having a set of attributes that satisfy the access structure attached to the cipher-text CT. The key generation algorithm KeyGen generates a private key Ku containing the set of attributes describing the user.

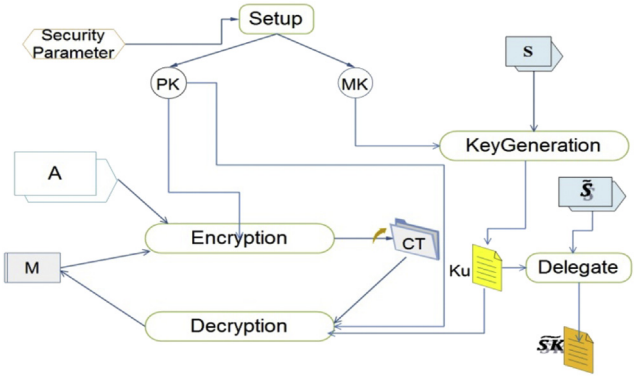

Fig. 5 – CP-ABE scheme.

The decryption algorithm Decrypt takes as input the public parameters PK, a cipher-text CT (associated with an access structure A), and a private key Ku (that corresponds to a set of attributes S). If the set S satisfies the access structure A, then the algorithm decrypts the cipher-text CT and returns the original message M.

5. Parallelization of searchable encryption

To take advantage of High Performance Computing (HPC) in the Cloud, we enhanced the original SSE-1 technique with xSSE-1 (eXtended SSE-1), so that it supports parallelization of search over the cluster's nodes. To achieve this, we provided a new structure of the index I, where the Array A is fragmented into blocs and a new data structure H is considered. The main role of H is to Locate in the array A the different blocs corresponding to the identifiers of documents containing a keyword.

5.1. *The new structure of the index*

In the new searchable encryption technique xSSE-1, the index I is composed of three data structures A, T, and H, where H is added to permit parallel search. The new structure of the index I is described in Fig. 6. The latter shows a keyword w_i occurring in n sub-lists. The array A contains yet chained lists, except that here, the document identifiers for a given keyword are split up into several lists, rather than grouped into only one list. Thus, the list $D(w_i)$ is divided into n sub-lists, where n is the cluster nodes number. Each sub-list includes $m = |D(w_i)|/n$ document identifiers. The address of each sub-list head is stored in a separate entry in H, and chained with the other sub-lists heads, for each keyword w_i. Chaining is performed in an encrypted and scrambled order as it is done for the array A. The address of the first sub-list, within H, is provided by the lookup table T. The other sub-list addresses are encrypted and chained to that of the first one.

5.2. *The new BuildIndex algorithm*

According to the new structure of the index I, the BuildIndex algorithm produces three data structures: the array A, the lookup table T, and the intermediate array H. At the initialization phase, for each keyword w_i in the data collection D, the $D(w_i)$ list of all documents containing the keyword is built. Next, the nodes of the list $D(w_i)$ are encrypted and stored in the array A. This list is, afterwards, divided into n sub-lists. The address, in the array A, of the first node of each sub-list, is stored it in the node X_{ip} of the array H. Finally, the address of the first sub-list of $D(w_i)$ is stored in an entry of the lockup table T. The algorithm BuildIndex(K,D,n) is given as follows.

```
Algorithm BuildIndex (K, D, n)
BEGIN
   /*initialization*/
   delta = construct_distinct_words(D)
   for all w_i in delta do
/*build D(w_i), the list of documents containing w_i*/
      for d_j in D do
         if(d_j contains w_i)
            then
               D(w_i).add(ID(d_j))
         endif
      end loop
   end loop
   ctrA ← 0
   ctrH ← 0
   /*Build the data structures of I : A, H, and T*/
   for all w_i in delta' do
      V_i1 ← generate_random_key(l)
      j ← 0
      for p ← 1 to n do /*for each node*/
         V_i(p+1) ← generate_random_key(l)/*V_ip Key used to encrypt H elements*/
         K_p1 ← generate_random_key(l) /*K_pt Key used to encrypt A elements*/
         t ← 1
         while(t <= |D(w_i)| DIV n + 1 AND j <= |D(w_i)|)
            j ← j + 1
            ctrA ← ctrA + 1
            K_(p+1)t ← generate_random_key(l)
            N_pt ← id(d_ij)//K_(p+1)t//ψ1_s(ctrA+1) /*ψ1 and ψ2 are pseudo-random permutations*/
            A[ψ1_s(ctr)] ← ENC(K_pt, N_pt)
            if(t = 1) /*for the first element*/
               then
               /*the head of the sub-list will be stored in H*/
               ctrH ← ctrH + 1
               X_p ← ψ1_s(ctrA)//V_i(p+1)//ψ2_s(ctrH+1)
               H[ψ2_s(ctrH)] ← ENC(V_ip, X_ip)
            endif
         end loop
      end loop
      /*Lockup table T*/
      AdrX_0 ← ψ2_s(1) /*the address of the first element in H corresponding to w_i*/
      value ← (AdrX_0//V_i1) ⊕ f_y(w_i) /f is a pseudo-random function/
      T[π_z(w_i)] ← value
 end loop
   return I = (A,H,T)

END
```

6. Search on encrypted data with access control

The xSSE-1 model, which is based on SSE-1, is mono-user. It does not allow a user to share data securely with other users, since no access control mechanism is assured. Besides, it supports search queries composed of only one keyword.

Our proposed solution, called xSE-ACAS (eXtensible Searchable Encryption under ACAS), is designed to (1) allow multi-users data exploitation under efficient access control mechanisms, and (2) deal with real world application requirements, where users can express queries in form of propositional expressions on keywords (like it is done with search engines). xSE-ACAS is based on xSSE-1, thus taking advantage of High Performance Computing. We present in the follows, step by step, the elaboration of xSE-ACAS starting from xSSE-1.

