The ISC Int'l Journal of Information Security
ISeCure

November 2022, Volume 14, Number 3 (pp. 135–149)
http://www.isecure-journal.org

Presented at the ISCISC'2022 in Rasht, Iran.

Attribute-Based Encryption with Efficient Attribute Revocation, Decryption Outsourcing, and Multi-Keyword Searching in Cloud Storage ☆

Sajjad Palanki [1], and Alireza Shafieinejad [1,*]
[1] *Department of Electrical and Computer Engineering, Tarbiat Modares University, Tehran, Iran.*

ARTICLE INFO.

Keywords:
Secure Cloud Storage,
Attribute-Based Encryption,
Attribute Revocation,
Multi-Keyword Searching

Type:
Research Article

doi:
10.22042/isecure.2022.14.3.14

dor:
20.1001.1.20082045.2022.14.3.14.8

ABSTRACT

Reliable access control is a major challenge of cloud storage services. This paper presents a cloud-based file-sharing architecture with ciphertext-policy attribute-based encryption (CP-ABE) access control mechanism. In CP-ABE, the data owner can specify the ciphertext access structure, and if the user key satisfies this access structure, the user can decrypt the ciphertext. The trusted authority embeds the private key of each attribute in a so-called attribute access polynomial and stores its coefficients publicly on the cloud. By means of the access polynomial, each authorized user will be able to retrieve the private key of the attribute by using her/his owned pre-shard key. In contrast, the data owner encrypts the file with a randomly selected key, namely the cipher key. The data owner encrypts the cipher key by CP-ABE scheme with the desired policies. Further, the data owner can create a different polynomial called query access polynomial for multi-keyword searching. Finally, the data owner places the encrypted file along the encrypted cipher key and query access polynomial in the cloud. The proposed scheme supports fast attribute revocation using updating the corresponding access polynomial and re-encrypting the affected cipher keys by the cloud server. Moreover, most of the calculations at the decryption and searching phases are outsourced to the cloud server, thereby allowing the lightweight nodes with limited resources to act as data users. Our analysis shows that the proposed scheme is both secure and efficient.

1 Introduction

Cloud storage for data sharing is an interesting and applied service, wherein access control to the shared elements is a major challenge. Users usually do not trust cloud servers due to the threats targeting user privacy, data integrity and/or confidentiality. In the design of a security protocol, it is usually

* Corresponding author.
☆ The ISCISC'2022 program committee effort is highly acknowledged for reviewing this paper.
Email addresses: s.palanki@modares.ac.ir, shafieinejad@modares.ac.ir
ISSN: 2008-2045

assumed that the cloud server is semi-honest-but-curious [1]. That is, the cloud server executes the protocol honestly, but at the same time, it is seeking to discover the connections and leakage of information and to analyze encrypted information based on the uploaded data. Further, it may act selfishly to save its computation and/or downloading bandwidth. Thus, access control cannot be assigned to the cloud server, mainly because it is assumed that it is a public and untrusted element. The major solution for the security of data sharing is to use encryption algorithms to prevent unauthorized access to classified information. However, data encryption has its challenges such as key management and distribution, access revocation and searching capability. The revocation of a key is required when user access is expired. Another problem is the ability to search in the encrypted data, which is either impossible or difficult to implement. Each algorithm that tries to resolve or mitigate the effect of these problems, depending on the design, can increase the system complexity at the data owner, the data user or the cloud server side. In this paper, we discuss the following set of issues:

- Efficient attribute revocation
- Multi-keyword searching among the encrypted files
- Outsourcing of decryption and searching to the cloud server

1.1 Related Work

Attribute-based encryption (ABE), which is a generalization of an ID-based encryption scheme, provides an access control mechanism at the algorithm level [2]. In cloud storage services, it can provide a basis for applying either ethical or legal restrictions on access to information such as an age verification policy. Further, access granted based on attributes can improve user anonymity and accordingly preserves the privacy of the user at a higher level than the access granted based on the identifier.

A major shortcoming of the primary schemes is revocation which for ABE systems refers to either user revocation or attribute revocation. The former stands for a user who is completely removed from the whole system. While the latter points to a user who loses some attributes and thereby misses some part of his/her access privileges, but he remains in the system as an active entity. Thus, attribute-level revocation provides more fine-grained and flexible access control than user-level revocation. Throughout this paper, we focus on attribute revocation. Wang *et al.* [3] proposed the hierarchical attribute-based encryption (HABE) model by combining a hierarchical identity-based encryption (HIBE) system and a ciphertext-policy attribute-based encryption (CP-ABE) system. It provides fine-grained access control and supports a scalable attribute revocation scheme.

In [4], the authors proposed a CP-ABE access control mechanism with efficient attribute and user revocation capability. It is achieved by dual encryption with the benefits of attribute-based encryption and selective group key distribution in each attribute group. However, maintenance of the keys in this scheme is a costly operation.

Zu *et al.* [5] proposed a CP-ABE scheme with fast attribute revocation. The master key is randomly divided into two parts: the secret key and the delegation key, which are sent to the user and the cloud service provider, respectively. The former is used in the attribute revocation process while the latter is used by the user to extract the secret key. The scheme has a lower storage overhead and communication cost.

Zheng *et al.* [6] proposed a verifiable attribute-based keyword search (VABKS). It enables a data user to search over the encrypted data, outsource search operations to the cloud, and verify whether the cloud has faithfully executed the search operations. Fan *et al.* [7] proposed an attribute-based encryption scheme that supports multi-keyword searching. Further, it allows multiple data owners to enhance access control over encrypted files.

Li *et al.* [8] proposed an ABE scheme that supports multi-keyword searching. It allows encrypted keywords to be inquired correctly by multiple users if they have a series of attributes that satisfy the access structure. Note that a common point of the schemes in [6–8] is that none of them supports attribute revocation.

Wang *et al.* [9] proposed an attribute-based encryption scheme supporting multi-keyword search and attribute revocation. Although the scheme supports attribute revocation, it imposes computational burden and communication overhead on both attribute authority and key generation server (KGS) for updating the secret key of non-revoked users.

Outsourcing some parts of the user process to the cloud server makes the computational overhead on the user side simple and constant. It becomes an important feature when major parts of the system on the user side are resource-limited devices. Since searching and decryption are the most frequent operations on the user side, outsourcing some part of decryption or searching to the cloud server has the potential impact on the efficiency of the system.

The schemes in [6–9] support multi-keyword searching on encrypted data by the cloud server. On the

other side, the schemes in [10–12] support outsourcing of decryption.

Liu *et al.* [10] proposed an attribute-based encryption that supports decryption outsourcing and attributes revocation. Xia *et al.* [11] proposed an attribute-based access control scheme with attribute revocation capability in cloud computing. The scheme employs an additional entity, a so-called access controller, which is responsible for assisting the attribute authority to generate the user's attributes. Further, the access controller is involved in decryption by generating an access token. Using expiring the access token, the scheme revokes an attribute from a specific user. Liu *et al.* [12] proposed an ABE scheme supporting decryption outsourcing in addition to attribute revocation and policy updating. Updating the access policy for encrypted data is suitable when the user's attributes are frequently changed or the data owner updates access control.

Miao *et al.* [13] proposed an ABE scheme for personal health records which supports multi-keyword searching in the multi-owner setting. Multi-owner points to a system where encrypted data is shared between multiple data owners.

Multi-authority is another feature in the ABE scheme which is a proper solution for cloud storage systems in which user attributes are issued by different authorities. Both [12] and [14] support multiple attribute authorities.

In addition to multi-authority, the scheme in [14] supports user revocation and hidden policy. The hidden policy states that the scheme protects the privacy of the access policy in addition to data confidentiality. Further, the authors in [15] proposed a CP-ABE scheme supporting hidden policy in addition to fast keyword search.

In [16], the authors presented a user collision avoidance CP-ABE scheme with efficient attribute revocation. The attribute revocation is implemented using the concept of an attribute group. When an attribute is revoked from a user, the group manager updates the secret keys of other users. The encryption policy in ABE systems is either a ciphertext policy or a key policy. The work in [17] proposed a key-policy ABE, KP-ABE, which puts the access structure in the key instead of ciphertext (CP-ABE). Further, it supports policy updating.

