IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 5 December 2023, accepted 6 December 2023, date of publication 8 December 2023,
date of current version 29 December 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3341095

RESEARCH ARTICLE

A Fully Outsourced Attribute-Based Signcryption Scheme Supporting Privacy-Preserving Policy Update in Mobile Cloud Computing

PATTAVEE SANCHOL AND SOMCHART FUGKEAW, (Member, IEEE)
School of Information, Computer, and Communication Technology (ICT), Sirindhorn International Institute of Technology, Thammasat University, Pathum Thani 12121, Thailand

Corresponding author: Somchart Fugkeaw (somchart@siit.tu.ac.th)

ABSTRACT Existing fine-grained and secure access control systems deployed in mobile cloud computing (MCC) typically focus on offloading encryption and/or decryption workloads to the delegated proxy in the cloud server. However, the privacy and authenticity management of outsourced data with flexible and efficient access policy update in MCC is generally overlooked by existing works. In fact, the signcryption feature is generally required in data access control system where the trust between data owner and multiple mobile users is crucial while the policy update management deals with the dynamic handling of user privilege control lifecycle. In this paper, we propose a privacy-preserving access control scheme supporting signcryption and efficient policy update with policy hiding in MCC setting. Essentially, a fully outsourced attribute-based signcryption (ABSC) and policy update method are devised to allows the data owner to offload ABS operation and policy update cost to be executed by the delegated proxy. Finally, we present our experiments to demonstrate that our proposed scheme is computationally more efficient compared to related works.

INDEX TERMS Access control, CP-ABE, ABSC, mobile cloud, key transformation, policy update.

I. INTRODUCTION

Mobile cloud computing (MCC) offers high flexibility and accessibility to the mobile users for consuming resources such as mobile applications and data outsourced in the cloud. Cloud services and data are fully managed and operated in the cloud platforms where plentiful system resources and storages are available to serve a high volume of users. Here, the computational processing of processes or tasks in using the services is done at the cloud server while mobile users only get the service or do partial processing at their mobile devices.

In addition to a ton of services offered by cloud service providers (CSPs), most organizations or individuals tend to outsource their applications and data hosted on cloud and allow their users to access them via mobile devices.

The associate editor coordinating the review of this manuscript and approving it for publication was Mehdi Sookhak.

Even though MCC delivers more flexibility through the mobility for accessing outsourced services, the access control together the data privacy of the shared resources including data is highly important. Specifically, the content of data that are transferred between the cloud and mobile devices should be protected both during the communication and at the user's device. To this end, a cryptographic-based access control featured with data encryption and access control enforcement is required to assure that the data located in the cloud is accessed by authorized mobile data users. Specifically, there are three major requirements for the rigorous access control in MCC. First, the cryptographic-related operations performed by both data owner and the data user should be lightweight. This is because the access control requirement can evolve at any time due to the change of user status and change of data sharing requirement. To this end, handling the secure policy update with optimized cost is crucial for the data owners in MCC environment. In fact, the data owners may need to share the data using the lightweight encryption

via their mobile devices and the authorized users can access the shared data with lightweight cost of decryption. Second, the anonymous authenticity of data outsourced is important for rigorous access control in MCC where a large volume of mobile users access the data. The users should assure that the data they used is shared by the legitimate owners and the integrity of the data is preserved. Finally, the flexible and secure enforcement of the access control in MCC to their users is non-trivial. This is because the access control requirement can be evolved any time as of the change of user status and change of data sharing requirement. To this end, handling the secure policy update with optimized cost is a crucial for the data owners in MCC environment.

In the context of fine-grained and secure access control, ciphertext policy attribute-based encryption (CP-ABE) has gained popularity for enabling one-to-many data sharing. In CP-ABE, data owners can define access policies and use them to encrypt data, while authorized users can employ their secret keys with matching attributes to decrypt the ciphertext. However, using CP-ABE directly in mobile cloud computing (MCC) is impractical due to its resource-intensive cryptographic operations, particularly in encryption and decryption. The primary cryptographic costs of CP-ABE involve pairing operations and exponentiation, which are unsuitable for resource-constrained devices. Additionally, the encryption and decryption overheads of CP-ABE increase linearly with the number of attributes in the access policy and the size of the message or ciphertext. Even though some approaches [36], [37] proposed the CP-ABE with constant-size decryption key and/or ciphertext, the cost of several decryption requests is non-trivial for mobile devices. In addition, they generally focused on the reduced cost for user-end.

For years, many research works [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11] have proposed the approaches that either partially or fully outsource expensive operations of CP-ABE to be done by the delegated proxy located on the cloud environment. This technique is recognized as the concept of proxy re-encryption (PRE), originally introduced by Kawai [12]. The contribution of the computation outsourcing model is to offload costly operations to be run in the cloud while the mobile clients only run smaller cost of decryption.

To meet the demand for cryptographic-based access control with signcryption capabilities, attribute-based signcryption (ABSC) stands out as a suitable method for ensuring both data confidentiality and authenticity simultaneously. ABSC has found application in various access control solutions based on attribute-based encryption (ABE). Nevertheless, these solutions were not originally designed to provide lightweight signcryption and designcryption functionalities in MCC.

Regarding the policy updates issue, data owners encounter the challenge of downloading, re-encrypting, or updating all relevant ciphertexts and then uploading them to store in the cloud. This process hinders the feasibility of deploying CP-ABE, particularly when policy changes are frequent, and there is a substantial user base and a large volume of ciphertexts stored in the cloud in data-sharing environments. Moreover, ciphertext generated through CP-ABE typically includes attribute information, which becomes a critical concern when the access policy contains sensitive attributes.

To the best of our knowledge, no prior works have comprehensively addressed the various research gaps mentioned, which include the development of a unified solution for lightweight, verifiable, and finely-grained access control with efficient policy updates in MCC. Essentially, satisfying all these requirements simultaneously is a real challenge since implementing secure with lightweight access control protocols for supporting both data owners and data users in mobile cloud is significant. Fulfilling these requirements is instrumental for enhancing the practicality of implementing the evolvable access control where there are substantial number of data owners and users in mobile cloud environment. In this paper, we are thus driven to introduce a Full Ciphertext Policy Attribute-based Signcryption (FCP-ABSC) outsourcing scheme that encompasses all of the aforementioned requirements. The contributions of our paper are summarized as follows.

1. We proposed a new lightweight signcryption and designcryption scheme that can be efficiently implemented in the MCC setting. This enables data owners to share data with privacy-preserving properties and anonymous authenticity. It also allows data users to verify that the accessed data is shared by the legitimate data owner.
2. Our proposed scheme supports lightweight decryption and policy update based on our outsourced decryption and outsourced policy update model. We also introduced ciphertext indexing method to query the affected ciphertexts when the policy is updated. This enables fast re-encryption process.
3. Our proposed scheme entails the privacy-preserving policy enforcement with the support of policy hiding where the plain attributes constituting in the policy are not revealed.
4. We conducted the comparative analysis and experiments to show the efficiency of our proposed scheme.

The remaining sections of this paper are organized as follows: Section II discusses works related to the scheme proposed in this paper. Section III provides the theoretical foundations of CP-ABE which are essential for understanding the proposed scheme. Section IV details the proposed lightweight data access control scheme for mobile cloud computing. Section V presents security analysis. Section V-D provides the results of evaluations and experiments. Section VI presents the concluding remarks of the paper.

II. RELATED WORK

In this section, we provide a review of related literatures entailing the outsourcing decryption, attribute-based signcryption, and access policy update.

A. OUTSOURCING DECRYPTION

In [1], Green et al. introduced a concept called a transformation key that allows the cloud to translate any ABE ciphertext satisfied by that user's attributes into a (constant-size) El Gamal-style ciphertext [2]. By avoiding the computationally intensive ABE bilinear pairing and distributing some of the decryption tasks to the users, the scheme achieves a more lightweight and efficient data access control. This optimization leads to reduced processing overhead on the cloud service provider's side and improves the overall performance for mobile users.

In [3], Asim et al. introduced an outsourced CP-ABE encryption and decryption approach using dual proxy model to support outsource encryption and decryption process. Specifically, the decryption is based on the concept of transformation key computation [1].

In [4], Fugkeaw proposed an efficient access control mechanism aimed at providing robust and detailed data access control in MCC while minimizing the computational burden of ciphertext decryption. This approach involves employing a two-tier encryption scheme. Initially, the data is encrypted using random strings, generating an intermediate ciphertext. Subsequently, the intermediate ciphertext undergoes an additional layer of encryption using the CP-ABE method, resulting in the production of the final ciphertext. To secure the random string and user secret key, they are both encrypted using the user's public key. For the decryption process, the proxy has the secret key to decrypt the outer layer of the ciphertext utilizing the CP-ABE decryption algorithm. Hereafter, the intermediate ciphertext is sent back to the user for final decryption.

In [10], Li et al. proposed a privacy-preserving access control scheme for mobile multimedia in the cloud. In this scheme, online encryption with policy hiding and user decryption are proposed. For the encryption, the data owner specifies the policy for online encryption using CP-ABE method. The decryption process involves partial outsourcing of CP-ABE decryption to the cloud service provider (CSP). The provider takes on the responsibility of performing the matching test between attribute names and access policies. As of the matching test, the user obtains the intermediate ciphertext. The final encrypted element is then returned to the user, requiring only one modular exponential operation to obtain the original message.

In [5], Li et al. presented a lightweight data access control scheme for mobile cloud computing. This scheme combines both symmetric encryption and the CP-ABE algorithm. The data is encrypted using the symmetric encryption algorithm, while the symmetric key itself is encrypted using the CP-ABE algorithm. A significant aspect of this scheme is the offloading of the access control tree transformation, which is a computationally intensive process in CP-ABE, to be executed by proxy servers. Furthermore, the authors introduced a lazy revocation method to support user revocation functionality.

In [6], Wang et al. proposed a fast CP-ABE system to support access control in mobile healthcare network. In this scheme, three semi-trusted third parties were employed to run key generation, encryption, and decryption separately. The scheme guarantees the accuracy of the decryption result at the mobile user by using a Boneh–Lynn–Shacham short signature scheme. However, this scheme heavily relied on the service providers to serve the core cryptographic operations.