6.1. *xSE-ACAS overview*

The xSE-ACAS scheme is designed to be a part of a secure Cloud storage service, where confidentiality is based on SSE-1

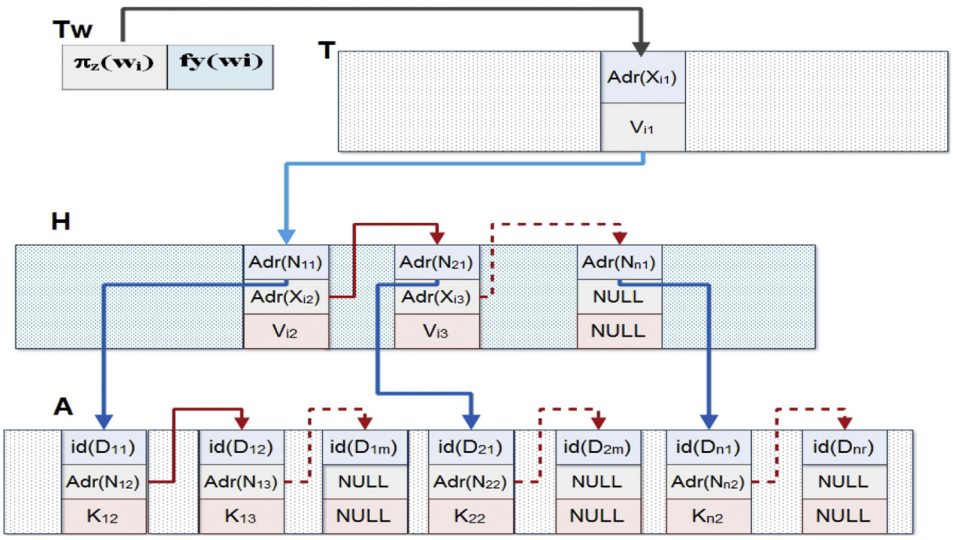

Fig. 6 – New structure of the index I.

encryption and access control is applied on the document identifiers using either KP-ABE or CP-ABE methods.

When a data owner decides to outsource his documents collection (see Fig. 7), he first builds (BuildIndex(D)) the index I from his collection as detailed in section 5, and then sends it (EncryptIndex(I, ABEType)) along with the access rights to the ABE Authority. The latter encrypts the document identifiers according to the ABE strategy, and returns them (EncryptedIndex(I)) to the data owner. It also returns (ABEUserKey(Ku)) the users' ABE keys Ku to users. After generation of the index I, the document collection D is encrypted (Encrypt(D)) providing the collection C. Next, both the encrypted document collection C and the index I are sent (SendToCloud(C, I)) to the Cloud storage service.

When a user wants to launch a search, he expresses his request as a logical proposition composed of a set of keywords w_j. His search application calls (CreateTrapdoor(w)) the TGA authority which generates and returns (Trapdoor(Tw)) a trapdoor corresponding to the logical proposition (see section 6.3). Next, the generated trapdoor is sent (CloudSearchRequest(Tw, Ku)), along with the user's ABE key Ku, obtained from the ABE Authority, to the Cloud storage service.

Once the Cloud search service receives the request, it performs (Search(I, Tw)) on the index I, a search to get the list of the identifiers of documents containing the keywords. Next, to ensure access control, the search result is filtered (AccessControl(List, Ku)) such that only documents accessible to the user are returned to him. The Cloud storage service returns (DocumentIdentifiers(L)) to the user those document identifiers. The user can then get (GetDocument(id)) the documents he is interested in (ReturnDocument(doc))

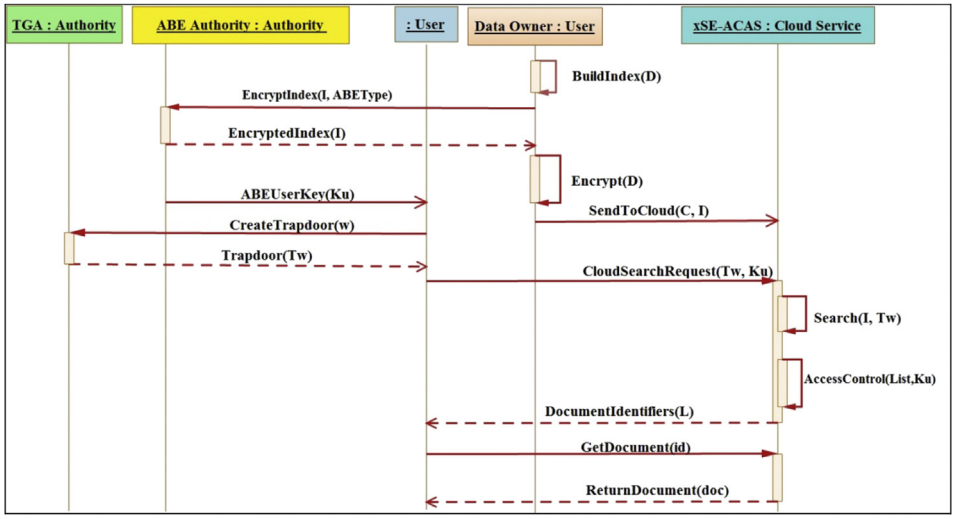

Fig. 7 – Cloud storage service with access control aware search.

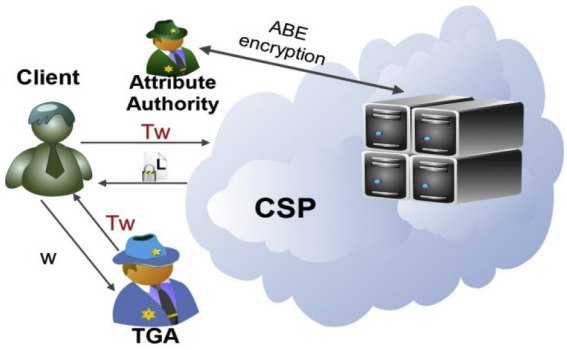

Fig. 8 – xSE-ABE model.

6.2. *Access control to the document identifiers*

To ensure access control on the result of Searchable Encryption, we propose to integrate ABE with xSSE-1. Thus, xSSE-1 is revised to provide a search result made of document identifiers encrypted with ABE rather than transcribed in a clear form as it is done in SSE-1. We call this solution xSE-ABE (eXtended Searchable Encryption - ABE). It is achieved through an ABE authority (see Fig. 8), implemented according to an ABE scheme (KP-ABE scheme or CP-ABE scheme).