In [18], the authors proposed PU-ABE, a variant of CP-ABE supporting efficient access policy updating that captures attributes addition and revocation to access policies. The ciphertexts received by the end-user are constantly sized and independent of the number of attributes used in the access policy. But, the computational and communication overhead depend on the number of attributes in the access policy. Upon updating the access policy, the ciphertext is re-encrypted by the data owner.

Xue *et al.* [19] proposed a CP-ABE access control algorithm to prevent the Economic Denial of Sustainability (EDOS) attacks in cloud storage wherein a malicious attacker can download thousands of files. In this scheme, users are first authenticated by the cloud provider before sending any download request. The authorized users confirm the resource consumption for this download to the cloud provider.

1.2 Contribution of the Work

The contribution of our work can be summarized as:

- We embed the private key of each attribute in an attribute access polynomial, thereby implementing attribute revocation by updating the access polynomial which is efficient in terms of computational and communication overhead imposed on the attribute authority and the data user.
- The proposed scheme supports multi-keyword searching by embedding the desired keywords of each file in a query access polynomial. The cloud server efficiently performs the search by simple hash and exponentiation operations instead of bilinear pairing operations.
- Most of the computations are outsourced to the cloud server, including decryption, searching and re-encryption of the cipher keys, which can greatly reduce the cost of decryption and searching on the user side.

This article is organized as follows. In Section 2, the required background is presented. In the Section 3, the concrete construction of the proposed scheme is explained. In Section 4, the security and performance of our scheme are discussed. In Section 5, the proposed scheme is compared with the current state-of-the-art schemes. In the Section 6, we present the summary and conclusion of the paper.

2 Preliminaries

2.1 Symbol Definition

Table 1 summarizes the notations and symbols used in the proposed scheme. Let G_0 and G_t denote two multiplicative cyclic groups with prime order p and g are the generator of G_0 and $e : G_0 \times G_0 \rightarrow G_t$ is a bilinear mapping. In the two-way mapping, for all $u, v \in G_0$ and $a, b \in Z_p$, we have : $e(u^a, v^b) = e(u, v)^{ab}$ where $e(g, g) \neq 1$. Moreover, for any $i \in Z_p$, the Lagrange coefficient is defined as: $\Delta_{i,L}(x) = \prod_{i \in L, l \neq i} \frac{x-l}{i-l}$.

2.2 Access Structure and Access Tree

The access structure is used for the access policy. Let $A_1, A_2, \ldots, A_n$ be a set of attributes. A monotone access structure is a set A of non-empty subsets of $\{A_1, A_2, \ldots, A_n\}$, i.e., $A \subseteq 2^{\{A_1, A_2, \ldots, A_n\}}$. The sets that belong to A are called authorized attribute sets, and the ones that are not in A are called unauthorized attribute sets. It is useful to represent the access structure in a tree denoted by Γ. Each non-leaf node of Γ acts as a threshold gate. Let $N(x)$ and k_x respectively denote the number of children of node x and its threshold value where $0 < k_x \leq N(x)$. When $k_x = 1$, the threshold gate acts as an OR gate while it becomes an AND gate when $k_x = N(x)$. Against, each leaf node x of the access tree corresponds to an attribute.

The parent of node x is denoted by $parent(x)$. Further, $att(x)$ refers to the attribute that is associated with the leaf node x in the tree. Moreover, there is an ordering between the children of every node, i.e., they are respectively numbered from 1 to $N(x)$. The $index(x)$ returns the number associated with the node x. Note that the index values are uniquely assigned to the children in the access structure for a given key in an arbitrary manner.

2.3 Key Distribution Based on Access Polynomial

Access polynomials were first introduced in [4] for secure intra-group communication. The polynomial allows group members to recover the encryption key. In summary, the key management includes the following steps:

- Initially, the group controller sends the pre-shared keys $K_1, K_2, \ldots, K_n$ respectively to the $1^{th}, 2^{th}, \ldots, n^{th}$ group member via a secure channel.
- The controller then generates the following polynomial using pre-shared keys and the master key K_C. Then, he/she computes the coefficients $(a_0, a_1, \ldots, a_{n-1})$ and broadcasts them to all of the group members:

$$f(x) = K_C + \prod_{i=1}^{n}(x - K_i) = a_0 + a_1 x + \cdots + a_{n-1}x^{n-1} + x^n \quad (1)$$

- Upon receiving the polynomial coefficients, the i_{th} member of group can retrieve the master key by $K_C = f(K_i)$.

Although the authors claimed that the scheme has forward and backward secrecy, it lacks both of them. The main weaknesses are: 1) The member who has left the group, can discover the new key of the group and 2) The new member can retrieve the previous key of the group [20]. Another vulnerability is raised when a member who can factorize the polynomial $f(x) - f(K_i)$, can retrieve the pre-shared key of the other users and thus can compute the master key after leaving the group. An effective countermeasure scheme is to use a type of hash-based polynomial [21]. In this paper, we use a version of a hash-based polynomial along with a random number as a seed. Precisely, our access polynomial is defined as:

$$f(x) = K_C + \prod_{i=1}^{n}(x - H(K_i, r)) \quad (2)$$

Where H is a one-way cryptographic hash function and r is a random value that is forcefully updated upon changing the polynomial. Further, to act in a finite group, all multiplications and additions are performed modulo a prime number.

Table 1. Symbols Definition

Symbols	Description
n	Number of data users
m	Number of attributes
SK	Secret Key
TK	Could decryption key
MSK	Master secret key
PK	Public key of the scheme
H	Hash function
Γ	Access Structure
ck	Cipher-key which is used for file encryption
r_q	Query-key which is used in query access polynomial
U_i	The identifier of the ith user
L	Set of all attributes
K_i	The pre-shared key of the ith user
a_j	jth attribute ($j \leq m$)
v_j	Private key of the attribute a_j
PK_j	Public key of the attribute a_j
e	Bilinear map
G_0	Bilinear group
p	Big prime number
g	Generator of group G_0
Z_p	A finite field of numbers $\{0, 1, \ldots, p-1\}$
r_j	A random attribute which is used in attribute access polynomial c
$f^j(x, r_j)$	Attribute access polynomial for jth attribute
$g(x, r_q)$	Query access polynomial
N_j	Number of users having attribute j

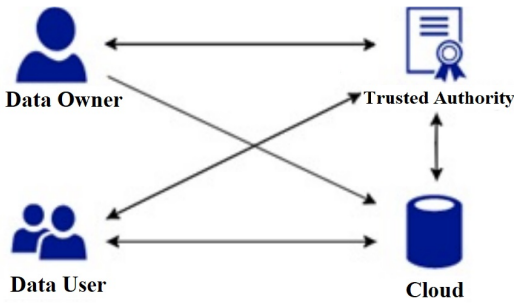

Figure 1. The overall architecture

3 Proposed Scheme

In this section, we describe the overall framework of our scheme and the construction of the solution.

3.1 A. The System Model

The overall architecture of the proposed scheme, shown in Figure 1, consists of the following components:

- The data owner encrypts a file and shares it by putting it on cloud storage.
- The data user who searches through the encrypted files by query keywords, selects one of them, downloads and decrypts it.
- The cloud server where the data owner puts the encrypted files. Further, most of the computation in decryption and searching is outsourced to the cloud server. Moreover, it performs re-encryption of the cipher key in the case of attribute revocation.
- The trusted authority is responsible for creating keys corresponding to each attribute and distributing them among the users. Further, it performs attribute revocation.