In [14], Yu et al. proposed an attribute-based signature (ABS) scheme providing expressive access policy and efficient the signature. The verification method is based on the signature transformation algorithm to encrypt a signature and sends to the server in the cloud compute an intermediate signature and returns it to the verifier.

In [32], we introduced the outsourced CP-ABE decryption method to support lightweight decryption in MCC. Initially, the data is encrypted using symmetric encryption, and then the symmetric key is further encrypted using the CP-ABE method. The outsourced proxy is responsible for running the CP-ABE decryption while the data user can compute the symmetric key through the symmetric key derivation function. Hence, the mobile user only deals with AES decryption in their device.

In [38], Li et al. proposed an ABE scheme with verifiable outsourced decryption scheme that is able to simultaneously check the correctness for transformed ciphertext for the authorized users and unauthorized users based on the different access policies. In addition, each ciphertext is added a MAC to enable the user to verify the correctness of the transformed ciphertext.

In [39], the authors proposed a decentralized attribute-based server-aid signature (DABSAS) scheme. In the DABSAS scheme, a server is delegated to perform costly computation in the signature and verification algorithms. Therefore, the users only perform light computation in the signing algorithm.

B. ATTRIBUTE-BASED SIGNCRYPTION

In 2010, Gagné et al. [24] firstly proposed an ABSC to achieve higher security of data. They formally defined the concept of *message confidentiality* and *ciphertext unforgeability* satisfying signcryption property in ABE model. Later, many works adopted this model for supporting ABSC in cryptographic-based data access control. For example, S. Belguith et al. [25] proposed the Threshold Attribute Based Signcryption for Cloud Application. Wang and Huang [26] applied an ABSC scheme in the monotone tree access structure. Then, Han et al. [27] proposed an ABSC scheme to be used in the nonmonotonic access structure supporting a constant length of signature.

In [28], Liu et al. proposed an ABSC scheme to support secure personal health record (PHR) system in cloud. In this scheme, the users are divided into professional domain and social domain which are specific to healthcare setting. However, this scheme does not support batch verification.

In [30], the authors proposed the ABSC scheme supporting verifiable outsourcing of designcryption. In this scheme, the designcryption process is outsourced to be executed in the cloud server while the constant computation is run on the user side.

In [31], Yu et al. proposed the ABSC by integrating the ciphertext-policy ABE and the key-policy ABS as a hybrid-policy ABSE scheme. In this scheme, the length of signature is designed to be constant. Also, the operations including signing, verification, and decryption are outsourced to be done in the fog nodes.

In [33], Ahene et al. proposed a heterogenous signcryption with proxy re-encryption (PRE) to provide secure access control to PHR system with anonymous authentication. In addition, the authors employed blockchain to support auditability and authentication.

However, all the above approaches do not achieve full outsourcing both signcryption and designcryption. In addition, they did not support policy update.

C. ACCESS POLICY UPDATE

Crucially, current approaches that tackle the issue of updating access policies within the CP-ABE model can be classified into two primary techniques: proxy re-encryption [15], [19], [20], [21], [22] and the ciphertext update [16], [17], [18].

In the PRE method, a semi-trusted proxy is introduced to serve the computationally intensive overheads of ciphertext re-encryption and key update [16] caused by the policy update or the revocation. For instance, Fugkeaw [17] proposed a lightweight policy update model to support data outsourcing in multi-authority environment. In this scheme, the ciphertext re-encryption caused by the policy update is fully offloaded to the proxy.

For ciphertext update, there is a process of the computation of ciphertext key update to enable the ciphertext elements containing the updated attribute in the policy to be updated without the re-encryption. For example, Li et al. [20] proposed a policy update outsourcing model based on the assistance of proxy server located in the cloud. When the policy updated initiated by the data owner, the data owner exploits the ciphertext key update parameters and sends to the proxy to perform the major ciphertext update operation. In [21], Guan et al. proposed a policy update mechanism in edge computing. This scheme supports outsourced decryption based on the proxy on cloud and policy update run by the cloud. If the policy is updated, the policy update key operation is required and it is used to update the ciphertext.

In [22], Belguith et al. proposed the PROUD model that supports verifiable policy update in IoT cloud. In this scheme, the designcryption overhead is outsourced to be executed by an edge server. Only the designcrypted result sent from the edge server is required to be done by the users. Nevertheless, this approach deals with the complexity of the cost of deciphering aggregated transformed key to compute the secret key. Furthermore, this approach does not support policy hiding.

However, all the above approaches did not provide mechanism to support ciphertext retrieval when there is a policy update case. This problem even becomes crucial when there are a large number of ciphertexts stored in the cloud because the naïve search to invoke the affected ciphertexts would significantly degrade the ciphertext re-encryption or ciphertext update process.

III. PRELIMINARIES

Bethencourt et al. [23] proposed the original ciphertext policy CP-ABE scheme in 2007. Its foundational construct is based on the bilinear maps. In this section, we define our access policy based on the access tree [23] where the attribute name and value are anonymized using hash-based method.

Definition 1: Access Tree T[24, 18]. Let T be a tree representing an access structure. Each non-leaf node of the tree represents a threshold gate, described by its children and a threshold value. If num_x is the number of children of a node x and k_x is its threshold value, then $0 < k_x \leq num_x$. When $k_x = 1$, the threshold gate is an OR gate and when $k_x = num_x$, it is an AND gate. Each leaf node x of the tree is described by an attribute and a threshold value $k_x = 1$. The *kofn* threshold gate is also allowed in T, in this case $k_x = k$ where k is the threshold value. determined in the *kofn* gate.

In our system, all attribute values are hashed and they are modeled in the leaf nodes with its corresponding attribute names. Accordingly, the values of all attributes are hidden while the policy is located in the cloud. Fig. 1 illustrates an example of the mapping between a normal policy and its corresponding hiding model.

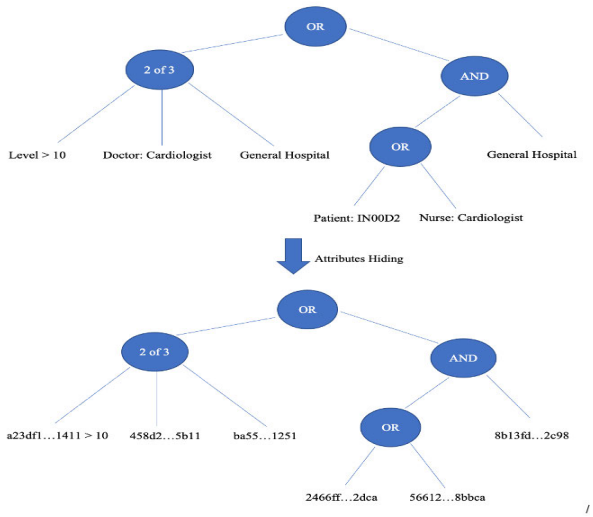

FIGURE 1. Access tree with attributes hiding.

IV. OUR PROPOSED SCHEME

In this section, we introduce the system model and the collection of cryptographic algorithms that constitute their

proposed scheme. The primary goal of our proposed scheme is to enable fine-grained, lightweight, and secure access control with policy update in the MCC environment.

A. SYSTEM OVERVIEW

We proposed a scalable and efficient Attribute Based SignCryption (ABSC) outsourcing scheme with the support of the efficient access policy update. Figure 2 illustrates the system overview of our proposed system model.

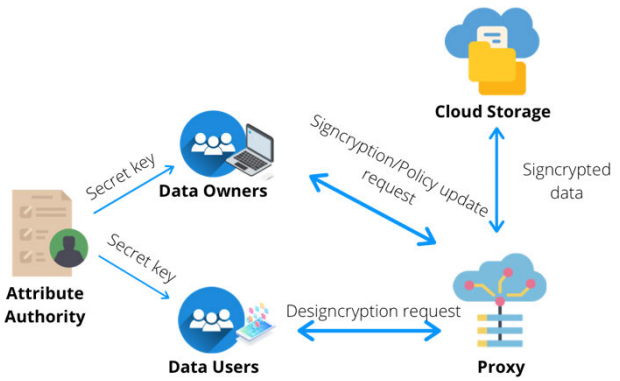

FIGURE 2. System model.

The system model consists of the following entities.

1) Attribute Authority (AA) is the trusted entity who issues, revokes, and updates users' attributes. AA is responsible for issuing the secret keys to data users.
2) Data Owners (DOs) outsource their data to cloud storage. DOs are responsible to update the access policies upon the access control requirement. In our system, DOs computes the hash value of message M. Then, M is encrypted based on the AES algorithm. Then, the symmetric key is signed and encrypted with the hash value of message M, $H(M)$ through our proposed ABSC-signcryption method.
3) Mobile users or data users (DU) are authorized to access the outsourced data using their mobile devices. Each user has a secret key issued by the AA. When DUs need to access the data, they send the designcryption request to the proxy.
4) Cloud Storage is the storage service provided by the cloud service provider (CSP). It is a centralized storage system where data can be stored and accessed by users over the internet.
5) Proxy server is a semi-trusted entity situated in the cloud environment, responsible for performing various cryptographic operations, such as data signcryption, designcryption, and re-signcryption. In this model, the proxy server is delegated the task of holding the user secret key (SK_{DU}), which suggests that the proxy acts on behalf of the user for certain cryptographic tasks, like decryption or re-signing. This delegation of authority enables the proxy to perform outsourced designcryption, reducing the computational burden on the user's device and enhancing the overall efficiency of the system.

B. SECURITY MODEL

In this section, we introduce four security models of our scheme.

1) MESSAGE CONFIDENTIALITY

This security model is defined on the indistinguishability of ciphertext against adaptive chosen ciphertext attack (IND-CCA2) in the selective access structure model. Our scheme is said to be indistinguishable for such attack if there is no probabilistic polynomial time (PPT) adversary that can win the Exp^{conf} security game with non-negligible advantage. The Exp^{conf} security game between adversary A and a challenger C is defined as follows.