The ABE authority encrypts the document identifiers stored in the array A of index I (see Fig. 9). In the case of KP-ABE, each document identifier is encrypted with the set of corresponding descriptive attributes. If CP-ABE is used, the document identifier is encrypted using a policy based on the users' attributes. The structure of the lockup table T and the arrays H and A are unchanged.

The ABE authority is also in charge of distributing the users' ABE keys K_u. In KP-ABE, the user's decryption key holds a policy defining the access to document identifiers. In CP-ABE, the decryption key contains the user's descriptive attributes.

Apart from the identifiers encryption using ABE, the indexation mechanism in xSE-ABE is identical to that of xSSE-1. The BuildIndex algorithm of xSE-ABE calls, in the initialization phase, the setup algorithm (abe-setup) to generate the master key MK and the public parameters PK. When building the array A, the algorithm performs the ABE encryption of the identifiers.

6.3. *Multi-keywords search*

The search mechanism in xSE-ABE is designed to support multi-keywords queries. The latter are propositional expressions formed of conjunctions, disjunctions, and/or negations. They are generated by the following grammar:

$w \rightarrow w_i|w_i o w_i|w_i o w$

$w_i \rightarrow c|cw_i$

$c \rightarrow a|b|\ldots|z|A|\ldots|Z$

$o \rightarrow and|or|not$

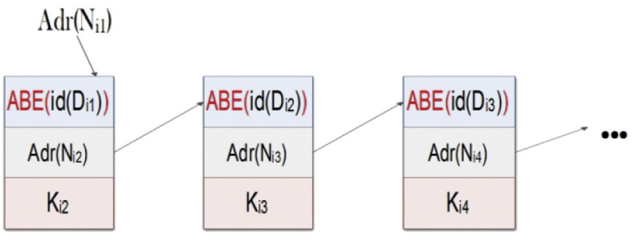

Fig. 9 – The array A in xSE-ABE.

where the non-terminal symbol w represents the axiom of the grammar. It provides the propositional expression used for the search request. The non-terminal symbol o generates the three logical operators of conjunction (and), disjunction (or), and negation (not). The non-terminal symbol w_i generates the keywords.

When a user wants to search documents, his query w, composed of connected keywords w_i, is replaced by a trapdoor Tw at the TGA authority (see Fig. 7). The resulting Tw, is a logical expression similar to w, where $\{w_i\}$ are replaced by the corresponding $\{Tw_i\}$. At receipt of the trapdoor Tw, the Cloud search service, first, decomposes Tw in a set of Tw_i to be searched separately. Next, as shown in Fig. 6, for each Tw_i, it uses the value of the second field, $f_y(w_i)$, to decrypt the element of the table T located at the address specified by the value $\pi_z(w_i)$ of the first field. This decryption noted $(Adr(X_{i1})||V_{i1}) \oplus fy(w_i)$ yields the address of the first node X_{i1} in the array H. Next, each node X_{ip} within H, is decrypted by means of a key V_{ip} yielding the address $Adr(N_{p1})$ of the node N_{p1} in the array A. Once the node N_{p1} is decrypted using the key $V_{i(p+1)}$, the whole sub-list p of $D(w_i)$ can be obtained. Each node N_{pj} of the array A will be used to locate and decrypt the next node $N_{p(j+1)}$ thanks to the address of this next node $N_{p(j+1)}$ and its decryption key $K_{p(j+1)}$.

The search results generated by all Tw_i are evaluated based on the logical expression Tw, before they are returned to the user. To do so, the search engine, first, evaluates all conjunctions and negations. Conjunctions are treated as intersections "∩" of the $D(w_i)$ resulting lists. Negations are carried out as subtractions "\" between the returned $D(w_i)$ lists - the right term is removed from the left one. Next, disjunction operators are evaluated as unions "∪" of connected sets. Once Tw is evaluated, the resulting document identifiers are returned to the user. These document identifiers are encrypted with ABE. The user decrypts those he has access to by using his ABE key.

To illustrate the mechanism of search with multi-keywords, let's consider an example where the document collection D is presented in Table 1, with the rows as documents and the columns as keywords.

The inverted index corresponding to this document collection D is described in Table 2.

Let's consider the search query "w_1 NOT w_2 OR w_3 AND w_5". It is first converted by the TGA authority to "Tw_1 NOT Tw_2 OR Tw_3 AND Tw_5" and sent to the Cloud search service. The latter first evaluates the negation expression "Tw_1 NOT Tw_2" and

Table 1 – Data collection D.

	w1	w2	w3	w4	w5	w6
d1	X	X		X		X
d2	X		X		X	
d3			X	X		
d4	X		X	X		X
d5		X			X	
d6	X	X	X		X	X
d7		X	X	X	X	

Table 2 – Inverted index corresponding to the document collection D.

Keyword w_i	$D(w_i)$
w_1	{ABE(d1), ABE(d2),ABE(d4),ABE(d6)}
w_2	{ABE(d1), ABE(d5), ABE(d6), ABE(d7)}
w_3	{ABE(d2), ABE(d3), ABE(d4), ABE(d6), ABE(d7)}
w_4	{ABE(d1), ABE(d3), ABE(d4), ABE(d7)}
w_5	{ABE(d2), ABE(d5), ABE(d6), ABE(d7)}
w_6	{ABE(d1), ABE(d4), ABE(d6)}

the conjunction expressions "Tw_3 AND Tw_5". The first expression returns the set of documents containing w_1 but not w_2, namely $L1 = D(w_1) \setminus D(w_2) = \{ABE(d2), ABE(d4)\}$. The second expression returns $L2 = D(w_3) \cap D(w_5) = \{ ABE(d4), ABE(d6)\}$. Next, the disjunction is evaluated. It results in $L1 \cup L2 = \{ABE(d2), ABE(d4), ABE(d6)\}$.

Now, suppose the user query "w_1 w_3 w_6" - where the space between the keywords is interpreted as a conjunction. The search engine receives the logical expression "Tw_1 AND Tw_3 AND Tw_6". A sequential search of Tw_1, Tw_3 and Tw_6 provides respectively $D(w_1)$, $D(w_3)$ and $D(w_6)$. An intersection between the three sets results in {ABE(d4), ABE(d6)}.