We assume that each data user has a pre-shared key with the trusted authority. It means that the trusted authority, through a secure channel, negotiates and establishes keys $K_1, K_2, \ldots, K_n$ to the first, second, ..., and n^{th} data user, respectively. The construction consists of the following phases:

- Initialization
- Secret key generation
- Encryption
- Pre-computation
- Searching
- Decryption
- Attribute revocation

3.2 Initialization Phase

The initialization algorithm gets the maximum number of attributes in the system, namely $|L|$, as input. Then it chooses a bilinear group G_0 of prime order p with generator g. Moreover, two elements $\alpha, \beta \in Z_p$ are randomly selected. Then, for each attribute a_j, the trusted authority selects a random value $v_j \in Z_p$ as the private key of the attribute. The public key of attribute a_j, denoted by PK_j, is computed as:

$$PK_j = g^{v_j} \tag{3}$$

For each attribute public key, ver_j is an integer variable that represents the version of the attribute. The version is initialized to 1 and incremented by one unit upon the generation of new private/public key pairs as the result of attribute revocation. The public key and master secret key of the scheme are defined as:

$$PK = \{G_0, g, h = g^{\beta}, e(g,g)^{\alpha}, (PK_1, ver_1), (PK_2, ver_2), \ldots, (PK_m, ver_m)\} \tag{4}$$

$$MSK = \{\beta, g^{\alpha}, v_1, v_2, \ldots, v_m\} \tag{5}$$

The public key is put on the cloud server as publicly accessible data. In addition to the public key, the trusted authority generates an attribute polynomial for each attribute. It is worth mentioning that, at the beginning of the process, the polynomials are null since no attributes are yet assigned to users. Actually, by assigning a specific attribute to some users during the secret key generation phase, its access polynomial is formed and grew up.

3.3 Secret Key Generation Phase

$KeyGen(U_i, S, MSK) \rightarrow SK$ The algorithm which is run by the trusted authority receives the identifier of the user, the set of attributes S requested by the user and the master secret key MSK as input. In this phase two actions are taken place: 1) Generation of the secret key and 2) Updating the corresponding attribute access polynomials. For the first time that a user requests, the former action is done. The trusted authority randomly generates a pair of r and $z \in Z_p$ which are unique for each system user and computes D as follows:

$$D = g^{\frac{\alpha + r}{\beta}} \tag{6}$$

Then, the trusted authority sends the secret key $SK = (z, D)$ to the system user. For the succeeding requests of this user, the former action is ignored. In our scheme, the trusted authority does not send any attribute key directly to the user. Indeed, the attribute keys are embedded in the attribute access polynomial which is put on the cloud server. For each user with distinct r and z private part of attribute a_j is defined as $g^{\frac{r}{v_j z}}$. Note that, among the parameters of this term, r and v_j are unknown to the user and only z which is sent to the user is known to him/her. This feature makes the system resistant to collusion attacks by prohibiting the usage of another user's attribute. Now, for the latter

action, the trusted authority creates polynomial coefficients of each requested attribute in S. For each attribute $a_j \in S, N_j$ is incremented by one. Further, assume that the pre-shared key of the current user and the users who have previously gotten a_j are respectively denoted by $(K_1, K_2, \ldots, K_{N_j})$. Moreover, suppose the value of $g^{\frac{r}{v_j z}}$ for these users are respectively be as $y_1, y_2, \ldots, y_{N_j}$. The access polynomial for attribute a_j is defined as Lagrange interpolation for the set of points $(H(K_1, r_j), y_1), (H(K_2, r_j), y_2), \ldots, (H(K_{N_j}, r_j), y_{N_j})$. It is done in two stages. First, the polynomial basis $L_i(x)$ for the user U_i is defined as:

$$L_i(x) = \prod_{1 \le t \le N_j, t \ne i} \frac{x - H(K_t, r_j)}{H(K_i, r_j) - H(K_t, r_j)} (mod\ p) \tag{7}$$

Finally, the attribute access polynomial for these users through the corresponding values $y_1, y_2, \ldots, y_{N_j}$ is defined by the linear combination of basis polynomials as:

$$\begin{aligned} &f^j(x, r_j) = \\ &y_1 L_1(x) + y_2 L_2(x) + \cdots + y_{N_j} L_{N_j}(x) (mod\ p) = \\ &\quad s_0 + s_1 x + s_2 x^2 + \cdots + s_{N_j - 1} x^{N_j - 1} (mod\ p) \end{aligned} \tag{8}$$

Where H is a one-way hash function and $r_j \in Z_p$ is a random value. Note that r_j acts as a salt to blind the factor of a single user in different attributes. Further, all the computations are done in modulo p. Thus, the attribute polynomial f^j is completely identified by polynomial coefficients s_0 to s_{N_j}. Each polynomial f^j denoted by 3-tuples $(a_j, ver_j, (s_0, s_1, s_2, \ldots, s_{N_j - 1}, r_j))$ is stored publicly accessible in the cloud storage in addition to the public key of the system. For the first time that accesses polynomial is created by only a single user, it becomes a constant polynomial as $f^j(x, r_j) = g^{\frac{r}{v_j z}}$. It reveals the secret key of the target user. To fix this problem, we can initialize the access polynomial be a set of dummy users, e.g., five users to avoid finding or guessing the secret key of the intended users. After receiving SK, the user U_i is capable of recovering the private key of each granted attribute which is required for the decryption/search phases. Assume that the user requests the attribute a_j. After reception of $SK = (z, D)$, he/she downloads the attribute access polynomial f^j from the cloud and does the following calculations:

$$D_j = f^j(H(K_i, r_j)) = g^{\frac{r}{z v_j}} (mod\ p) \tag{9}$$

It means that f^j is computed at the point $H(K_i, r_j)$. Similarly, the user can obtain a unique D_j for each own attribute $a_j \in S$. Now, the user creates Transform key TK as:

$$TK = \{g^{(\alpha + r)/\beta}, (D_j, ver_j) : \forall a_j \in S\} \tag{10}$$

We will use TK in both the decryption and search phases.

3.4 Encryption Phase

In this phase, the instructions that the data owner must follow to encrypt her/his file M with a specific policy are described. The process includes the following steps:

- The data owner receives the latest version of the system public key from the cloud server.
- The data owner randomly generates a pair of keys $ck, r_q \in Z_p$. The former is a cipher key used for encryption of the content of M. The latter is query-key which is used to encrypt the keywords extracted from the M.
- The file M is encrypted with ck and a symmetric encryption algorithm like AES. The encrypted file is denoted by $E_{ck}(M)$.

  The data owner first selects a polynomial q_x for each node x in the access tree Γ from the root and continues in a top-down manner until reaching to leaf nodes. For each node x in Γ, the degree of polynomial q_x is $k_x - 1$. Starting from the root R, the algorithm selects a random $s \in Z_p$ and sets $q_R(0) = s$ and then it randomly chooses $k_R - 1$ other points of q_R to completely define it. For any node $x \in \Gamma$, it sets $q_x(0) = q_{parent(x)}(index(x))$ and selects $k_x - 1$ other points randomly to completely define q_x. In Γ, let X be the set of attributes associated with the leaf nodes. The algorithm creates the ciphertext CT as:

$$\begin{aligned} CT = (\hat{C} = ck.e(g, g)^{\alpha}, C = h^s, \\ C_j = PK_j^{q_x(0)} : \forall a_j = att(x) \in X) \end{aligned} \tag{11}$$

- The data owner determines a set of keywords associated with the content of the file M. Suppose $W = w_1, w_2, \ldots, w_u$ denotes the set of chosen keywords. The data owner generates the query access polynomial of M using keywords in W and query-key r_q, as:

$$\begin{aligned} Q(x, r_q) &= (x - H(w_1, r_q))(x - H(w_2, r_q)) \cdots (x - H(w_u, r_q)) \\ &= t_0 + t_1 x + t_2 x^2 + \cdots + t_{u-1} x^{u-1} + x^u \end{aligned} \tag{12}$$

Note that the degree of the polynomial is identified by the number of keywords in W. This step is optional and can be bypassed if the data owner decides to disable multi-keyword searching on the encrypted file. Finally, the data owner appends $\hat{(R)}$ and $(t_0, t_1, \ldots, t_{u-1})$ to CT as the following five-tuple:

$$CT = (\hat{C} = ck.e(g,g)^{\alpha}, C = h^s, C_j = PK_j^{q_x(0)} : \forall a_j = att(x) \in X, Index = (t_0, t_1, \ldots, t_{u-1})) \quad (13)$$

Now, the data owner stores CT along with $E_{ck}(M)$ in the cloud.