Setup. The challenger C runs CreateAttributeAuthority and KeyGetFuncGenerator algorithms to generate master key MK, $KeyG_{FN}$., then gives the PK and the $KeyG_{FN}$. to the adversary A.

Phase1: The adversary A makes repeated secret key SK queries corresponding to sets of attributes $S_1, S_2, \ldots, Sq$. The challenger responds the adversary by executing the User.Key Gen algorithm.

Challenge. The adversary A submits two equal lengths of message M_0 and M_1 such that $|M_0| = |M_1|$ (both the messages are of equal length). In addition, A also provides access structure T^* such that none of the sets $S_1, S_2, \ldots, Sq$ from phase 1 satisfies the access structure T^*. The challenger flips a coin b: $b \in \{0; 1\}$ and generate random value RV_b and signcrypt RV_b under the access structure T^*. while M_b is encrypted by AES_Encryption(M_b, $KeyG_{FN}(RV_b)$) The ciphertext $CT_{RV}{}^*$ corresponding to an access structure T^* and $CT_M{}^*$ is given to the adversary A

Phase 2. Phase 1 is repeated with the restriction that none of the sets $S_{q+1}, S_{q+2}, \ldots, S_{q+r}$ satisfies the access structure T^* corresponding to the challenge ciphertext $CT_{RV}{}^*$ and $CT_M{}^*$.

Guess The adversary outputs a guess b_0 of b and wins a game if $b' = b$. Let $\text{Adv} = \Pr\left[b' = b\right] - \frac{1}{2}$ be the advantage of the adversary A in this game is defined.

Definition 2: Our scheme satisfies the INDCCA2 confidentiality property if the probability $\text{Adv}\left[Exp^{conf}\right]$ is negligible for adversaries.

2) CIPHERTEXT UNFORGEABILITY

This security model is defined on existential unforgeability against adaptive chosen message attack in the selective attribute set model through the following game between adversary A and a challenger C.

Setup. The challenger C runs CreateAttributeAuthority and KeyGetFuncGenerator algorithms to generate master key MK, $KeyG_{FN}$., then gives the PK and the $KeyG_{FN}$ to the adversary A.

Query Phase: The adversary A makes signcryption queries for message M and random value RV corresponding to sets

of attributes $S_1, S_2, \ldots, Sq.$ and Access Policy T. The challenger responds the adversary by executing the User.KeyGen algorithm. Then the challenger C performs signcryption RV_b under the access structure T, and encrypts M_b using AES_Encryption(M_b, $KeyG_{FN}(RV_b)$). Then, C outputs the ciphertext $CT_{RV}{}^*$ corresponding to an access structure T and $CT_M{}^*$.

Forgery Phase: In this phase, the adversary A computes and outputs the ciphertext CT_{RV^*} and CT_{M^*}, that did not obtain from the Query Phase. Then the challenger C performs designcryption of the CT_{RV^*} and CT_{M^*} by executing the User.Decrpt and Proxy.Decrypt algorithm. In this case, A wins a game if the RV and M are correctly designcrypted.

Definition 3: Our scheme is unforgeable against chosen-message attack (EUF-CMA), if the probability Adv $\left[Exp^{unf}\right]$ is negligible for all adversaries.

3) VERIFIABILITY

This security model is defined on the following game between adversary A and a challenger C.

Setup. The challenger C runs CreateAttributeAuthority and KeyGetFuncGenerator algorithms to generate master key MK, $KeyG_{FN}$. Then C forwards the PK and the $KeyG_{FN}$ to the adversary A.

Challenge: The adversary A makes secret key SK queries corresponding to sets of attributes $S_1, S_2, \ldots, Sq.$ The challenger responds the adversary by executing the User.KeyGen algorithm and sends SK to A.

Output: The adversary A provides access structure T^* such that the sets $S_1, S_2, \ldots, Sq$ satisfies the access structure T^* to the challenger C for signcrypting the message M and random value RV and runs the designcryption phase to get RV and M. In this model, A win this game if $RV \notin \{RV^*, \perp\}$, where M^* is the associated plaintext of CT^*.

Definition 4: Our scheme is verifiable if any data user who possesses a set of attributes S_{DU} satisfying the access control policy T can successfully verify and decrypt the signcrypted message associated with that policy.

4) PRIVACY

Signer privacy means that the signature's distribution is not influenced by the secret key employed in its creation. Essentially, the signature of a message should not disclose any extra details about the attribute set determined by the access structure, except for confirming its adherence to the access structure. Our system is considered computationally private if an adversary A, operating within polynomial time, cannot achieve victory in the Exp^{ano} security game with non-negligible advantage. The Exp^{ano} security game is formally outlined in the interaction between an adversary A and a challenger C as follows:

Setup: the adversary A chooses an attributes universe U where $|U| = n$ and sends it to the challenger C. Then, the challenger runs CreateAttributeAuthority and KeyGetFuncGenerator algorithms to generate master key MK, $KeyG_{FN}$., then gives the PK and the $KeyG_{FN}$ to the adversary A.

Challenge Phase: the adversary A chooses an access policy T, two attribute sets A_{S1} and A_{S2} satisfying the threshold access policy such that $|A_{S1} \cap S| = |A_{S2} \cap S| = t$, a random value RV and a message M and sends them to the challenger C. Afterwards, C picks a random bit b: $b \in \{0; 1\}$ and executes the User.Keygen and computes a signcryption S_{RVb} by running the algorithm Proxy.Signcrypt(PK, RV_b, SK_{Eb}, T) $\rightarrow CT_{RVb}$. The signcrypted random value CT_{RVb} is sent to the adversary A as a challenge message.

Guess: the adversary outputs a bit b' and wins the game if $b' = b$. The advantage of the adversary A in the above games defined as $Adv\left[Exp^{aMSE-CDH}\right] = |Pr\left[b = b'\right] - \frac{1}{2}|$.

Definition 5: A threshold attribute-based signcryption scheme is computationally private if Adv $[Exp^{ano}]$ is negligible with respect to the security parameter κ, for any adversary A running in a polynomial time.

C. OUR CRYPTOGRAPHIC CONSTRUCTION

This section presents the list of our proposed cryptographic algorithms. To ease of explanation, we define the notations used in our model as shown in Table 1.

Our cryptographic model encompasses of five major phases including System Setup, Key Generation, Signcryption, Designcryption, and Policy update.

- **Phase 1: System Setup**

There are two algorithms in this phase including CreateAttributeAuthority and KeyGetFuncGenerator. The first algorithm is run by the AA for creating the attribute authority (AA) system responsible for issuing the attribute-based secret key to the users while the latter algorithm is for generating an initial key get function ($KeyG_{FN}$) used to generate the symmetric key for data encryption. The algorithmic details of these two operations are described as follows.

(1) $CreateAttributeAuthority\,AA.Initial(1^{\kappa}) \rightarrow (PK, MK)$. The algorithm accepts a security parameter κ as its input and generates public key PK and master key MK as outputs. It selects a tripple bilinear group G_0, G_1 and G_T of prime order p, a collusion resistant hash function H: {0,1}* $\rightarrow$ (Z / pZ)* and a specifies an encodinng function τ: $U \rightarrow$ (Z / pZ)*, where $|U|$ = m and U is an attribute universe, m is number of attribute. then selects generator g of G_0 and h of G_1. Next choose a set $D = \{d_1, d_2, \ldots, d_{i-1}\}$, where $i <=$ m – 1. pairwise different elements of (Z / pZ)*. Then, it chooses two randomα, $\gamma \in (Z / pZ)^*$. The public key is computed as:

$$PK_k = \tau \left\{ G_0, G_1, G_T, \hat{e}, \{h^{\alpha\gamma^i}\}_{\{i=0,\ldots,2m-1\}}, \mathrm{D}, \tau, u = g^{\alpha\gamma}, e\left(g^{\alpha}, h\right), H \right\}.$$

and the master key MK is (g, α , γ) where α , γ are two random from values (Z / pZ)*.

TABLE 1. Notation used in our model.

Notation	Description
AA_k	The attribute authority k
DUid	A data user *DUid* using mobile to access the outsourced data.
DOid	A data owner *DOid* using mobile to signcrypts the outsourced data.
S_{AAk}	A set of attributes in the CP-ABE setting.
S_{DO}	A set of attributes issued to DO.
S_{DU}	A set of attributes issued to DU.
HS_{DU}	A set of hashed values of the attributes issued to user *DUid.*
HS_{DO}	A set of hashed attributes issued to data owner *DOid*.
SK_{DU}	A CP-ABE secret key issued to DU.
SK_{DO}	A CP-ABE secret key issued to DO.
PK	An AA's public key.
MK	An AA's Master Key.
$KeyG_{FN\_id}$	Keyget function used to compute a symmetric key.
SymKey	An 256-bit AES Symmetric key.
RV	A 256-bit string random value.
RV_C	A 256-bit string random value concatenated with *H(M)* and ID of $KeyG_{FN}$
RV_{new}	A new 256-bit string random value generated used in re-encryption phase.
M	A message or data outsourced in the cloud storage.
H(M)	The hash value of message *M* signcrypted with *RV*.
CT_M	The ciphertext of a message *M*.
CT_{RV}	The ciphertext of the random value *RV*.
CT_{RV_new}	The new ciphertext of *RV* produced *in the* re-encryption phase.
$Enc\text{CT}_{\text{List}}$	The list of encrypted *CTs*.
RefKeyId	The reference key value id corelated with the SK_{DU} used by the proxy.
T	A CP-ABE based access policy where the all attributes value are hashed.