6.4. *Access control under ACAS*

xSE-ABE allows a user to have access only to the document identifiers authorized for him through his ABE key. However, this scheme does not satisfy the ACAS property whose principle requires that no additional information to the requested one should be provided by the search process. xSE-ABE leaks the number of documents containing a keyword expression w in the data collection. This number is deduced from the search mechanism which returns even the document identifiers not allowed for the user. To address this issue and ensure the ACAS property, we propose the model xSE-ACAS.

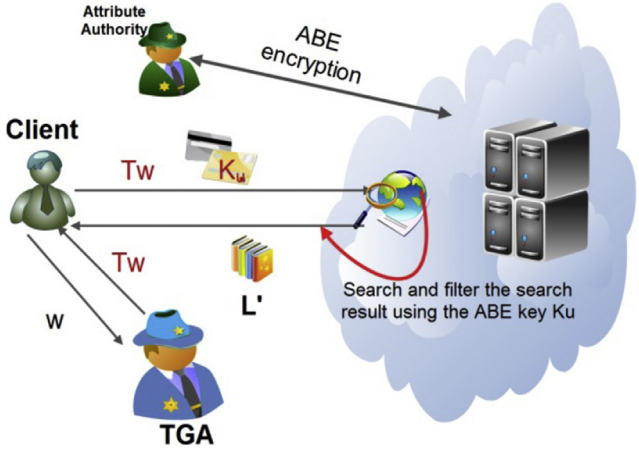

Fig. 10 – The xSE-ACAS model.

In xSE-ACAS (Fig. 10), the user sends to the Cloud search service a request composed of the trapdoor Tw and the user's ABE key Ku in an encrypted form. The search service uses the trapdoor to get the list of encrypted identifiers in the same way as done with xSE-ABE. Next, it decrypts this list using the user's ABE key. To take advantage of the HPC calculation power, ABE decryption of the resulting document identifiers is distributed over all nodes of the cluster. Thus, each node decrypts a sub-set of identifiers and returns only those accessible to the user. The search result returned to the user is the aggregation of all nodes result. It does not leak any additional information.

The search algorithm Search(I,Tw) describes search for the trapdoor Tw within xSE-ACAS.

```
Algorithm Search (I, Tw, Ku)
BEGIN
   for Tw_i in Tw
      theta ← T[γ]
      alpha_V ← theta ⊕ η
      alpha ← alpha_V[0]
      V ← alpha_V[1]
      j ← 0
      while (alpha!=null)
         X ←DEC(V,H[alpha])
         adrA ← X[0]
         k ← X[1]
         V ← X[2]
         Alpha ← X[3]
         Lw[j] ← SearchOnHPCNode(j, adrA, k)
         j ← j +1
      End loop
      L[Tw_i] ← U_j L_w[j]
   end loop
   Lexp ← Evaluate(L, Tw)
   L_ABE ← ABEDecrypt(Lexp,Ku)
   return L_ABE
END
```

Let's consider the first example of section 6.3. Every element of the search result {ABE(d2), ABE(d4), ABE(d6)} is decrypted in a separate cluster node using the user's ABE key Ku submitted in the query. Only documents accessible to the user are sent to him.

7. Solution analysis

The proposed xSE-ACAS model is designed to secure data outsourced to the Cloud storage service, by using cryptographic primitives for data confidentiality and access control. To ensure confidentiality, the xSE-ACAS model encrypts and scrambles the index structure so that it cannot be discovered by the Cloud storage service. The latter is assumed to be "honest-but-curious". This means that the search service behaves honestly but may catch confidential information about users.

xSE-ACAS also provides an ABE access control aware search mechanism ensuring that a user receives, after search, only documents he is allowed to access. The proposed access control mechanism has also been designed to protect against internal malicious statistical attacks launched to obtain information inaccessible through the access control mechanism. This protection is achieved through an access control on the document identifiers, performed on the side of the Cloud, by means of the user's ABE key. We note that sending the ABE key, in an encrypted form, to the Cloud does not constitute a security threat. It does not cause any indiscretion concern on the side of the Cloud nor does it fail the security scheme as the ABE key is used to protect against users' curiosity rather than the Cloud's one. As a result, the proposed model is designed to support secure multi-user data-access service. Here, multi-user data-access service means that more than one user may access to a shared data collection to retrieve only pieces of information not denied to him.

Two main access control mechanisms based on descriptive attributes are used, namely the KP-ABE and CP-ABE solutions. KP-ABE supports a strategy that does not necessarily require continuous work – with the arrival of new users. The strategy responds to a global policy regarding the enterprise security, defined on data and then distributed to the users; users share the same key if they have the same rights. The strategy is thus independent of changes in the users' arrival/departure. However, as many users may share the same key, undesirable access cannot be personally identified. Thus, KP-ABE provides a technique which is rather flexible but ineffective in case of intrusive access. As to CP-ABE, it relies on a control directed towards users. Every user has his own key, calculated on the basis of his specific attributes. Thus, new users call for the calculation of new keys, but every access is personally identified. So, in terms of access strategy, it appears that KP-ABE controls access to data, whereas CP-ABE controls user's access.

xSE-ACAS is suitable for real-world applications where users can express their search requests using expressions containing many keywords connected by means of conjunction, disjunction and negation operators.

However, the xSE-ACAS model does not deal with data integrity as the search service is assumed to be "honest-but-curious", so exempt from forgery attacks. It does not provide data accountability either.

Finally, the xSE-ACAS is designed to take advantage of high performance computing by parallelizing the search mechanism and distributing it on the different nodes, which permits to the Cloud service provider to reduce the search time. The analysis of the xSE-ACAS security is summarized in Table 3.