3.5 Pre-Computation Phase

$Transform(CT, TK) \rightarrow TQ \; or \; null$

This is a prerequisite phase for both the searching and decryption phases. During this phase, partial decryption of both cipher-key and query-key is performed by the cloud server. It consists of two steps: 1) Updating the transform key which is performed by the data user and 2) Partial decryption which is done by the cloud server.

3.5.1 Updating Transform Key By The Data User

This step forces the data user to get the latest version of his/her attributes. For each attribute, the data user checks the current version of his/her local D_j with the version of a_j on the cloud. If the cloud server has a newer version, as the result of attribute revocation, the data user downloads new f^j from the cloud and then updates D_j according to Equation 9. Finally, TK, along with CT is sent to the cloud server.

3.5.2 Partial Decryption By The Cloud Server

The cloud server gets the ciphertext CT and D_j of user attributes, from the data user. Let SK' denote the set of D_j of data user attributes. The cloud server performs the partial decryption of ciphertext as the following operations: The decryption process is defined by $DecryptNode(CT, SK', x)$, a recursive algorithm with CT, SK' and node x of Γ as inputs. This algorithm is similar to [10]. For a leaf node x, assume $a_j = att(x)$. If $a_j \notin S$ then decryption is failed and we set $DecryptNode(CT, SK', x) = null$. If $a_j \in S$ then, it returns:

$$e(D_j, C_x) = e(g^{r/(v_j z)}, g^{v_j q_x(0)}) = e(g,g)^{rq_x(0)/z} \quad (14)$$

For a non-leaf node x, we call $DecryptNode$ for each child y of x. Suppose S_x denotes a set of child nodes that returns the non-null value for $DecryptNode$. If $k_x < |S_x|$, the decryption at node x is failed and it returns null. It means that the user attributes at node x are not sufficient since the number of nodes that return a non-null value is less than k_x, the threshold of node x. Otherwise, we can continue to decrypt. Let S_{k_x} denotes a desired subset of S_x with k_x nodes. The decryption at the node x can be done as:

$$\begin{aligned} &DecryptNode(CT, SK', x) \\ &= \prod_{y \in S_{k_x}} DecryptNode(CT, SK', y)^{\Delta_{j,T_x}(0)} \\ &= \prod_{y \in S_{k_x}} (e(g,g)^{rq_y(0)/z})^{\Delta_{j,T_x}(0)} \\ &= \prod_{y \in S_{k_x}} (e(g,g)^{rq_{parent(y)}(index(y))/z})^{\Delta_{j,T_x}(0)} \\ &= \prod_{y \in S_{k_x}} (e(g,g)^{rq_x(j)/z})^{\Delta_{j,T_x}(0)} \\ &= e(g,g)^{rq_x(0)/z} \end{aligned} \quad (15)$$

Where $j = index(y)$ and $T_x = \{index(y) : y \in S_{k_x}\}$. We call $DecryptNode$ at the root of the access structure. If the SK' satisfies the required attributes of Γ, the decryption algorithm is taken place. It leads to the following value :

$$A = DecryptNode(CT, TK', R) = e(g,g)^{rs/z} \quad (16)$$

Further, the cloud server computes B and Q respectively as:

$$B = \frac{\hat{C}}{e(C,D)} = \frac{ck.e(g,g)^{\alpha s}}{e(g^{(\alpha+r)/\beta}, h^s)} = \frac{ck}{e(g,g)^{rs}} \quad (17)$$

$$Q = \frac{\hat{R}}{e(C,D)} = \frac{r_q.e(g,g)^{\alpha s}}{e(g^{(\alpha+r)/\beta}, h^s)} = \frac{r_q}{e(g,g)^{rs}} \quad (18)$$

Finally, the cloud server returns $TQ = \{A, B, Q\}$ to the data user. Otherwise, if SK' does not satisfy the required attributes of Γ, the decryption is failed and the algorithm returns $null$.

3.6 Searching Phase

$Search(w_1, w_2, \ldots, w_v) \rightarrow (b_1, b_2, \ldots, b_v)$

This phase enables the data user to search multiple keywords through the encrypted file. The result of the search is an array of boolean values wherein each element respectively identifies the occurrences of the intended keyword. Using the TQ received in the pre-computation phase, the data user is capable of computing query-key r_q as:

$$Q.A^z = \frac{r_q.(e(g,g)^{rs/z})^z}{e(g,g)^{rs}} = r_q \quad (19)$$

Let, $W_Q = \{w_1, w_2, \ldots, w_v\}$ denotes the set of query keywords. The data user computes $WD = \{H(w_1, r_q), H(w_2, r_q), \ldots, H(w_v, r_q)\}$ from W_Q by r_q, and sends it to the cloud server. The cloud server regenerates the query access polynomial of the intended file by the coefficients of $t_0, t_1, \ldots, t_u$ as follows:

$$Q(x) = t_0 + t_1 x + t_2 x^2 + \cdots + t_{u-1} x^{u-1} + x^u \quad (20)$$

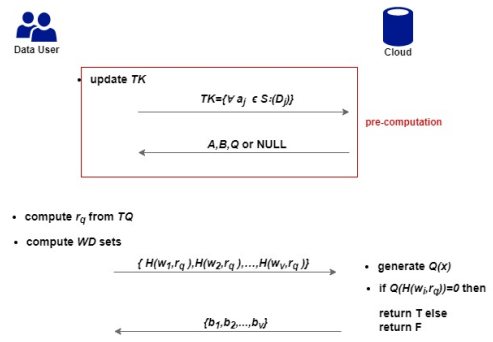

Figure 2. Searching Phase

Then, for each received value it checks the following condition:

$$Q(H(w_i, r_q)) == 0 \tag{21}$$

If the condition is met, the intended keyword exists in the file M. The cloud server sends an array of Boolean values as$\{b_1, b_2, \ldots, b_v\}$ to the data user where in b_i indicates the existence of w_i in the encrypted file. The details of the phase are shown in Figure 2. Since the algorithm takes the hash of each keyword concatenated by the random value r_q, the cloud server is not able to get any information on query keywords since r_q is unknown to the cloud server.

3.7 Decryption Phase

This phase enables a data user to decrypt an encrypted file if his/her attributes meet the access policy of the intended file. The complete process is shown in Figure 3. Like the searching phase, the data user must first perform the pre-computation phase to obtain TQ from the cloud server. If the access structure Γ is satisfied by the user's attributes, partial decryption of CT is successfully done and the cloud server returns TQ, otherwise, it returns $null$. Assume the cloud server returns TQ, the data user is now capable of recovering cipher-key ck by:

$$B.A^z = \frac{ck.(e(g,g)^{rs/z})^z}{e(g,g)^{rs}} = ck \tag{22}$$

Now, the data user downloads the encrypted file $E_{ck}(M)$ and simply decrypts it by ck to retrieve the plain file M. The searching phase enables the user to initially search for desired keywords in the encrypted file. If the result meets the user's request, then he/she runs the decryption phase to download and decrypt the target file. This makes the data user avoid unnecessary decryptions.