(2) $KeyGetFuncGenerator(randomFunc()) \rightarrow KeyG_{Fn}$

The below function describes how the KeyG_{Fn} is generated.

```
Def keyGetFuncGenerator(randomFunction()): KeyG_Fn
{
        RandomValueList R =randomFunction()
        return Def function(string RV): string {
          k = RV
          For r_i in R:
              k = k⊕ r_i
          End for
          return k
        }
}
```

The randomFunc is produced based on the cryptographically secure pseudorandom number generator (CSPRNG). It outputs the 256-bit *RV.* The following code illustrates the procedure of randomFunc()

```
Def randomFunc(): SecretKey {
    int keyBitSize = 256
    KeyGeneratorinit(keyBitSize,
    SecureRandom());
    SecretKey R = KeyGenerator.generateKey();
    return R;
}
```

When the $KeyG_{FN\_id}$ is obtained, DO encrypts it by using DUs' public keys before it is forwarded to all DUs. In our scheme, we assume that all users are required to have a key pair and the X.509 cerificate issued by the certification authority. Upon the receipt of the encrypted $KeyG_{FN\_id}$, the DU decrypts it by using his private key to obtain the $KeyG_{FN\_id}$ which will be used for the data decryption.

- **Phase 2: Key Generation**

This phase is run by the AA to generate the user secret key and data owner secret key which are generated through the User.KeyGen and DataOwner.KeyGen algorithm respectively.

(1) $User.KeyGen(PK, MK, S_{DU}) \rightarrow (SK_{DU})$.

AA executes the User.KeyGen algorithm using *PK, MK,* and a set of attributes assigned to the *DU* as inputs. The algorithm then outputs the secret of the data user SK_{DU}.

For any subset $Si \subset U$ the algorithm first chooses a random $r \in (Z/pZ)^*$. Then it generates the key as:

$$SK = (\{g^{\frac{r}{\gamma+\tau(a)}}\}_{a\epsilon S}, \{h^{r\gamma^i}\}_{i=0,\ldots,m-2}, h^{\frac{r-1}{\gamma}})$$

When the DU gets SK_{DU}, he then forwards it to the proxy via the SSL communication. Then, the proxy encrypts the SK_{DU} with its public key and keeps it in its server. Hereafter, the proxy system generates the *RefKeyId* for the key retrieval and return to DU, he uses in the future decryption stage.

(2) $DataOwner.KeyGen(PK, MK, S_{DO}) \rightarrow (SK_{DO})$.

AA runs the DataOwner.KeyGen by taking as inputs *PK*, *MK*, and a set of attributes issued to the DO. It outputs the secret of the data owner, SK_{DO}. The process of creating *SK's DO* is the same as the User.KeyGen algorithm. Finally, the algorithm produces the SK_{DO} which is sent to the DO.

- **Phase 3: Signcryption**

This phase is run by the *DU* and the proxy to signcrypt message *M*. In this phase, there are three major algorithms including DO.Encrypt, Proxy.Signcrypt and Proxy.CiphertextIndexingGen.

(1) $DO.Encrypt(KeyG_{Fn\_id}, M) \rightarrow CT_M$

DO runs $KeyG_{Fn\_id}$ to get the symmetric key to encrypt the message or data (*M*). Then, the algorithm outputs ciphertext of the data, CT_M. The below function shows how the CT_M. is produced:

```
Def DO.Encryption(KeyG_Fn_id , M): CT_M {
    RV = randomFunc()
    SymKey = KeyG_FN_id(RV)
    CT_M = AES_Encrypt(M, SymKey)
    H(M) = Hash(M)
    RV_C = RV.concat(H(M))
    Store and return CT_M
}
```

As shown in the above function, DO (1) runs randomFunc to compute the initial RV and takes it into the $KeyG_{FN\_id}$ to produce a $SymKey$; (2) encrypts M by using the $Symkey$ to generate the ciphertext CT_M; and (3) computes hash value of the message $H(M)$. Hereafter, DO concatenates RV with ID of $KeyG_{FN}$ and $H(M)$ as RV_C then sends them to the proxy for signcryption.

(2) $Proxy.Signcrypt(PK, RV_C, SK_{DO}, T) \rightarrow CT_{RV}$.

The proxy runs this algorithm by taking as inputs PK, RV_C, SK_{DO},and T to signcrypt the RV_C and it outputs the ciphertext of RV_C, CT_{RV}. We give the details of how the ABSC-Signcrypt algorithm works. First, the algorithm uses the secret key of the data owner SK_{DO} and the aggregate algorithm $Aggreg$ to computes T_1 as follows.

$$T_1 = Aggreg\left(\left\{g^{\frac{r_{DO_{id}}}{\gamma+\tau(a)}}, \tau(a)\right\}_{a\epsilon HS_{DO}}\right) = g^{\frac{r_{DO}}{\prod_{a\epsilon HS_{DO}}(\gamma+\tau(a))}}$$

Then the algorithm defines the polynomial $P_{(DO_{id},S)}(\gamma)$

$$P_{(DO_{id},S)}(\gamma) = \frac{1}{\gamma}\Big(\prod_{a\epsilon S\cup D_{n+1-1-s\backslash DO_{id}}}(\gamma+\tau(a)) - \prod_{a\epsilon S\cup D_{n+1-1-s\backslash DO_{id}}}\tau(a)\Big)$$

Next, the algorithm takes as input $\{g^{\frac{r}{\gamma+\tau(a)}}\}_{a\epsilon S}$ from SK_{DO} for signcryption.

$$B = h^{r_{DO_{id}}P_{(HS_{DO},S)}(\gamma)/\prod_{a\epsilon S\cup D_{n+1-1-s\backslash DO_{id}}}\tau(a)}$$

The algorithm computes hash value of RV_C, $H(RV_C)$and generates the signature $\sigma = \sigma_1, \sigma_2, \sigma_3$ as:

$$\sigma_1 = T_1 \cdot g^{g^{\frac{H(RV_C)}{\prod_{a\epsilon HS_{DO}}(\gamma-\tau(a))}}}$$

$$\sigma_2 = \{h^{r\gamma^i}\}_{i=0,\ldots,m-2} \cdot B \cdot h^{H(RV_C)P_{(DO_{id},S)}(\gamma)/\prod_{a\epsilon S\cup D_{n+t-1-s\backslash DO_{id}}}\tau(a)}$$

$$\sigma_3 = h^{\alpha H(RV_C)}$$

Finally, the algorithm picks a random $k \in (Z/pZ)^*$. and computes C_1, C_2, C_3 as:

$$C_1 = (g^{\alpha\cdot\gamma})^{-k}$$

$$C_2 = h^{k\alpha\cdot\prod_{a\epsilon S}(\gamma+\tau(a))\prod_{d\epsilon D_{n+t-1-s}}(\gamma+d)}$$

$$C_3 = \hat{e}(g,h)^{\alpha\cdot k}\cdot\hat{e}(g,h)^{\alpha\cdot H(RV_C)}\cdot RV_C = K\cdot RV_C$$

Thus,

$$CT_{RV_C} = (C_1, C_2, , C_3, , \sigma_1, \sigma_2, \sigma_3, P_{(DO_{id},S)}(\gamma), B)$$

(3) $Proxy.CiphertextIndexingGen(CT_{RV}, CT_M, T) \rightarrow EncCT_{List}$.

The proxy runs this function which takes as inputs the ciphertext of RV, CT_{RV}, the ciphertext of the message, CT_M, and access policy T to generate the access policy indexing. Once the index data has been generated, it then creates the relation between the access policy index and the encrypted ciphertext list. This indexing scheme is beneficial for optimizing the ciphertext retrieval process when there is a policy update case which requires the ciphertext re-encryption. This function consists of the following procedures.

(1) Stores CT_{RV}, CT_M on the cloud storage and gets the location of CT_{RV}, CT_M.

(2) Calculates the indexing data of access policy T by $index = Hash(T)$, In our system, the access policy can be defined as the string. For example: "(Doctor and Doctor_Id = D001) or Patient_Id = HN111". Therefore the index is generated by common Hash(String).

(3) Searches existing access policy using the index. If the policy index exists, the proxy adds CT_{RV}, and CT_M location on the $EncCT_{List}$. Otherwise, the proxy adds a new index to the access policy index table and creates a new $EncCT_{List}$. The access policy index data structure is presented in Table 2 as follows.

TABLE 2. Access policy index.

Access policy index	Encrypted ciphertext list
b89eaac7e…	$EncCT_{List\_1}$
bfd28043…	$EncCT_{List\_2}$
ffe2d567a…	$EncCT_{List\_3}$

Then, the $EncCT_{List}$ is created in JSON format. The example of $EncCT_{List}$ is presented as follows.

```
{
  "Index":" b89eaac7e",
  "Version": 1,
  "CTlist": [
        {
          CT_RV: /store/ctrv1.enc
          CT_M: /store/ctm1.enc
        },
        {
          CT_RV: /store/ctrv2.enc
          CT_M: /store/ctm2.enc
        }
      ]
}
```

In summary, our proposed Signcryption algorithm enables the separation of duty of data encryption and signing function between data owner and the proxy. The DO only deals with the data encryption using the symmetric encryption while the signcryption operation having more expensive cost is offloaded to the proxy. Finally, the proxy generates the indices of final ciphertexts before they are stored in the cloud storage.

- **Phase 4: DeSigncryption**

This phase is run by the proxy and the DU to designcrypt message M. It consists of Proxy.Decrypt algorithm and User.Decrypt algorithm run by the proxy and DU respectively.

(1) Proxy.Decrypt(PK, $RefKeyId$, CT_{RV}) $\rightarrow$ (RV_C).

The proxy runs the algorithm by taking PK, RefKeyId, and CT_{RV}. It returns the RV_C as the output. Then, the decryption process is done through the following steps.

a. DU forwards the RefKeyId to the proxy.
b. The proxy retrieves CT_{RV} and CT_M from the cloud.
c. The proxy decrypts the encrypted SK_{DU} with its private key.
d. The proxy runs the below ABSC-designcryption algorithm by taking as inputs SK_{DU} and CT_{RV}.

$$RV_C = ABSC - Designcrypt(CT_{RV}, SK_{DU})$$

Then, the function outputs the RV_C. Then, RV_C and CT_M are forwarded to the user.

(2) User.Decrypt(RV_C, CT_M,) $\rightarrow$ (M).