8. Experimentation

To evaluate the proposed solution, we implemented the new model xSE-ACAS as well as the SE-ACAS model developed in Kaci and Bouabana-Tebibel (2014), Kaci et al. (2014). The new implementation of SE-ACAS considers a multi-keywords search query and a sequential search, whereas the previous published one was based on a single-keyword search query and a sequential search. The implementation of xSE-ACAS is performed on a cluster (HPC) composed of 32 nodes. Experimentations on xSE-ACAS and SE-ACAS models are conducted with the two variants KP-ABE and CP-ABE for the cryptographic access control, thus yielding two implementations for each model. The implementations are performed, for each of the KP-ABE and CP-ABE variants, with the appropriate ABE Authority. The used programming language is Python, which is supported by the open source Cloud platform AppScale. Three main experiments are carried out. In the first experiment, we test the reliability of xSE-ACAS. In the second experiment, we evaluate and compare the search time provided by each of xSE-ACAS and SE-ACAS models. We focus on the search time since the new work comes up with a solution whose main concern is to improve this metric. Furthermore, as observed in our previous work (Kaci and Bouabana-Tebibel, 2014), the search time is independent of the size of the whole collection. So, we study herein the search time variation according to the size of lists of documents containing the searched keywords. In the third experiment, we compare the xSE-ACAS performance resulting from the two variants KP-ABE and CP-ABE.

8.1. *xSE-ACAS security*

In this experiment, we test xSE-ACAS reliability with regard to access control. Users with different rights encrypted into their private keys try to access documents. We show herein that only documents authorized for them are returned to them. We consider queries composed of three keywords.

8.1.1. *xSE-ACAS security based on KP-ABE*

To deal with the xSE-ACAS model based on KP-ABE, we implemented the ABE Authority using the functional encryption library libfenc (Green). We enhanced our implementation using the charm-crypto framework (v0.43) (Joseph Ayo Akinyele).

Each user's key, provided by KP-ABE authority, contains an access policy in the form of a threshold access structure composed of conjunctions and disjunctions over the space of attributes. We conducted our experiments using the scenario showed in Fig. 11. Three users are considered. Each one is related to a policy, encrypted in his private key, defining which documents are authorized for him. Each document identifier

Table 3 – Security analysis.

Attack type	Proposed solution
Leakage	-Documents are encrypted using Advanced Encryption Standard (AES) primitives. -Indexes are encrypted with AES primitives and scrambled using pseudo-random functions and permutations.
Intrusion	-Document identifiers are encrypted using ABE -CP-ABE supports a personalized identification of intrusive access while KP-ABE does not
Insider statistical attacks	The search result is filtered on the side of the Cloud according to access control rules using ABE.
Falsification	-Use of secure communication protocols. -Assumption of a Cloud that is "honest-but-curious"
Cyber crimes	Accountability should be supported by a complementary solution for an adequate auditing with the Cloud Computing context.

is labeled with a set of descriptive attributes, whose values are taken in: A, B, C, D, E, F.

We also consider seven documents d1–d7 whose attributes are given in Table 4, and six keywords w_1–w_6 distributed in the documents as shown in Table 4.

We execute four search queries Q1–Q4, each one composed of three keywords. The search results are presented in Table 5 per query and per user. For instance, the query Q2 corresponds to the set of document identifiers {d1,d2,d5,d6,d7}. However, according to the ABE access control policy embded in the users' keys (see Fig. 11) user1 gets only document d1, while user2 accesses documents {d2,d6,d7} and user3 gets no document.

8.1.2. *xSE-ACAS security based on CP-ABE*

To evaluate xSE-ACAS security using a CP-ABE scheme, we implemented the ABE Authority using the cpabe toolkit (Advanced Crypto Software Collection). We conducted our experiments using the scenario showed in Fig. 12, where the users' CP-ABE keys are generated by the ABE authority based on the users' descriptive attributes whose values are taken in the set: A, B, C, D, E, F. In this variant of xSE-ACAS model, the outsourced document identifiers are encrypted according to a policy defining the conditions that users' attributes must satisfy to permit a decryption.

We also considered seven documents d1-d7 whose attributes are given in Table 6, and six keywords w_1-w_6 distributed in the documents as detailed in Table 6.

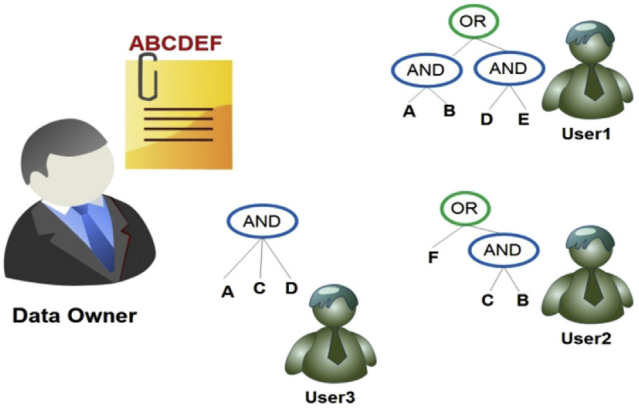

Fig. 11 – The experimental scenario for xSE-ACAS with KP-ABE.

We execute four search queries Q1-Q4, each one composed of three keywords. The search results are presented in Table 7 per query and per user. For instance, the query Q2 corresponds to the set of document identifiers {d1,d2,d5,d6,d7}. However, according to the ABE access control policy embded in the encrypted documents (see Fig. 12) user1 and user2 get only the documents {d2,d5}, while user3 gets the documents {d1,d2, d5,d6}.

8.2. *xSE-ACAS performance based on parallelization*

In this experiment, we are interested in xSE-ACAS performance based on parallelization. The used scenarios are presented in Table 8. We consider search queries composed of five keywords for both xSE-ACAS and SE-ACAS. We

Table 4 – Experimental data for security analysis of xSE-ACAS with KP-ABE.

Document	Attributes	w_1	w_2	w_3	w_4	w_5	w_6
d1	A B	x	x		x		
d2	C D F		x		x	x	
d3	A B E	x		x			x
d4	C E				x		
d5	A E		x			x	
d6	A F	x	x	x		x	x
d7	B F		x		x		

Table 5 – Results of security experiments for xSE-ACAS with KP-ABE.

Search query	User	Search result
Q1 = "w_1 AND w_2 AND w_3"	User1	∅
	User2	{d2}
	User3	∅
Q2 = "w_2 AND w_4 OR w_5"	User1	{d1}
	User2	{d2,d6,d7}
	User3	∅
Q3 = "w_1 AND w_3 OR w_6"	User1	{d3}
	User2	{d6}
	User3	∅
Q4 = "w_2 AND w_4 AND w_5"	User1	∅
	User2	{d2}
	User3	∅

showed in Kaci and Bouabana-Tebibel (2014) that the search time is independent of the size of the whole collection. So, we evaluate the search time variation according to the size of lists of documents containing the searched keywords.