3.8 Attribute Revocation Phase

This algorithm takes an attribute and a set of users as input. It revokes a_j from the users in Ω. After a revocation, none of the users in Ω is capable of decrypting by a_j. Attribute revocation is one of the most challenging actions in the attribute-based encryption scheme. Our scheme implements attribute revocation efficiently in terms of the data owner computations, the messages sent between parties and re-encryption imposed on the cloud server. The attribute revocation consists of the following three steps:

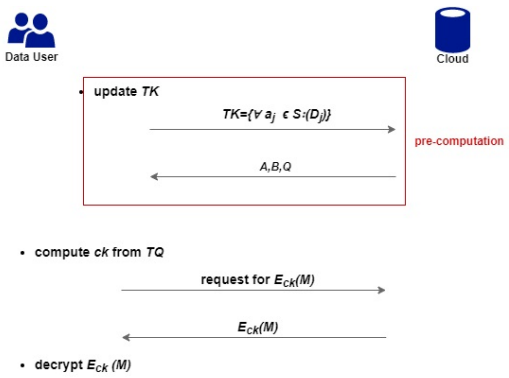

Figure 3. Decryption Phase

- Generating new attribute by the trusted authority
- Updating attribute access polynomial by the trusted authority
- Re-encryption of cipher keys by the cloud server

3.8.1 Generating New Attribute Private/Public Key

Suppose we want to revoke the attribute a_j. Further, assume that the private key of a_j is v_j and its version equals ver_j. Now, the trusted authority randomly selects a $v_j^* \in Z_p(v_j^* \neq v_j)$ as a new private key. Let U_j denote the modular division of the new key to the previous key as:

$$U_j = \frac{v_j^*}{v_j} \tag{23}$$

The trusted authority sends U_j to the cloud server. The cloud server updates the public key and the version of attribute a_j as:

$$PK_j^* = PK_j^{U_j} = g^{v_j^*} \tag{24}$$

$$ver_j^* = ver_j + 1 \tag{25}$$

3.8.2 Updating Attribute Access Polynomial

Assume that μ represents the set of users for whom the attribute has not been revoked. Thus, the new value of N_j^* will be equal to the size of μ, i.e., $N_j^* = |\mu|$. The trusted authority must enable these users to update the corresponding private key of attribute a_j. This is simply done by updating the attribute access polynomial of a_j. Precisely, the new polynomial is

reconstructed by pre-shared keys of the users in μ according to Equation 7 and Equation 8:

$$f^{j^*}(x, r_j{}^*) = s_0^* + s_1^* x + s_2^* x^2 + \cdots + s_{N_j^*-1}^* x^{N_j^*-1} \tag{26}$$

Where $r_j{}^* \in Z_p$ is a new random value to protect the system against factorization attacks. Finally, the trusted authority sends $(a_j^*, ver_j^*, (s_0^*, s_1^*, \ldots, s_{N_j^*-1}^*, r_j^*))$ to the cloud server to update the attribute access polynomial. By this polynomial, the users in μ are able to retrieve the private part of the new attribute, D_j, according to Equation 9. Against, each user for whom the attribute is revoked, is unable to compute D_j since her/his pre-shared key is excluded from the new attribute access polynomial. Unlike [10], in which attribute revocation imposes a highly intensive messaging overhead to the trusted authority to inform new attributes to the non-revoked users, it is enough in our scheme that the trusted authority updates the attribute access polynomial on the cloud server. On the other hand, when a non-revoked user becomes online, he/she is enforced to update her/his transform key TK by running before any decryption or searching. It means that the attribute updating is done on demand and initiated by the user.

3.8.3 Re-Encryption of the Cipher-Keys

This step which is done by the cloud server completes the puzzle of attribute revocation. Re-encryption of cipher keys prohibits the revoked user from recovering the cipher keys by the old attributes. Upon receiving U_j from the trusted authority, the cloud server updates the ciphertexts associated with the revoked attribute a_j. The re-encryption process can be completely done using U_j. The cloud server searches for the ciphertext which is encrypted by attribute a_j. Suppose C_{a_x} denotes a leaf node contains a_j, the cloud server updates these terms by exponentiating with U_j. Since:

$$C_{a_x}^* = (C_{a_x})^{U_j} = (g^{v_j q_x(0)})^{v_j^*/v_j} = g^{v_j^* q_x(0)} \tag{27}$$

The whole process for a ciphertext CT is formalized as:

$$CT^* = \{\hat{C}^* = \hat{C}, \hat{R}^* = \hat{R}, C^* = C, \\ \forall a_x = a_j : C_{a_x}^* = (C_{a_x})^{U_j}, \forall a_x \neq a_j : C_{a_x}^* = C_{a_x}\} \tag{28}$$

Note that the rest terms in which $a_x \neq a_j$ remain unchanged. Re-encryption of cipher keys makes the data owners get rid of file re-encryption which could be a very costly operation. It is worth mentioning that the server only knows U_j and accordingly, neither v_j^* nor v_j is known to the cloud server.

Algorithm 1. Computing the coefficients of access polynomial

GetPolynomialCoefficents$(K_1, K_2, \ldots, K_n, r)$

```
1  α_0 ← 1
2  α_1 ← −H(K_1, r) (mod p)
3  α_i ← 0 for 2 ≤ i ≤ n
4  for i ← 2 to n
5       do
6            β ← H(K_i, r) (mod p)
7            α_i ← 1
8            for j ← i − 1 downto 1
9                 do
10                     α_j ← α_{j−1} − β α_{j−1} (mod p)
11           α_0 ← −β α_0 (mod p)
12 return (α_0, α_1, ..., α_n)
```

3.9 Computations Related to Access Polynomial

The computation of access polynomial coefficients is a major operation in both secret key generation and attribute revocation phases for the trusted authority and keywords encryption for data user. Thus, its effective implementation has a crucial impact on the performance of the scheme and accordingly, on the system's scalability. Algorithm 1 briefly describes the computation of access polynomial coefficients. It is a primary operation for polynomial basis computation as well as computing query access polynomial during the encryption phase. It acts incrementally. Initially, the result is initialized by the $(x - h(K_1, r))$. At the i^{th} iteration of the outer loop, the current polynomial is multiplied by the binomial $(x - H(K_i, r))$. Thus, we must update the coefficients as:

$$\begin{aligned} F(x) &= (\alpha_0 + \alpha_1 x + \cdots + \alpha_{i-1} x^{i-1})(x - h(K_i, r)) \\ &= -\alpha_0 H(K_i, r) + (\alpha_0 - \alpha_1 H(K_i, r))x + \cdots + \\ &\quad (\alpha_{i-2} - \alpha_{i-1} H(K_i, r))x^{i-1} + \alpha_{i-1} x^i \end{aligned} \tag{29}$$

This is done in the inner loop by index j. Note that all of the operations are done modulo p. At the end, the algorithm returns the n-tuple $(\alpha_0, \ldots, \alpha_{n-1})$ as the final coefficients.

4 Security and Efficiency Analysis

4.1 Collusion Resistance

Our scheme is resistant to collusion attack. Suppose user U_1 and U_2 respectively have attribute a_j and $a_{j'}$ and they want to collude and create a user with both a_j and $a_{j'}$. The former has $g^{r/(v_j z)}$ as his own secret key of a_j while the latter has $g^{r'/(v'_j z')}$ as secret key of $a_{j'}$. Note that each user has its r and z and the value of r and v_j are unknown to the user. The successful collusion needs to generate $g^{r/(v_{j'} z)}$ from $g^{r/(v_j z)}$ and $g^{r'/(v'_j z')}$. But, this is impossible. Since U_1 does not know the value of r and v_j and the same thing is true about r' and $v_{j'}$ for U_2. Indeed, the secret key is

blinded with a distinct random value that is uniquely assigned to each user and is unknown to him/her.

4.2 Forward Secrecy

Forward secrecy in our scheme means that a user whose attributes are revoked is unable to decrypt the ciphertext that is encrypted by new attributes. The proposed scheme has a secure attribute revocation since the new private key of the attribute is embedded in a new access polynomial and further, all associated cipher keys are re-encrypted using the new private key.

Suppose an insider attacker has access to the secret key of all users, i.e., $g^{r/(zv_j)}$ for users who have attribute a_j. Actually, this is not a realistic assumption. But, we want to show that even this attacker with enough computational power and time is unable to recover a new attribute private key after revocation. Suppose that the attacker is able to factorize $f^j(x, r_j) - g^{r/(zv_j)}$ for each secret key and obtain the primitive factors $x - H(K_i, r_j)$ for each U_i. Due to the exclusion of its pre-shared key from the new access polynomial, the attacker's goal is to retrieve the new $g^{r/(zv_j)}$ by using any $H(K_i, r_j)$ of a non-revoked user still included in the new polynomial. But, when the attribute is revoked, r_j is replaced by r_j^*, and the new access polynomial is built by $H(K_i, r_j^*)$. Thus, the attacker faces the following problems:

- Computing K_i from $H(K_i, r_j)$ and r_j, and then computing $H(K_i, r_j^*)$
- Computing $H(K_i, r_j^*)$ directly from $H(K_i, r_j)$, r and r_j^* without revealing K_i

The first option yields hash reversing, which contradicts the one-wayness of cryptographic hash functions. For the second, we conjecture that it is as hard as the first one. Therefore, a successful attack on the access polynomial such as [20] is impossible. It is worth mentioning that the state-of-the-art factoring polynomial with rational coefficient is not so efficient. The first polynomial-time algorithm for factoring polynomials over finite fields was proposed in [22]. Its running time is $O(n^{12} + n^9(log|f|)^3)$ where n is the degree of polynomial and $|f|$ for $f(x) = a_0 + a_1x + \cdots + a_nx^n$ is defined as:

$$|f| = (a_0^2 + a_1^2 + \cdots + a_n^2)^{1/2} \tag{30}$$

However, even assuming the feasibility of accessing polynomial factoring, the revoked user is incapable of obtaining the private key of the new attribute. More precisely, the forward secrecy of our scheme is based on the one-wayness of the cryptographic hash function.