DU runs this function by taking as inputs the RV_C and CT_M for computing the $SymKey$ used to decrypt the ciphertext of message M. The detailed procedures are described below:

a. Extracts RV, $H(M)$, and ID of $KeyG_{FN}$ from RV_C and uses ID of $KeyG_{FN}$ to select $KeyG_{FN\_id}$.
b. Applies RV into $KeyG_{FN\_id}$ to compute the Symkey through the following function.
$Symkey = KeyG_{FN\_id}(RV)$
c. Decrypts CT_M to obtain M.
$M = AESDecrypt(CT_M, Symkey)$
d. Calculates the hash value of the message $H(M)$ to verify M as follows.
$MessageCorrectly = H(M)_{DU} == H(M)$

The process of ABSC-Designcrypt function is done through the following steps. First, the algorithm verifies the CT_{RV} that it has been correctly signed by the legitimate DO. The function is defined as:

$$SS = \hat{e}\left(u^{-1}, \sigma_2\right) \cdot \hat{e}\left(\sigma_1^{\frac{1}{U_1}}, h^{\alpha \cdot \prod_{a \epsilon S \cup D_{n+t-1-s}} (\gamma + \tau(a))}\right)$$

$$\cdot \hat{e}\left(g^{\alpha}, h\right)^{H(RV_C)} = \hat{e}(g^{\alpha}, h)$$

where $U_1 = \prod_{a \epsilon S \cup D_{n+t-1-s}/HS_{DO}} \frac{\gamma + \tau(a)}{\tau(a)}$

Second, for all $a \epsilon HS_{DU}$, the algorithm aggregates the data user attributes as:

$$T_2 = Aggreg\left(\left\{g^{\frac{r_{DU_{id}}}{\gamma + \tau(a)}}, \tau(a)\right\}_{a \epsilon HS_{DU}}\right) = g^{\frac{r_{DU}}{\prod_{a \epsilon HS_{DU}} (\gamma + \tau(a))}}$$

Then, the algorithm uses the aggregated secret key T_2 and $\{h^{r\gamma^i}\}_{i=0,\ldots,m-2}$ from SK_{DU} to computes:

$$\left[\hat{e}\left(C_1, h^{r_{DU} P_{(HS_{DO}, S)}(\gamma)}\right) \cdot \hat{e}(T_2, C_2)\right]^{\frac{1}{U_1}} = e(g, h)^{(K \cdot \alpha) \cdot r_{EU}}$$

Next the algorithm deduces the deciphering key DK as:

$$DK = \hat{e}(g, h)^{\alpha \cdot k} \cdot \hat{e}(g, h)^{\alpha \cdot H(RV_C)}$$

Finally, the algorithm recovers the RV by computing:

$$RV_C = C_3/K$$

In essence, our proposed Designcryption algorithm minimizes the decryption burden on mobile users by shifting the primary pairing operation of ABSC to the proxy, while allowing the user to calculate the symmetric key and utilize it for decrypting the ciphertext with significantly reduced cost.

- **Phase 5: Policy Update**

In this phase, the data owner executes the algorithm as follows:

PolicyUpdate(T, T', SK_{DO}, PK) $\rightarrow$ (CTs_{RV_new}).

A new access policy denoted as T' is taken as input. The policy update procedure consists of three steps:

a. DO sends the request to update the policy to the proxy. The request contains the old access policy (T), a new access policy (T'), PK, and SK_{DO}.
b. The proxy retrieves the encrypted CT_{RV} and calculates the searching index by hashing the before-updated policy (T) then using Hash(T) to search all encrypted $EncCTList$ from the access policy index data.
c. The proxy runs the Re-encryption function presented below to re-encrypt all affected $EncCTList$ and stores the CTs_{RV_new} onto the cloud storage then update access policy index data.

To support the policy update of a high number of ciphertext, we introduced the indexing search algorithm to efficiently get the list of affected CT_{RV} and parallel programming by generating multiple threads to parallelize re-encryption requests. The algorithmic process is presented below.

Multiple thread re-encryption procedure

Input: SK_{DU}, Old access policy (T), New access policy (T), PK
Output: List of CT_{RV_new}(CTs_{RV_new})
Def searchEncryptedCiphertext(T):
    index = Hash(T)
    $EncCT_{List}$ = Select $EncCT_{List}$ where accessPolicyIndex = index
    Return $EncCT_{List}$
End Def
Def ReEncryption(SK_{DU}, SK_{DO}, T, T', PK, results):
    $EncCT_{List}$ = searchEncryptedCiphertext(T)
    For CT_{RV} in JSON($EncCT_{List}$-> $CTlist$):
        RV_C = ABSC_Designcrypt(SK_{DU}, CT_{RV})
        CT_{RV_new} = ABSC_Signcrypt(PK, SK_{DO}, RV_C, T')
        results.add(CT_{RV_new})
    End for
    Return results
End Def
N = List of CT_{RV} number
T_N = Thread number
numPerThread = N/ T_N
For t_n in T_N:
    t_n.execute(ReEncryption(SK_{DU}, SK_{DO}, CTs_{RV}[$t_{n.num}$: $t_{n.num}$ + numPerThread], T', PK, results))
End for
Return results

The above algorithm first takes as inputs SK_{DU}, old access policy T, new access policy T' and PK. Then it takes as input the access policy T to run the searchEncryptedCiphertext function for searching ciphertext list from the access policy index table. Then, the algorithm will create multiple threads to execute ReEncryption function to re-encrypt all affected ciphertext by assigning multiple affected ciphertexts to be executed by each thread.

Our proposed algorithm for re-encrypting ciphertexts, utilizing parallel programming, aids in optimizing the re-encryption cost, which happens to be the most substantial expense. This is because, with every update, all affected ciphertexts need to undergo re-encryption using the ABSC method. For instance, if there are n ciphertexts affected by a policy update T, the cost of re-encrypting each ciphertext depends on the number of affected ciphertexts. Parallelizing the re-encryption process for multiple ciphertexts, in contrast to sequentially re-encrypting each affected ciphertext, significantly enhances the efficiency of policy updates. This results in access control accurately representing the current state of data sharing with optimized efficiency in terms of time.

V. SECURITY ANALYSIS

In this paper, our security system is analyzed in a game-based model. As our scheme relies on CP-ABE and ABSC, a comprehensive security proof can be found in the original works of CP-ABE [23] and ABSC [24].

In cloud computing environment, we assume that data owners are fully trusted. The users are assumed to be dishonest, i.e., they may collude to access outsourced data. It is also assumed that the adversary can corrupt authorities only statically, but key queries can be made adaptively. The attack of the security can be done by an adversary requesting a key from the attribute authority. We conduct a security analysis of our proposed solution based on the proof of security models with the theorems defined below.

A. MESSAGE CONFIDENTIALITY

Theorem 1: There is no probabilistic polynomial-time adversary that can break the security of symmetric encryption / decryption algorithm and ABSC, with non-negligible advantage.

Proof: Suppose a group of probabilistic polynomial-time the adversary A has non-negligible advantage against our proposed scheme. We show how to build an adversary A that breaks ours scheme with non-negligible advantage.

Initialization: We consider the adversary A consisting of a group of data users with non-negligible advantage against our proposed scheme and that solves the augmented multi-sequence of exponents computational Diffie-Hellman aMSE-CDH problem [24], [25].

Setup: The adversary A gets the PK and $KeyG_{FN}$ from the challenger C. The challenger C runs CreateAttributeAuthority and KeyGetFuncGenerator algorithms; The CreateAttributeAuthority algorithm selects a *triple* bilinear group G_0, G_1 and G_T of prime order p, a collusion resistant hash function $H: \{0,1\}^* \rightarrow (Z/pZ)^*$ and a specifies an *encoding* function $\tau: U \rightarrow (Z/pZ)^*$, where $|U| = m$ and U is an attribute universe, m is number of attribute. then selects generator g of G_0 and h of G_1. Next choose a set $D = \{d_1, d_2, \ldots, d_{i-1}\}$, where $i <= m - 1$. pairwise different elements of $(Z/pZ)^*$. Then, it chooses two random $\alpha, \gamma \in (Z/pZ)^*$. The public key is computed as:

$$PK_k = \tau \left\{ G_0, G_1, G_T, \hat{e}, \{h^{\alpha\gamma^i}\}_{\{i=0,\ldots,2m-1\}}, \mathrm{D}, \tau, u = g^{\alpha\gamma}, e\left(g^{\alpha}, h\right), H \right\}.$$

And $KeyG_{Fn}$ computed as

randomValue = randomFunc()

$KeyG_{Fn}$ = keyGetFuncGenerator(randomValu).

then C sends the PK and the $KeyG_{Fn}$ to A.

Phase1: In this phase, A answers designcryption query. Suppose the algorithm issues private key query for a set of attributes S to adversary A, such that S cannot satisfy an access structure T^*. To answer the query, A makes a private key query for the set S to the challenger, which generates the secret key and $KeyG_{FN}$.

$$SK = (\{g^{\frac{r}{\gamma+\tau(a)}}\}_{a\epsilon S}, \{h^{r\gamma^i}\}_{i=0,\ldots,m-2}, h^{\frac{r-1}{\gamma}})$$

Then, it is forwarded to A.