8.2.1. *xSE-ACAS performance based on KP-ABE*

In this experiment, we study the search time variation according to the number of occurrences of searched keywords in the data collection, as shown in Table 8. We executed search queries for the two models SE-ACAS (without parallelization) and xSE-ACAS (on a cluster formed of 32 nodes) according to the scenarios of Fig. 11. The results of the experiments are presented in Fig. 13. They show a considerable decrease in the search time for xSE-ACAS thanks to the parallelization process. This time is of about 4.7 s for keywords distributed over 12000 documents. It decreases linearly with the increase of the cluster nodes.

8.2.2. *xSE-ACAS performance based on CP-ABE*

We conducted for this experiment the same tests as those carried out in section 8.2.1 for KP-ABE, but considering, this time, the CP-ABE variant. The experiment results are presented in Fig. 14. They are similar to those obtained with the KP-ABE variant regarding the considerable improvement in the search time using parallelization. However, the search time for SE-ACAS is better under CP-ABE relative to KP-ABE.

8.3. *xSE-ACAS performance based on KP-ABE and CP-ABE*

In this experiment, we analyze the performance of xSE-ACAS under KP-ABE and CP-ABE variants. To deal with this test, we augment the size of the search space. The used scenarios are presented in Table 9. We consider queries composed of five keywords.

The search time curves for xSE-ACAS according to the two ABE variants are shown in Fig. 15. Their progression is similar,

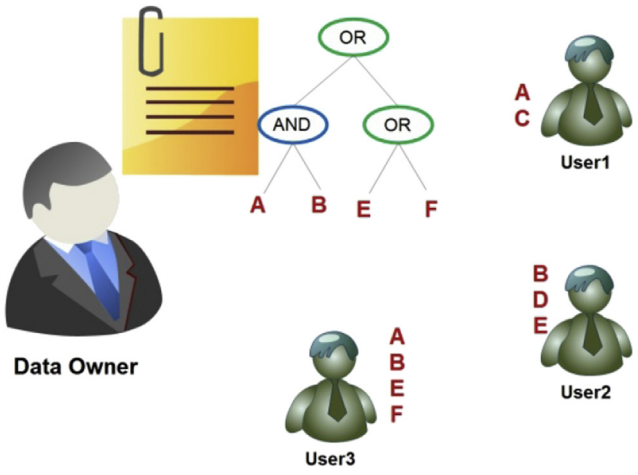

Fig. 12 – The experimental scenario for xSE-ACAS with CP-ABE.

Table 6 – Experimental data for security analysis of xSE-ACAS with CP-ABE.

Document	Attributes	w_1	w_2	w_3	w_4	w_5	w_6
d1	A *and* B	x	x		x		
d2	C *or* D *or* F		x		x	x	
d3	A *and* B *or* E	x		x			x
d4	C *or* E				x		
d5	A *or* E		x			x	
d6	A *and* F	x	x	x		x	x
d7	B *and* F		x		x		

Table 7 – Results of security experiments for xSE-ACAS with KP-ABE.

Search query	User	Search result
Q1 = "w_1 AND w_2 AND w_3"	User1	∅
	User2	∅
	User3	∅
Q2 = "w_2 AND w_4 OR w_5"	User1	{d2,d5}
	User2	{d2,d5}
	User3	{d1,d2,d5,d6}
Q3 = "w_1 AND w_3 OR w_6"	User1	∅
	User2	{d3}
	User3	{d3}
Q4 = "w_2 AND w_4 AND w_5"	User1	∅
	User2	{d2}
	User3	{d2}

except that the search time with CP-ABE is shorter than that with KP-ABE, thus electing xSE-ACAS based CP-ABE as a more efficient technique.

9. Conclusion

We proposed, in this article, a model of search on encrypted data for the Cloud Computing, whose main purpose is to enhance the access control on data. Integration of Searchable Encryption with access control mechanisms ensures, on one hand, confidentiality of data at both Cloud and user sides and limits, on the other hand, data access only to authorized users. However, this hybridization failed to satisfy the ACAS property.

To remedy this, we proposed to move the access control mechanism to the side of the Cloud, thus providing a new xSE-ACAS model, which reduces, furthermore, the decryption time by parallelization over the HPC cluster. xSE-ACAS has also been conceived to take advantage of HPC in order to speed up users' requests. To achieve this, the SSE-1 search method has been revised to support parallelization. The performance study showed a considerable improvement of the search time.

xSE-ACAS also integrates SSE-1 with two ABE techniques, namely, Key-Policy Attribute Based Encryption (KP-ABE) and Ciphertext-Policy Attribute Based Encryption (CP-ABE). A solution analysis highlighted that CP-ABE is well designed to identify undesirable access relative to KP-ABE. Furthermore, experiments showed that the search time with CP-ABE is better than that provided with KP-ABE. Finally, xSE-ACAS

Table 8 – Query keywords distribution for xSE-ACAS and SE-ACAS.

Scenario	Occurence of w1	Occurence of w2	Occurence of w3	Occurence of w4	Occurence of w5
Scenario1	50	45	40	35	30
Scenario2	100	90	80	70	60
Scenario3	200	180	160	140	120
Scenario4	300	270	240	210	180
Scenario5	500	450	400	350	300
Scenario6	700	630	560	490	420
Scenario7	1000	900	800	700	600
Scenario8	1500	1350	1200	1050	900
Scenario9	2000	1800	1600	1400	1200
Scenario10	2500	2250	2000	1750	1500
Scenario11	3000	2700	2400	2100	1800