4.3 Statistical Analysis

Since we assume the cloud server is semi-honest-but-curious, there is a potential for the cloud server to investigate the query keywords that are requested by the data users over a long period of time. But, the proposed scheme is secure against statistical analysis. It basically originated from the fact that we encrypt the keywords of the file into query access polynomials. Because the data user sends its query in terms of $H(w, r_q)$ per each keyword w, the cloud server cannot guess or even obtain a verifiable text for w since r_q is only known to the users with corresponding attributes. Further, since r_q is unique per encrypted file, the query for a single word w in two different files leads to distinct terms $H(w, r_q)$ and $H(w, r'_q)$. Thus, the cloud server could not learn about the common query pattern for different files. Therefore, statistical analysis of the user queries is not feasible.

4.4 Complexity of Updating Access Polynomial

Updating the access polynomial is one of the most time-consuming operations for the trusted authority during secret key generation and attribute revocation. Suppose n denotes the number of users who incorporate in access polynomial for attribute a_j. For each user, a basis polynomial with degree $n-1$ is computed according to Algorithm 1. Since the number of multiplications/additions in iteration i of this algorithm is equal to $i-1$, the whole number of modular multiplications/additions becomes $1 + 2 + \cdots + (n-2) = \frac{1}{2}(n-1)(n-2)$. As the access polynomial consists of n basis polynomial, the process for Equation 7 consumes $n\{\frac{1}{2}(n-1)(n-2)+n\}$ operations. Moreover, the Lagrange interpolation according to Equation 8 takes n^2 multiplications/additions. Thus the whole process takes $n\{\frac{1}{2}(n-1)(n-2)+n\} + n^2 = \frac{1}{2}(n^3 - n^2 + 2n)$ operations. Further, the number of Hash operations is equal to n. In summary, the complexity of updating the access polynomial is $O(n^3)$.

5 Evaluations

This section presents an experimental analysis of the computational overhead of the proposed scheme. Then, we make a comparison of our scheme with some well-known architectures in terms of supported capabilities.

5.1 Evaluation of Access Polynomial Coefficients

The only bottleneck that can be raised in the proposed scheme is due to the updating access polynomial when the number of users grows up. Although this operation

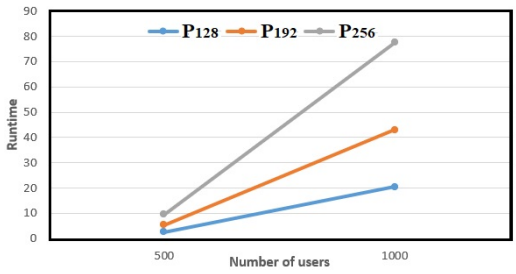

Figure 4. Runtime for computation of the access polynomial coefficients (On Intel core i7 with 4G bytes RAM)

is $O\left(n^2\right)$, it is possible to provide good scalability by selecting an appropriate prime for the operations in GF(p). The choice of p is dependent on the symmetric algorithm used for file encryption. Since AES which is used in our scheme supports 128/192/256-bit keys, we consider 128, 192 and 256-bit prime numbers which have a fast reduction algorithm. They are respectively selected as:

$$P_{128}=2^{128}-2^{97}-1 \tag{31}$$

$$P_{192}=2^{192}-2^{64}-1 \tag{32}$$

$$P_{256}=2^{256}-2^{224}+2^{192}+2^{96}-1 \tag{33}$$

Note that P192 and P256 are part of the proposed prime numbers for NIST ECC in GF(p). A prime with a fast reduction algorithm significantly speeds up the modular multiplications and thus efficiently accelerates the computations of access polynomials. We use SHA-256 for the hash function in access polynomials when P256 is used. For P_{128} and P_{192}, we use the least significant 128 and 192 bits of the hash output, respectively. We focus on the implementation of Algorithm 1 which is $O\left(n^2\right)$ and run by the trusted authority in both secret key generation and attribute revocation phases.

The result of the implementation is shown in Figure 4. The program written in C++ is compiled with Visual C++. It runs on a laptop PC with Intel Core i7 with 4GB of RAM. We see that a trusted authority with a regular system setting can run the algorithm for 1000 users on GF(P_{256}) in 77 seconds. Further, the running time for GF(P_{128}) and GF(P_{192}) are respectively equal to 20.5 and 43 seconds. It is a trade-off between security, the size of the prime and system scalability. We can select the optimum point based on the number of users that the system needs to support.

5.2 Comparison to Prior Work

This section provides an analysis and makes a comparison of the proposed scheme with prior work in terms of characteristics and efficiency. In Table 2, our scheme is briefly compared with the previous well-known work. The criteria for comparison are:

- Attribute revocation
- Decryption outsourcing
- Multi-keyword searching
- Search outsourcing
- Multiple attribute authorities
- Multi-owner setting
- Policy update
- Hidden Policy
- Verification of search result done by the cloud server
- Constant-size ciphertext
- Encryption policy

Attribute revocation is a major challenge in most of the ABE schemes. The schemes in [6–8, 13, 15, 17, 19, 23, 27] do not support attribute revocation. Further, in some schemes that support attribute revocation, it puts a great burden of computational and communication overhead on the data owner and the trusted authority. For example, in [9–12], the trusted authority is forced to communicate with all of the users to send the updated private key upon attribute revocation. We efficiently reduce the communication overhead by updating the access polynomial of each attribute on the cloud server. Each user updates his/her private keys on demand just whenever it is needed by downloading the polynomial coefficients from the cloud server.

The computational overhead on the user side can be mitigated by outsourcing the decryption process to either a cloud or a proxy server. This is supported in our scheme as well as in [10–12, 16, 23, 24, 26]. The next feature is necessary to enable data users to quickly find the required files from a vast amount of encrypted data. Further, it is amplified in terms of efficiency by outsourcing the multi-keyword search to the cloud/proxy server. It enables the data users to intelligently select the files for decryption and thereby reducing the overall computational and communication overhead of the scheme. We can see that the proposed scheme supports both multi-keyword searching and outsourcing of search operations, as well as the work in [6–9, 13, 15, 23, 27]. In all of them, the type of search is exactly match, which means the query items must be exactly included in the text without any prefix or suffix. We can see that the proposed scheme is the only scheme that simultaneously supports the first four features mentioned above, i.e., attribute revocation, decryption outsourcing, multi-keyword searching and outsourcing of search operation.