Challenge:A submits two equal lengths of message M_0 and M_1 such that $|M_0| = |M_1|$ (both the messages are of equal length) and access policy T^* on which A need to be challenged to the challenger C. Thus, C select $b: b \in \{0; 1\}$ and generate and signcrypt the RV_b by set $C_3^* = \mathcal{T} \cdot RV_b$,choose a random value for encryption as $\mathcal{K}' = \mathcal{K}/\propto$ and computes $H(RV_b)$. Then, $C_2^* = h_0^{\mathcal{K}' \cdot g(\gamma)}$, $C_1^* = g_0^{\mathcal{K}' \cdot \gamma \cdot f(\gamma)}$ and $\sigma_3 = h_0^{H(RV_b)}$.While M_b will be encrypted by AES_Encryption(M_b, $KeyG_{FN}(RV_b)$), C forwards challenged ciphertext of random value and ciphertext of message to A

$$CT_{RV_b}{}^* = (C_1, C_2, , C_3, , \sigma_1, \sigma_2, \sigma_3, P_{(DO_{id},S)}(\gamma), B)$$
$$CT_{M_b} = \mathrm{Encrypt}(M_b, KeyG_{FN}(RV_b))$$

Phase 2. A does designcryption queries as in phase 1, with a condition that none of the key satisfies access structure T^*

Guess. Finally, Aoutputs a guess $b' : b \in 0, 1\}$.If $b' = b$, Challenger C answer 1,meaning that $\mathcal{T} = e(g_0, h_0^{\mathcal{K} \cdot f(\gamma)})$ Otherwise,Challenger C answer 0, meaning that $\mathcal{T}$ is a random element. and A concludes game by outputting b'. the advantage of adversary A against our scheme is

$$\Pr\left[Exp^{aMSE-CDH}\right] = |\Pr\left|b' = b\right| real] - \Pr|b' = b|random]|$$

When the event real occurs, then A is playing a real attack and therefore $|\Pr\left|b' = b\right| real] - \frac{1}{2}]| = \Pr\left[Exp^{Conf}\right]$. During the random event, the view of A is completely independent of the bit b; in this case the probability $\Pr\left[b' = b\right] = 1/2$ Then,

$$\Pr\left[Exp^{aMSE-CDH}\right] \geq \frac{1}{2} Pr\left[Exp^{Conf}\right]$$

Hence, A has nonnegligible advantage against our scheme, which completes the proof of the theorem.

B. CIPHERTEXT UNFORGEABILITY

In the security game, the adversary A attempts to forge a valid signature by performing following steps. First, the adversary A chooses a set of signcryption attributes for access policy T. Subsequently, the adversary A requests the execution of the User.KeyGen queries. Additionally, the adversary A requests for the signcryption of a random value RV and a message M under different signcryption attribute sets and access policy by executing the Query Phase. Here, the adversary A aims to gain insights into secret values from the User.KeyGen and Proxy.signcrypt algorithms. Then, the adversary A attempts to generate a valid signcryption with respect to the challenge access policy T. To achieve this, A must solve the aMSE−CDH problem for proving that he has the required attributes to satisfy the access policy T. Moreover, thanks to the random values added to the generated signature, it is important to note that the inclusion of random values in the generated signature prevents A from acquiring information about the secret values utilized in key generation and the signcryption process. The subsequent discussion provides evidence that our construction is resistant to forgery in the face of chosen message attacks, as outlined in Theorem 2.

Theorem 2: The scheme is adaptive-message unforgeable under chosen message attacks for a universe U of m attributes, challenge signcryption attribute set S^* and challenge access policy T^*, In this context, there exists a solver for the aMSE-CDH problem, such that $\Pr\left[Exp^{aMSE-CDH}\right] \geq \Pr\left[Exp^{unf} = 1\right]$ aMSE-CDH problem holds.

Proof: We consider the same setting for the setup phase as detailed in the message confidentiality security game.

Query Phase: the adversary A can request the key extraction operations User.KeyGen and Proxy.Signcrypt multiple times. For each KeyGen query i, the adversary A seeks the extraction of a subset A_A from the challenger C, as detailed in the message confidentiality security game setup phase.

The adversary A can adaptively request a SignCrypt query i, while considering the signcryption attribute set S_{DU}, random value RV_i and message M_i. the challenger C executes User.Keygen to generate secret key SK and executes Proxy.Signcrypt to signcrypt the RV_i and M_i.Then, the SK is forwarded to the adversary A.

Forgery Phase: during this phase, A outputs a signcrypted random value CT_{RV^*} and a signcrypted message CT_{M^*}, with respect to the threshold challenge access policy T. For this purpose, A can compute C_1, C_2, C_3 and σ_3, based on public parameters PK and the selected random k and $H(Mi)$. In the end, A must calculate valid σ_1 and σ_2 to succeed in the unforgeability security game. Consequently, A is required to solve the aMSE-CDH problem to demonstrate possession of the necessary attributes for satisfying access policy T. Therefore, A shares the same advantage as the Exp^{Conf} message confidentiality security game.

C. VERIFIABILITY

Theorem 3: Our scheme is verifiable if the Diffie–Hellman (DL) assumption holds in prime order multiplicative cyclic groups.

Proof: Let the adversary A who attacks our scheme with nonnegligible advantage and the challenger C which can solve the Diffie–Hellman (DL) problem in the prime order multiplicative cyclic group system with nonnegligible advantage. The detailed interaction is described as follows.

Setup. The challenger C runs CreateAttributeAuthority and KeyGetFuncGenerator algorithms to generate master key MK, $KeyG_{FN}$. Then, C gives the PK and the $KeyG_{FN}$. to the adversary A.

Challenge: The adversary A makes secret key SK queries corresponding to sets of attributes $S_1, S_2, \ldots, Sq$. The challenger responds the adversary by executing the User.Key Gen algorithm and response SK to A.

Output: The adversary A provides access structure T^* such that the sets $S_1, S_2, \ldots, Sq$ satisfies the access structure T^* to the challenger C for signcrypting the message M and random value RV. Consequently, A request the challenger C to execute a designcryption phase to get RV and M. The correctness analysis of the proposed access control scheme demonstrates that any data user who possesses a set of attributes S_{DU} satisfying the access control policy T can successfully verify and decrypt the signcrypted message associated with that policy.

We denote by COR and the quantity $= \hat{e}\left(u^{-1}, \sigma_2\right) \cdot \hat{e}\left(g^{\alpha}, h\right)^{H(RV)\cdot\left(1-U_1-\frac{1}{U_1}\right)} \cdot \hat{e}\left(\sigma_1^{\frac{1}{B1}}, h^{\alpha\prod a\epsilon S\cup D_{n+t-1-s}(\gamma+\tau(a))}\right)^{H(RV)\cdot\left(1-U_1-\frac{1}{U_1}\right)}$, where $\mathrm{U}_1 = \prod a\epsilon S\cup D_{n+t-1-s\backslash As}\frac{\gamma+\tau(a)}{\tau(a)}$

The data user can verify that the data owner has correctly signed the random value RV as:

$$\begin{aligned}\mathrm{COR} &= \hat{e}\left(g^{-\alpha\gamma}, h^{\frac{r_{DO}-1}{\gamma}\cdot h^{\frac{(rDO+H(RV)*P_{(HS_{DO},S)^{\gamma}})}{\prod a\epsilon S\cup D_{n+t-1-s/A_S(\tau(a)+\gamma)}}}}\right)\\ &\cdot \hat{e}(g^h, h)^{H(RV)\cdot\left(1-U_1-\frac{1}{U_1}\right)}\\ &\cdot \hat{e}\left(g^{\frac{(rDO+H(RV))}{\prod_{a\epsilon DO}(\gamma+\tau(a))\cdot\prod a\epsilon S\cup D_{n+t-1\backslash HS_{DO}^{\tau(a)}}}}, h^{\alpha\prod a\epsilon S\cup D_{n+t-1-s}(\gamma+\tau(a))}\right),\\ &= \hat{e}\left(g^{\alpha}, h\right)\end{aligned}$$

Next the data user aggregates his secret key to derive the deciphering key K. Finally, the data user can retrieve the random value RV as

$$RV = \frac{C3}{\hat{e}\,(g,h)^{\alpha\cdot(H(RV)+k)}} = \frac{C3}{K}$$

Then, the data user computes $H(RV)$ using the retrieved RV, to verify that σ_3 is equal $h^{\alpha\cdot H(RV)}$.

D. PRIVACY

Theorem 4: Our scheme achieves signer privacy.

Proof: The anonymity game begins when the challenger C runs CreateAttributeAuthority and KeyGetFuncGenerator algorithms to generate master key MK, $KeyG_{FN}$., then gives the PK and the $KeyG_{FN}$ to the adversary A.

Subsequently, the adversary A chooses two attribute sets A_{S1} and A_{S1} satisfying the threshold access policy T and sends them to the challenger C. The challenger generates the secret keys related to the sets of attributes A_{S1} and A_{S2} by executing the User. Key Gen algorithm as follows:

$$SK_{E1} = (\{g^{\frac{r}{\gamma+\tau(a)}}\}_{a\epsilon S1}, \{h^{r\gamma^i}\}_{i=0,\ldots,m-2}, h^{\frac{r-1}{\gamma}})$$
$$SK_{E2} = (\{g^{\frac{r}{\gamma+\tau(a)}}\}_{a\epsilon S2}, \{h^{r\gamma^i}\}_{i=0,\ldots,m-2}, h^{\frac{r-1}{\gamma}})$$

Additionally, adversary A outputs a challenge message M and random value RV and asks the challenger C to generate the signcryption of RV using one of the secret keys SK_{E1} or SK_{E2}. Consequently, the challenger selects a random bit b: $b\in \{0; 1\}$, and computes a signcryption S_{RVb} by running the algorithm Proxy.Signcrypt(PK, RV_b, SK_{Eb}, T) $\rightarrow$ CT_{RVb}. The challenger's ability to generate a valid signcryption on the random value RV is crucial. To establish the scheme's privacy, the task is to demonstrate that signcrypted random values created using SK_{E1} or SK_{E2} are identical. Using the private key related to $SKEb$, the generated signcryption $S_{RVb} = (C_1, C_2, C_3, \sigma_1, \sigma_2, \sigma_3)$. Then, the adversary A verifies the signature, by computing:

$$S^*_{RVb} = \hat{e}\left(g^{-\alpha\gamma}, h^{\frac{r_{DO}-1}{\gamma}} \cdot h^{\frac{(rDO+H(RV)*P_{(HS_{DO},S)^\gamma})}{\prod a\epsilon S\cup D_{n+t-1-s/A_{S(\tau(a)+\gamma)}}}}\right)$$
$$\cdot \hat{e}(g^h, h)^{H(RV)\cdot\left(1-U_1-\frac{1}{U_1}\right)}$$
$$\cdot \hat{e}\left(g^{\frac{(rDO+H(RV))}{\prod a\epsilon DO\,(\gamma+\tau(a))\cdot\prod a\epsilon S\cup D}_{n+t-1\backslash HS^{\tau(a)}_{DO}}}, h^{\alpha\prod a\epsilon S\cup D_{n+t-1-s}(\gamma+\tau(a))}\right)$$
$$= \hat{e}\left(g^{\alpha}, h\right)$$

Since $|A_{Sb} \cap S| = |A_{S1} \cap S| = |A_{S2} \cap S| = t$, we can prove that the signature generated using the set of attributes A_{S1} (in other words, using the secret key SK_{E1}) is similar to the signature generated using the set of attributes A_{S2} (using the secret key SK_{E2}). Hence, the challenger A cannot know the set of attributes used to generate the signature. Thus, our proposed attribute based signcryption scheme is computationally private.