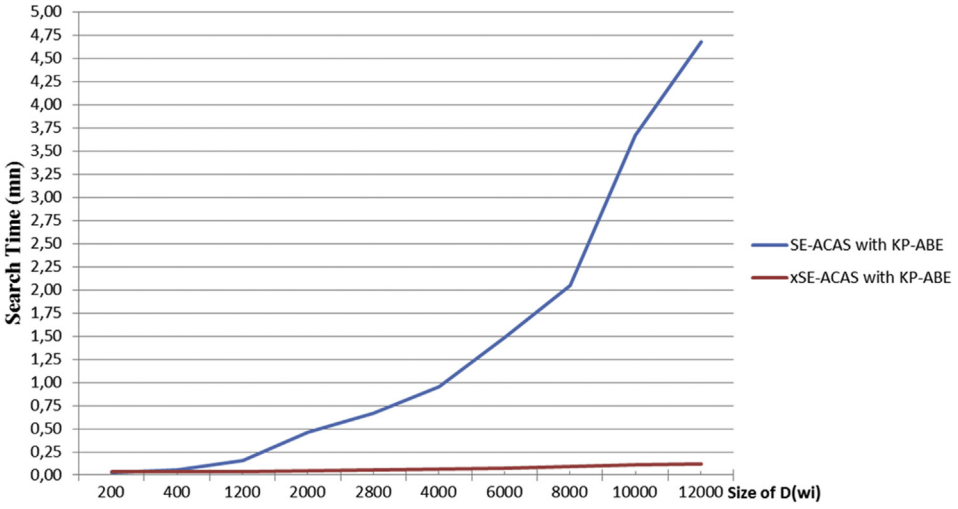

Fig. 13 – Comparison between xSE-ACAS and SE-ACAS under KP-ABE.

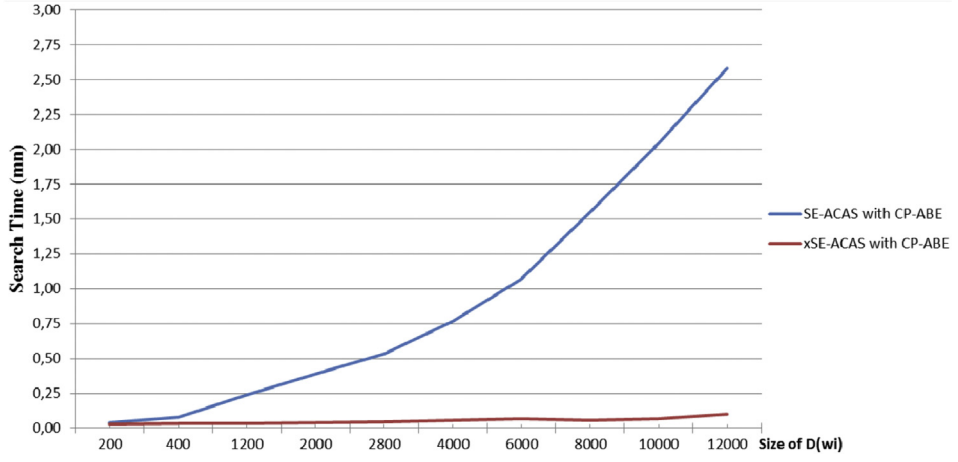

Fig. 14 – Comparison between xSE-ACAS and SE-ACAS under CP-ABE.

Table 9 – Query keywords distribution for xSE-ACAS.

Scenario	Occurrence of w1	Occurrence of w2	Occurrence of w3	Occurrence of w4	Occurrence of w5
Scenario1	50	45	40	35	30
Scenario2	100	90	80	70	60
Scenario3	200	180	160	140	120
Scenario4	300	270	240	210	180
Scenario5	500	450	400	350	300
Scenario6	700	630	560	490	420
Scenario7	1000	900	800	700	600
Scenario8	3000	2700	2400	2100	1800
Scenario9	5000	4500	4000	3500	3000
Scenario10	7000	6300	5600	4900	4200
Scenario11	10 000	9000	8000	7000	6000
Scenario12	30 000	27 000	24 000	21 000	18 000
Scenario13	50 000	45 000	40 000	35 000	30 000
Scenario14	70 000	63 000	56 000	49 000	42 000
Scenario15	100 000	90 000	80 000	70 000	60 000
Scenario16	130 000	117 000	104 000	91 000	78 000
Scenario17	150 000	135 000	120 000	105 000	90 000

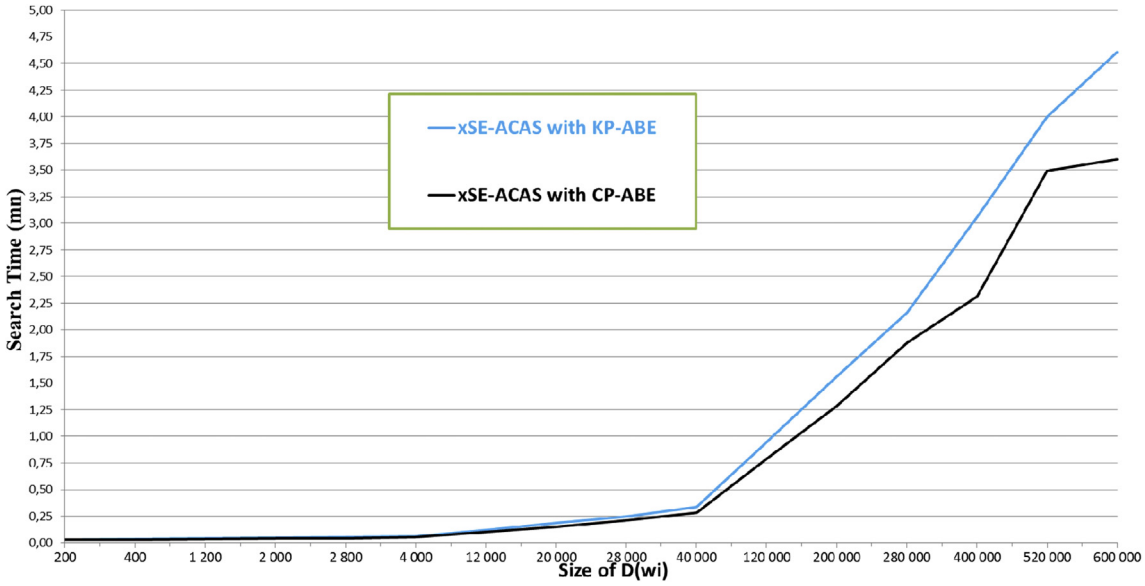

Fig. 15 – Search time comparison between KP-ABE and CP-ABE.

allows users to express multi-keywords search queries. We, currently, are working on enhancing the parallelization process in order to obtain better search times.