However, some features are not supported in our scheme. Among them, multi-authority refers to an ABE scheme that is appropriate for access control

Table 2. Comparison with other schemes (CP: Ciphertext-Policy KP: Key-Policy)

Feature	ours	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]	[13]	[14]	[15]	[16]	[17]	[18]	[23]	[24]	[25]	[26]	[27]	[28]
Attribute revocation	✓	✓	×	×	×	✓	✓	✓	✓	×	✓	×	✓	×	✓	×	✓	✓	✓	×	✓
Decryption outsourcing	✓	×	×	×	×	×	✓	✓	✓	×	×	×	✓	×	×	✓	✓	×	×	✓	×
Multi-keyword searching	✓	×	✓	✓	✓	✓	×	×	×	✓	×	✓	×	×	×	✓	×	×	×	✓	✓
Search outsourcing	✓	×	✓	✓	✓	✓	×	×	×	✓	×	✓	×	×	×	✓	×	×	×	✓	×
Multi-authority	×	×	×	×	×	×	×	×	✓	×	✓	×	×	×	×	×	×	×	×	×	✓
Multi-owner	×	×	×	✓	×	×	×	×	×	✓	×	×	×	×	×	×	×	×	×	×	×
Policy update	×	×	×	×	×	×	×	×	✓	×	×	×	×	✓	✓	×	×	×	×	×	✓
Hidden Policy	×	×	×	×	×	×	×	×	×	×	✓	✓	×	×	×	×	×	✓	×	×	×
Verification of Search	×	×	✓	✓	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	✓	×
Constant-size ciphertext	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	✓	✓	✓	✓	×
Encryption policy	CP	CP	CP	CP	CP	CP	CP	CP	CP	CP	CP	CP	CP	KP	CP	CP	CP	CP	CP	CP	CP

in the cloud storage system where the users hold attributes issued by different authorities. The schemes in [12, 14, 28] support multiple attribute authority as well as attribute revocation.

Further, multi-owner ABE is a scheme, in which multiple data owners are allowed to share their data with a flexible access policy and thereby, authorized data users can issue search queries according to their corresponding attributes. It is supported by the schemes proposed in [7, 13]. Policy updating is another feature that allows data owners to flexibly adjust the access control policy over their encrypted data. A trivial solution is to make the data owner respectively retrieve the ciphertext from the cloud, decrypt it, re-encrypt the plaintext under a new access policy, and finally send back the new ciphertext to the cloud server. This method is not efficient due to the high communication overhead imposed on network bandwidth as well as the high computation on data owners. Policy updating is supported by the schemes in [12, 17, 18, 28]. The main idea is that the data owner generates new keys and sends them to the cloud server. Then, the cloud server updates the ciphertext using new keys. Hidden policy refers to a CP-ABE scheme that preserve the privacy of the access structure in the ciphertext. The results show that the schemes [14, 15, 25] support hidden policy.

The schemes in [6, 7, 27], in addition to outsourcing the search operations, allow the data user to verify whether the cloud server has faithfully executed the search operations. In [7], it is achieved by generating an authentication tag by the data owner for each encrypted file and a challenge-response protocol between the data user and the cloud server. The schemes in [24–27] have an odd feature named constant-size ciphertext. It means that against the other schemes in which the ciphertext length is growing up by the number of attributes involved in the encryption, these schemes have a fixed-length ciphertext and thereby reducing the storage cost.

Finally, the encryption policy for the ABE scheme is either ciphertext-policy or key-policy. We can see that [17] is the only work that uses key-policy, KP-ABE, which puts the access structure in the key instead of ciphertext (CP-ABE). Table 3 and Table 4 respectively compare the communication overhead and computational complexity of our scheme with the schemes in [9, 10, 12, 23, 26–28]. The complexity of each algorithm is measured based on the required primitive operations. We mainly focus on four kinds of primitive operations: exponential/multiplication on G_0, exponential/multiplication on G_t, bilinear pairing and hash operation.

We can see that the encryption which is done by the data owner has about the same cost for all the schemes. Precisely, regarding the group operations in G_0, our scheme has the same complexity as the schemes in [10, 12, 26–28] and lower complexity than the schemes in [9, 23].

For keyword encryption, existing only in the schemes supporting multi-keyword search, our scheme has more complexity than [9, 23] in terms of group operations in G_0 since we embed the keywords in a query access polynomial. In contrast, our scheme acts more efficient than [9, 23, 27, 28] in searching phase due to the following reasons:

- In our scheme, search is accomplished only by multiplications in group G_0 while in [9], it requires bilinear pairing operations in addition to group operation.
- In our scheme, the number of group operations is equal to while it is equal to in [9] since the cloud server is to arbitrarily select N_Q compo-

nents from N_W words. Note that $\binom{n}{m}$ denotes the combination, the number of choices m elements from a set of n elements.

Moreover, on the user side, our scheme is simpler than [9, 23, 27, 28]. Because the data user in [9] must perform $3N_Q$ group operations to generate search token in addition to N_Q hash operations necessary in our scheme. Further, the schemes in [23, 27, 28] need to perform at least a single bilinear pairing operation per each query keyword at the user side. The decryption operation on the user side, for the schemes which do not support outsourcing, is dependent on the number of attributes that appeared in ciphertext [9, 28]. For example, in [9], the decryption is accomplished by $2k + 1$ group operations which k denotes the number of attributes owned by the data user. In contrast, for our scheme along with [10, 12, 23, 26] which supports outsourcing, decryption at the user side is carried out by a constant number of group operations. The main part of decryption is done by the cloud server in the transform/pre-computation phase. It includes both group and bilinear mapping operations. The number of operations is dependent on the number of attributes either appearing in ciphertext or owned by the data user. The last two operations, update key and ciphertext update are required for attribute revocations. The latter which is done by the cloud server is the same for all the schemes since it requires exponentiation for each ciphertext consisting of the revoked attribute. While the former, which is done by the attribute authority, is simply accomplished by only one exponentiation in literature [9, 10, 12]. But, in our scheme, it triggers an update event for access polynomial of the revoked attribute which imposes $\frac{1}{2}N_j^3 \cdot G_0 + N_j C_H$ on the attribute authority.

However, our scheme outperforms other schemes in terms of communication overhead triggered by attribute revocation. Table 3 shows the number of required messages during the update key phase of attribute revocation. The attribute authority in [9, 10] must send new keys to all non-revoked users, i.e., N_j. While in our scheme, it is sufficient for the attribute authority to send only a single message containing the attribute access polynomial to the cloud server. The data users are gradually updating their keys on demands when they are going to run the decryption or searching phase, i.e., getting the latest version of attribute access polynomials.

It is worth noting that users are not always online in practical applications. In previous revocation schemes such as [3–5, 9–12], the attribute authority is responsible for updating and sending new attribute keys to non-revoked data users. It leads to a more complex attribute authority that needs continuous checking of whether a data user is online to establish a secure channel via session key agreement protocol and thereby transmit the new keys to him/her.

Table 3. Communication Overhead

Operation	[9]	[10]	Our Scheme
Key-Update	N_j	N_j	1

6 Conclusion

This paper presented a CP-ABE scheme that supports attribute revocation and decryption outsourcing as well as the ability to multi-keyword searching and outsourcing of search operations. The idea behind both attribute revocation and multi-keyword searching is the use of access polynomials. The results showed that the scheme outperforms previous schemes in terms of communication overheads imposed on the attribute authority and field computations imposed on data users.

References

[1] Q. Chai and G. Guang. Verifiable symmetric searchable encryption for semi-honest-but-curious cloud servers. In *IEEE International Conference on Communications (ICC)*, 2012.

[2] J. Bethencourt, S. Amit, and B. Waters. Ciphertext-policy attribute-based encryption. In *IEEE symposium on security and privacy (SP'07)*, 2007.

[3] G. Wang, Q. Liu, J. Wu, and M. Guo. Hierarchical attribute-based encryption and scalable user revocation for sharing data in cloud servers. *Computers & Security*, 30:320–331, 2011.

[4] J. Hur and D. K. Noh. Attribute-based access control with efficient revocation in data outsourcing systems. *IEEE Transactions on Parallel and Distributed Systems*, 22:1214–1221, 2010.

[5] L. Zu, Z. Liu, and J. Li. New ciphertext-policy attribute-based encryption with efficient revocation. In *IEEE International Conference on Computer and Information Technology*, 2014.

[6] Q. Zheng, S. Xu, and G. Ateniese. Vabks: verifiable attribute-based keyword search over outsourced encrypted data. In *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*, 2014.

[7] Y. Fan and Z. Liu. Verifiable attribute-based multi-keyword search over encrypted cloud data in multi-owner setting. In *IEEE Second International Conference on Data Science in Cyberspace (DSC)*, 2017.