VI. EVALUATION

To evaluate our proposed scheme, we performed the comparative analysis by comparing the functional features and the computation cost of our scheme and three related works supporting outsourced cryptographic operations and the policy update in the CP-ABE based access control. In addition, we did the experiments to demonstrate the performance of major cryptographic operations of our scheme and related works.

A. FUNCTIONALITY ANALYSIS

Table 3 presents a comparison of the functional features between our proposed scheme and several related works, including Li et al. scheme [20], Guan et al. scheme [21], Belguith et al. scheme, and Yu et al. scheme [31]. The comparison is based on three key aspects: policy update method, outsourced cryptographic operations, and the support of MCC. Policy update method refers to how the ciphertext is altered in response to a policy update. Generally, this can be achieved through ciphertext updates, allowing either the data owner or the cloud to perform the necessary computations, or through proxy re-encryption, which delegates all re-encryption tasks to the proxy. Both methods can be implemented in any Access Control applications that use Attribute-Based Encryption (ABE).

TABLE 3. Functionality comparison.

Scheme	Policy Update Method	Outsourced		Mobile/Edge Cloud
		Encryption/Signcryption	Decryption/ DeSigncryption	
[21]	*CT Update*	*x*	*x*	*x*
[22]	*CT Update*	*x*	✓	✓
[23]	*CT Update*	*x*	✓	✓
[32]	*Not Supported*	✓	✓	✓
Ours	*PRE*	✓	✓	✓

The outsourced cryptographic operations function gauges the extent of outsourcing attribute-based encryption/ signcryption and decryption/designcryption. Complete outsourcing of both operations indicates a significant reduction in computational load for both data owners and data users. Lastly, the third function determines whether the proposed cryptographic operations are suitable for execution in a Mobile-Edge Cloud (MEC) environment.

As shown in TABLE 3., the works supporting policy update chosen to compare in this paper are based on both PRE and CT update method. Our scheme, Guan et al. scheme [21], and Belguith et al. scheme [22] support the policy update in mobile or edge computing while Li et al.scheme [20] was implemented in the general cloud setting. Out of the selected works for comparison, only scheme [31] did not provide support for policy update. Notably, only our scheme supports policy update method based on PRE technique, which can reduce high-computation cost from the mobile/edge cloud to the proxy. Considering the outsourced cryptographic operations, scheme [20] did not support the outsourced encryption/decryption. In this scheme, the data owner needs to encrypt the data and sends the ciphertext to be stored in the cloud storage. Also, scheme [20] and [21] did not support

signcryption/designcryption on mobile/edge cloud whereas Belguith et al. scheme [22] support outsourced designcryption. Essentially, only Yu et al. scheme [31] and ours support both outsourced signcryption and designcryption.

TABLE 4. Computation cost comparisons.

Scheme	Encryption/Signcryption cost		Decryption/Designcryption cost		Policy Update	
	Data owner	*Proxy*	*Data user*	*Proxy*	*Data owner*	*Proxy*
[21]	$(2\|T_A\| + 1)G_0 + G_m + G_T$	-	$(2 \mid A_u \mid) G_e + (2\|T_A\|+2) G_T$	-	$3Zp$	$3G_0 + Zp$
[22]	$(2\|T_A\| + 1)G_T + 4G_0 + AES$	-	$AES + G_0 + 3G_T + 2G_e$	$2\|A_u\|G_e + (2\|T_A\|+1)G_T$	$2G_0$	$3G_0 + 2G_T$
[23]	$(\|T_A\| + 1) G_0 + (3 + k - \|T_A\|)G_1 + 2G_T$	-	$G_0 + G_1 + G_H$	$6G_e + 3G_0 + G_1 + G_T$	N/A	$2\|T_A\|G_0 + (\|T_A\|+1)G_1 + G_e$
[32]	$AES + (2\|T_A\| + 14) G_0 + G_T$	$(2\|T_A\| + 2) G_0$	$AES + 2G_0 + 2G_T$	$7G_e + (2\|T_A\|+1) G_0$	-	-
Ours	$AES + XOR + G_H$	$(\|T_A\| + 6) G_0 + 2 G_T$	$AES + XOR + G_H$	$9G_e + (\|T_A\|+5)G_0 + 4G_T$	N/A	$9G_e + (2\|T_A\|+11)G_0 + 6G_T$

B. COMPUTATION COST ANALYSIS

Table 4. gives the comparative analysis of the computational overheads of our scheme and related works. Here, we give the following notations used to represent the computation cost.

$|A_U|$: Number of attributes that belongs to the data user.
$|T_A|$: Number of leaf nodes in the access control policy.
G_0: Exponentiation and XOR operation in group G_0.
G_1: Exponentiation and XOR operation in group G_1.
G_T: Exponentiation and XOR operation in group G_T.
G_e: Pairing operation in group G_0.
G_m: Multiplication operation in group G_0.
G_H: Hash operation of M
Z_p : The group $\{0, 1, \ldots, p\text{-}1\}$ multiplication modulo p.
XOR: XOR operation in 256bits data.
E_{KeyGFn}: The size of element in $KeyG_{FN}$.
AES: AES encryption/decryption operation.
E_{RV}: The size of random value *RV is 256bit.*
U: The attribute universe
K: The size of the attribute universe U

Table 4 illustrates the comparison of CP-ABE encryption in each scheme, which is influenced by the number of attributes involved in the access policy, resulting in exponentiation operations on G_0 and therefore the encryption cost is subject to the size of access policy. Our proposed scheme reduces encryption costs by using outsourced signcryption and encrypting only the symmetric or transformation key, resulting in computational efficiency. In scheme [20], and [22], the data owner needs to fully apply CP-ABE or ABSC operation since they did not support outsourced encryption. To effectively handle CP-ABE or ABSC for data encryption, the data owner must perform pairing and exponentiation operations, which can be resource-intensive and impractical to execute on constraint devices. In contrast, scheme [21], [31], and our proposed scheme utilize ABSC/CP-ABE to encrypt the symmetric key or transformation key, which generally have smaller size than the message. Importantly, only Yu et al. system [31] and our proposed method support both outsourced signcryption and designcryption while Belguith et al. scheme [22] supports partial outsourced designcryption. Particularly, in our design, we delegate symmetric encryption, hashing, and XOR operations exclusively to data owners, in contrast to scheme [21], [22], [31], which requires both the data owner and the data user to handle exponentiation operations on G_0 *and* G_T. This distinction highlights the computational efficiency and improved security of our approach.

Regarding the policy update cost, our proposed scheme and Belguith et al. [22] strategies eliminate data owner involvement in policy updates. However, scheme [22] requires the proxy to perform the ciphertext update through ABSC operation. Meanwhile Li et al. scheme [20] and Guan's schemes [21] require both the data owner and the proxy to update the key and ciphertext. Thus, their policy update cost is subject to full CP-ABE operation.

C. EXPERIMENTAL ANALYSIS

For the experimental analysis, we performed a comparison of the signcryption and designcryption times of our scheme, [22] and [31]. For the policy update, we conducted the experiments to compare the policy update time between ours, schemes [20], [21], and [22].

To set up the experiments, we utilized Open SSL as the core PKI service to generate key pairs for users and the proxy within our system. Additionally, we employed the core CP-ABE toolkit and Java Pairing-Based Cryptography [29] to simulate the cryptographic operations of schemes [20], [21], [22], and [31]. The tests were conducted on CP-ABE container running on Google Cloud platform, 1 vCPU and 2048MB of RAM. For the mobile device, we used Samsung Galaxy A12 Octa-core device with 4 × 2.35 GHz CPU and 4 GB of RAM, running on Android 11.0 (Red Velvet Cake) installed with Java Pairing-Based Cryptography and CP-ABE toolkit. In our experiment, we used the CP-ABE container to execute all the encryption/signcryption operation of our scheme and related works while the decryption/designcryption performance was done on a mobile device Samsung Galaxy A12.

Our performance evaluation was designed to compare the major cost of ABSC operations and policy update of our

scheme and related works. To this end, we first conducted the experiments to measure the signcryption and designcryption performance. Then, experiment for measuring the performance of policy update is provided. Regarding the experiment setting. we varied the file size within the range of 10 KB to 320 KB, using an access policy consisting of 10 attributes and a user's secret key containing 5 attributes. In addition to assessing performance by varying file size, we also conducted evaluations by varying the number of attributes contained in the access policy. We employed a 40-KB file with 5 attributes contained in the user's secret key, while the number of attributes in the policy ranged from 10 to 80. To measure the cost of policy updates, we increased the number of ciphertexts that required re-encryption, considering a range of 500 to 16,000 files. The extent to which encrypted files needed to be re-encrypted depended on the number of files affected after the policy update. Each experiment was repeated 20 times, and the average response time was calculated for display in the graphs.

Figures 3a and 3b exhibit the signcryption and designcryption performance of our scheme and related works based on the varying file size respectively. Figures 4a and 4b illustrate how performance varies with changes in the number of attributes in the access policy. Figure 5 illustrates the policy update time when updating policies by measuring the time taken for re-encryption when the number of ciphertexts increases. In Figure 6, we compared the policy update time of our proposed scheme when the proxy utilized our proposed indexing technique and without the index.