REFERENCES

Advanced Crypto Software Collection, http://hms.isi.jhu.edu/acsc/cpabe.

Bethencourt J, Sahai A, Waters B. Ciphertext-policy attribute-based encryption. In: IEEE symposium on security and privacy, 2007. SP '07; 2007. p. 321–34.

Boneh D, Crescenzo GD, Ostrovsky R, Persiano G. Public key encryption with keyword search. In: Cachin C, Camenisch JL, editors. advances in cryptology – EUROCRYPT 2004. Berlin Heidelberg: Springer; 2004. p. 506–22.

Buyya R, Broberg J, Goscinski AM. Cloud computing: principles and paradigms. John Wiley & Sons; 2010.

Cheng Y, Ren J, Wang Z, Mei S, Zhou J. Attributes Union in CP-abe algorithm for large Universe cryptographic access control. In: 2012 second international conference on cloud and green computing (CGC); 2012. p. 180–6.

Curtmola R, Garay J, Kamara S, Ostrovsky R. Searchable symmetric encryption: improved definitions and efficient constructions. In: Proceedings of the 13th ACM conference on computer and communications security. New York, NY, USA: ACM; 2006. p. 79–88.

Goh E-J. Secure indexes. 2003.

Goyal V, Pandey O, Sahai A, Waters B. Attribute-based encryption for Fine-grained access control of encrypted data. In: Proceedings of the 13th ACM conference on computer and communications security. New York, NY, USA: ACM; 2006. p. 89–98.

Matthew Green, JosepH ayo akinyele: https://code.google.com/p/libfenc.

Han F, Qin J, Zhao H, Hu J. A general transformation from KP-ABE to searchable encryption. Future Gener Comput Syst (FGCS), Elsevier 2014;30:107–15.
Ibraimi L, Asim M, Petkovic M. Secure management of personal health records by applying attribute-based encryption. In: 2009 6th international workshop on wearable micro and nano technologies for personalized health (pHealth); 2009. p. 71–4.
Joseph Ayo Akinyele, Gary Belvin, Christina Garman, Matthew Pagano, Michael Rushanan, Paul Martin, Ian Miers, Matthew Green, Avi Rubin: http://www.charm-crypto.com.
Kaci A, Bouabana-Tebibel T. Access control Reinforcement over searchable encryption. In: The 15th IEEE international conference on information reuse and integration – IEEE IRI 2014; 2014. San Francisco, USA.
Kaci A, Bouabana-Tebibel T, Challal Z. Access control aware search on the cloud computing. In: The third international conference on advances in computing, communication and informatics – ICACCI 2014; 2014. New Delhi, India.
Kamara S, Lauter K. Cryptographic cloud storage. In: Proceedings of the 14th international conference on financial cryptograpy and data security. Berlin, Heidelberg: Springer-Verlag; 2010. p. 136–49.
Lounis A, Hadjidj A, Bouabdallah A, Challal Y. Secure medical architecture on the cloud using wireless sensor networks for emergency management. In: 2013 eighth international conference on broadband and wireless computing, communication and applications (BWCCA); 2013. p. 248–52.
Malek B, Miri A. Combining attribute-based and access systems. In: International conference on computational science and engineering, 2009. CSE' 09; 2009. p. 305–12.
Rong C, Nguyen ST, Jaatun MG. Beyond lightning: a survey on security challenges in cloud computing. Comput Electr Eng 2013;39:47–54.
Ruj S, Stojmenovic M, Nayak A. Privacy preserving access control with authentication for securing data in clouds. In: 2012 12th IEEE/ACM international symposium on cluster, cloud and grid computing (CCGrid); 2012. p. 556–63.
Sahai A, Waters B. Fuzzy identity-based encryption. In: Proceedings of the 24th annual international conference on theory and applications of cryptographic techniques. Berlin, Heidelberg (: Springer-Verlag; 2005. p. 457–73.
Shamir A. Identity-based cryptosystems and signature schemes. In: Blakley GR, Chaum D, editors. Advances in cryptology. Berlin Heidelberg: Springer; 1985. p. 47–53.
Si X, Wang P, Zhang L. KP-abe based verifiable cloud access control scheme. In: 2013 12th IEEE international conference on trust, security and privacy in computing and communications (TrustCom); 2013. p. 34–41.
Singh A, Srivatsa M, Liu L. Search-as-a-service: outsourced search over outsourced storage. ACM Trans Web 2009;3(13):1–13. 33.
Song DX, Wagner D, Perrig A. Practical techniques for searches on encrypted data. In: 2000 IEEE symposium on security and privacy, 2000. S P 2000. proceedings; 2000. p. 44–55.
Tang Q. Search in encrypted data: theoretical models and practical applications, theory and practice of cryptography solutions for secure information systems. IGI; 2013. p. 84–108.
Yin C, Zhang R. Access control for the smart meters based on abe. In: 2011 international conference on cyber-enabled distributed computing and knowledge discovery (CyberC); 2011. p. 79–82.
Yu S, Ren K, Lou W. Attribute-based content distribution with hidden policy. In: 4th workshop on secure network protocols, 2008. NPSec 2008; 2008. p. 39–44.

Thouraya Bouabana-Tebibel received a Ph.D. in Computer Science from USTHB University (Algeria) in collaboration with Pierre & Marie Curie University (France) in 2007. She has been an engineer/researcher for eight years and is now a full professor of computer science at Ecole Nationale Supérieure d'Informatique in Algeria. She also is a researcher director at Laboratoire de Communication dans les Systèmes Informatiques. She has successfully conducted numerous research projects and supervised BS, MS, and PhD theses. She has over 80 refereed publications and a book edited by Editions Universitaires Européennes. Her research interests include formal methods, network security and information reuse and integration.

Abdellah Kaciis an Assistant Professor of Computer Science at Ecole Nationale Supérieure de Technologie (ENST) in Algiers. He received a BS degree from Université des Sciences et de la Technologie Houari Boumediène (USTHB), and a Master degree from Ecole nationale Supérieure d'Informatique (ESI) in Algiers, where he is currently a PhD student. His research interests are on the Cloud Computing, Information Security, and Cryptography.