[8] R. Li, D. Zheng, Y. Zhang, H. Su, M. Yang, and P. Lang. Attribute-based encryption with multi-keyword search. In *IEEE Second Interna-*

Table 4. Comparison of computational complexity

Operation	[9]	[10]	[12]	[23]	[26]	[27]	[28]	Our Scheme
Encryption	$(3t+2k+4)G_0+G_t$	$(t+2)G_0+2G_t$	tG_0+4G_t	$(2t+1)G_0+G_t$	$(t+2)G_0+C_e$	$(t+6+N_W)G_0$	$(t+N_W)G_0+5G_t$	$(t+3)G_0+2G_t$
Keyword encryption	$2N_WG_0+N_WC_H$	-	-	N_WG_0	-	-	-	$\frac{1}{2}N_W^2G_0+N_WC_H$
Decryption (cloud)	-	$(2\|S\|+1)G_t+(k+1)C_e$	$O(\|S\|)G_0+O(k)C_e$	$(2\|S\|+2)G_t+(k+1)C_e$	$(t+2)G_0+2C_e$	-	-	$(2\|S\|+2)G_t+(k+1)C_e$
Decryption (User)	$(2k+1)G_0+(2k+1)G_t$	$2G_t$	$O(1)G_t$	$2G_t$	$2G_0+C_e$	$2G_0+C_e$	$(t+5)G_0+(t+2)G_t$	$2G_t$
Searching (User)	$3N_QG_t+N_QC_H$	-	-	$(2k+1)C_e+N_WG_t$	-	$(N_Q+3)C_e$	$(N_Q+3)C_e+2G_0$	N_QC_H
Searching (Cloud)	$\binom{N_W}{N_W}(N_Q+3)(C_e+G_t)$	-	-	$(2k+1)G_0+N_WG_0$	-	-	-	$N_QN_WG_0$
Key-Update	G_0	G_0	G_0	-	-	-	$(k-1)(G_0+C_e)$	$G_0+1/2N_j^3.G_0+N_jC_H$
CT-Update	$N_{C_j}G_0$	$N_{C_j}G_0$	$N_{C_j}G_0$	-	-	-	$N_{C_j}G_0$	$N_{C_j}G_0$

G_0	Multiplication/Exponentiation in G_0	C_e	Bilinear pairing operation
G_t	Multiplication/Exponentiation in G_t	C_H	Hash operation
S	Least internal nodes satisfying T	N_Q	Number of keywords in user query
t	The number of attributes in ciphertext	N_W	The number of keywords in file
k	The number of attributes owned by the user	N_{C_j}	The number ciphertexts encrypted by attribute j

tional Conference on Data Science in Cyberspace (DSC), 2017.

[9] S. Wang, L. Yao, and Y. Zhang. Attribute-based encryption scheme with multi-keyword search and supporting attribute revocation in cloud storage. *PloS one*, 19, 2010.

[10] H. Liu, P. Zhu, Z. Chen, P. Zhang, and Z. L. Jiang. Attribute-based encryption scheme supporting decryption outsourcing and attribute revocation in cloud storage. In *IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC)*, 2017.

[11] Z. Xia, L. Zhang, and D. Liu. Attribute-based access control scheme with efficient revocation in cloud computing. *China Communications*, 13:92–99, 2018.

[12] Z. Liu, Z. L. Jiang, X. Wang, and S. M. Yiu. Practical attribute-based encryption: Outsourcing decryption, attribute revocation and policy updating. *Journal of Network and Computer Applications*, 108:112–123, 2018.

[13] Y. Miao, J. Ma, X. Liu, F. Wei, Z. Liu, and X. A. Wang. m 2-abks: Attribute-based multi-keyword search over encrypted personal health records in multi-owner setting. *Journal of medical systems*, 40:246–253, 2016.

[14] H. Zhong, W. Zhu, Y. Xu, and J. Cui. Multi-authority attribute-based encryption access control scheme with policy hidden for cloud storage. *Soft Computing*, 22:243–251, 2018.

[15] H. Wang, X. Dong, and Z. Cao. Multi-value-independent ciphertext-policy attribute based encryption with fast keyword search. *IEEE Transactions on Services Computing*, 13(6), 2017.

[16] J. Li, W. Yao, J. Han, Y. Zhang, and J. Shen. User collusion avoidance cp-abe with efficient attribute revocation for cloud storage. *IEEE Systems Journal*, 12:1767–1777, 2017.

[17] S. Belguith, N. Kaaniche, and G. Russello. Lightweight attribute-based encryption supporting access policy update for cloud assisted iot. In *Proceedings of the 15th International Joint Conference on e-Business and Telecommunications-Volume 1: SECRYPT*, 2018.

[18] W. Yuan. Dynamic policy update for ciphertext-policy attribute-based encryption. In *IACR Cryptology ePrint Archive*, page 457, 2018.

[19] K. Xue, W. Chen, W. Li, J. Hong, and P. Hong. Combining data owner-side and cloud-side access control for encrypted cloud storage. *IEEE Transactions on Information Forensics and Security*, 13:2062–2074, 2018.

[20] A. Kamal. Cryptanalysis of a polynomial-based key management scheme for secure group communication. *Int. J. Netw. Secur.*, 15(1):68–70, 2013.

[21] X. Sun, X. Wu, C. Hoang, Z. Xu, and J. Zhong. Modified access polynomial based self-healing key management schemes with broadcast authentication and enhanced collusion resistance in wireless sensor networks. *Ad Hoc Networks*, 37:324–336, 2016.

[22] Arjen K Lenstra, Hendrik Willem Lenstra, and László Lovász. Factoring polynomials with rational coefficients. *Mathematische annalen*, 261(ARTICLE):515–534, 1982.

[23] Y. Cui, X. Gao, Y. Shi, W. Yin, E. Panaousis, and K. Liang. An efficient attribute-based multi-keyword search scheme in encrypted keyword generation. *IEEE Access*, 8:99024–99036, 2020.

[24] Y. Zhao, X. Xie, X. Zhang, and Y. Ding. A revocable storage cp-abe scheme with constant ciphertext length in cloud storage. *Mathematical biosciences and engineering: MBE*, 16(5):4229–4249, 2019.

[25] A. Wu, D. Zehng, Y. Zhang, and M. Yang. Hidden policy attribute-based data sharing with direct revocation and keyword search in cloud computing. *Sensors*, 17(7):2158, 2018.

[26] Y. W. Hwang and I. Y. Lee. Cp-abe access control that block access of withdrawn users in dynamic cloud. *KSII Transactions on Internet*

and Information Systems (TIIS), 14(18):4136–4156, 2020.

[27] S. Wang, S. Jia, and Y. Zhang. Verifiable and multi-keyword searchable attribute-based encryption scheme for cloud storage. *IEEE Access*, 7:50136–50147, 2019.

[28] M. Ali, C. Xu, and A. Hussain. Authorized attribute-based encryption multi-keywords search with policy updating. *Journal of New Media*, 2(1):31–48, 2020.

Sajjad Palanki was born in Chaypara, Iran in 1995. He received a B.Sc. degree in computer engineering from Payame Noor University, in 2017. He also received an M.Sc. degree in computer engineering from Tarbiat Modares University, Tehran, Iran, in 2020. Since 2019, he is a researcher in ABE (Attribute Based Encryption) at Tarbiat Modares University. His research area includes attribute-based encryption and blockchain. His current interest is privacy protocols for IoT and blockchain. He also works as a front-end developer.

Alireza Shafieinejad is an Assistant Professor in the Department of Electrical and Computer Engineering, at Tarbiat Modares University, Tehran, Iran. He received a B.Sc. degree in computer engineering from the Sharif University of Technology, Tehran, in 1999. He also received M.Sc. and Ph.D. degrees in electrical engineering from Isfahan University of Technology, Isfahan, Iran, respectively in 2002 and 2013. Since 2015, he has been an Assistant Professor at Tarbiat Modares University, Tehran, Iran. His research interests are in the area of Wireless Network Coding, Network Security, and the Design and Analysis of Cryptography Algorithms. At Tarbiat Modares University, he leads research in network security and penetration testing in SE-Lab (Security Evaluation Laboratory).