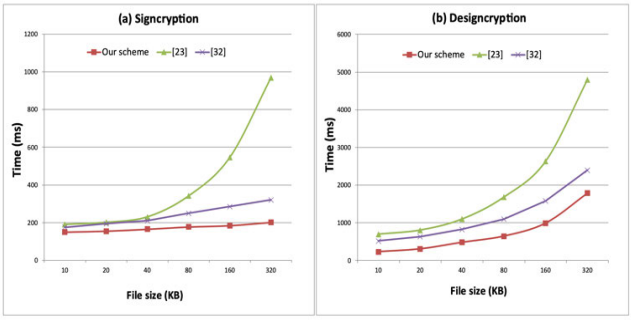

FIGURE 3. Signcryption time with varying file size.

As can be seen from Figure 3 and Figure 4, the graphs show that our scheme took least cost of both signcryption and designcryption with varying file size and number of attributes in the policy.

In our proposed scheme, the data undergoes an initial encryption process using symmetric encryption, and the random value used to derive the symmetric key is then encrypted using the ABSC (Attribute-Based Signcryption) method. In contrast, in scheme [22], CP-ABE and ABSC are employed for data encryption and decryption, respectively. When evaluating the cost of designcryption at the user's end, our scheme involves only key transformation computations and XOR operations, incurring no additional

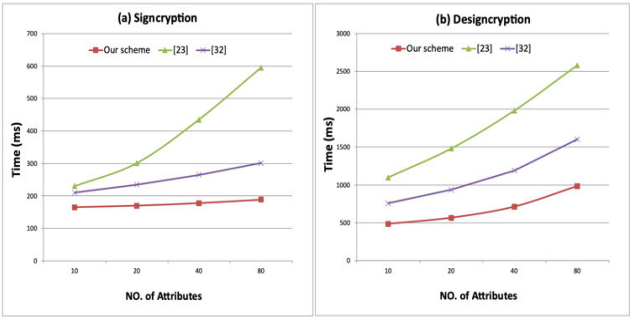

FIGURE 4. Designcryption time with varying no. of attributes.

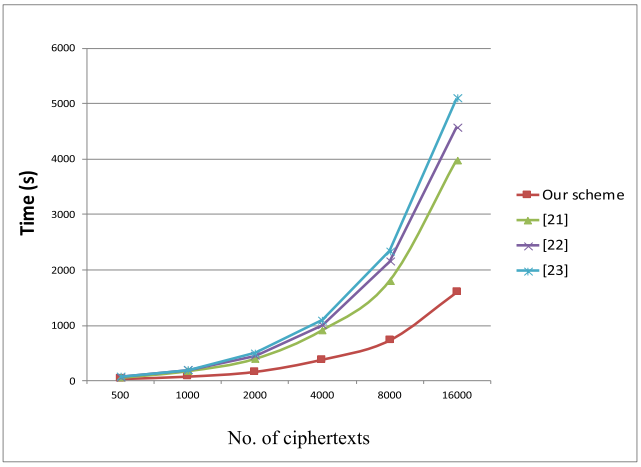

FIGURE 5. Policy update time.

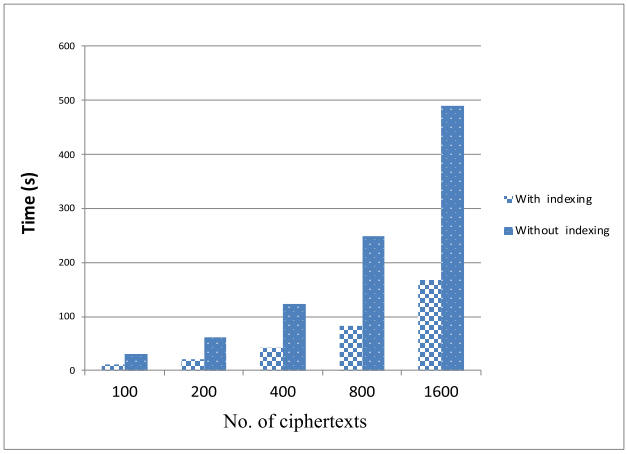

FIGURE 6. Comparison of policy update cost between using indexing and without indexing.

ABSC-related costs. However, in schemes [22] and [31], there's a need for partial CP-ABE or ABSC computations at the mobile user's end to perform the designcryption of the data. This results in additional computational overhead for the user, making our scheme more efficient in comparison. In addition to the signcryption and desirncryption cost, we also provide the evaluation of the policy update

performance of our proposed scheme and related works. The process time for policy updates is non-trivial in any attribute-based access control system as it represents a substantial portion of the overall cost involved in updating all affected ciphertexts according to the current state of the access control policy. The importance of policy update performance lies in its ability to gauge the efficiency of access control in a dynamic data sharing environment.

From Figure 5, the graphs clearly demonstrate that our proposed scheme exhibits the least cost for policy update operations. This significant advantage can be attributed to the use of the proxy re-encryption technique, where the proxy only needs to perform re-encryption on the CT_{RV} (ciphertext of the random value) which has a much smaller size compared to the original message. Furthermore, our proposed indexing search algorithm enhances the retrieval of affected ciphertexts during policy updates. This indexing search capability further contributes to the overall efficiency of the policy update process.

Figure 6 shows the comparison of the policy update processing time taken by the proxy with our proposed index searching algorithm and without index searching algorithm. With our proposed strategy, the major costs of re-encryption are CP-ABE designcryption and re-signcryption cost on the proxy side. Clearly, the proxy was offloaded with all policy updating operations including searching operation for list of policy update and designcrypting the ciphertexts and re-signcrypts the ciphertext of the random value while the data owner only prepares policy update information. The performance depicted in the graphs demonstrates that our proposed technique is highly practical and well-suited for evolvable access control in a MCC environment.

VII. CONCLUSION AND FUTURE WORK

We have introduced a mobile-cloud-based access control system that supports outsourced signcryption, designcryption, and policy updates. Our approach utilizes the proxy outsourcing model to handle the resource-intensive CP-ABE signcryption and designcryption tasks. Our scheme delivers a dual benefit of full outsourcing functionality, alleviating the responsibilities of both data owners and data users, while also achieving superior computational efficiency when compared to existing solutions. Additionally, our outsourced policy update model efficiently manages the costs associated with ciphertext re-encryption at the data owner's end, streamlining processes and enhancing overall efficiency. We've also introduced a ciphertext indexing method that facilitates the rapid retrieval of affected ciphertexts requiring re-encryption in the event of a policy update. Furthermore, we've substantiated the security of our proposed model within the given security model. As part of our future research, we intend to explore lightweight batch auditing and searchable encryption techniques optimized for use in the mobile-cloud computing domain.

REFERENCES

[1] M. Green, S. Hohenberger, and B. Waters, "Outsourcing the decryption of ABE ciphertexts," in *Proc. 20th USENIX Conf. Secur.*, 2011, p. 34.

[2] T. Elgamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Trans. Inf. Theory*, vol. IT-31, no. 4, pp. 469–472, Jul. 1985.

[3] M. Asim, M. Petkovic, and T. Ignatenko, "Attribute-based encryption with encryption and decryption outsourcing," in *Proc. 12th Austral. Inf. Secur. Manage. Conf.* ECU Security Research Institute, 2014, pp. 21–28.

[4] S. Fugkeaw, "A fine-grained and lightweight data access control model for mobile cloud computing," *IEEE Access*, vol. 9, pp. 836–848, 2021.

[5] R. Li, C. Shen, H. He, X. Gu, Z. Xu, and C.-Z. Xu, "A lightweight secure data sharing scheme for mobile cloud computing," *IEEE Trans. Cloud Comput.*, vol. 6, no. 2, pp. 344–357, Apr. 2018.

[6] S. Wang, H. Wang, J. Li, H. Wang, J. Chaudhry, M. Alazab, and H. Song, "A fast CP-ABE system for cyber-physical security and privacy in mobile healthcare network," *IEEE Trans. Ind. Appl.*, vol. 56, no. 4, pp. 4467–4477, Jul. 2020.

[7] S. Fugkeaw and H. Sato, "Improved lightweight proxy re-encryption for flexible and scalable mobile revocation management in cloud computing," in *Proc. IEEE 9th Int. Conf. Cloud Comput. (CLOUD)*, Jun./Jul. 2016, pp. 894–899.

[8] Z. Zhou and D. Huang, "Efficient and secure data storage operations for mobile cloud computing," in *Proc. 8th Int. Conf. Netw. Service Manag.*, Oct. 2012, pp. 37–45.

[9] Z. Li, W. Li, Z. Jin, H. Zhang, and Q. Wen, "An efficient ABE scheme with verifiable outsourced encryption and decryption," *IEEE Access*, vol. 7, pp. 29023–29037, 2019.

[10] Q. Li, Y. Tian, Y. Zhang, L. Shen, and J. Guo, "Efficient privacy-preserving access control of mobile multimedia data in cloud computing," *IEEE Access*, vol. 7, pp. 131534–131542, 2019.

[11] S. Zhang, W. Li, Q. Wen, H. Zhang, and Z. Jin, "A flexible KP-ABE suit for mobile user realizing decryption outsourcing and attribute revocation," *Wireless Pers. Commun.*, vol. 114, no. 4, pp. 2783–2800, Oct. 2020.

[12] Y. Kawai, "Outsourcing the re-encryption key generation: Flexible ciphertext-policy attribute-based proxy re-encryption," in *Proc. Int. Conf. Inf. Secur. Pract. Exper.*, vol. 9065. Cham, Switzerland, Springer, 2015, pp. 301–315.

[13] G. Yu, X. Zha, X. Wang, W. Ni, K. Yu, P. Yu, J. A. Zhang, R. P. Liu, and Y. J. Guo, "Enabling attribute revocation for fine-grained access control in blockchain-IoT systems," *IEEE Trans. Eng. Manag.*, vol. 67, no. 4, pp. 1213–1230, Nov. 2020.

[14] Y. Chen, J. Li, C. Liu, J. Han, Y. Zhang, and P. Yi, "Efficient attribute based server-aided verification signature," *IEEE Trans. Services Comput.*, vol. 15, no. 6, pp. 3224–3232, Nov. 2022, doi: 10.1109/TSC.2021.3096420.

[15] K. Yang, X. Jia, K. Ren, R. Xie, and L. Huang, "Enabling efficient access control with dynamic policy updating for big data in the cloud," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2014, pp. 2013–2021.

[16] L. Touati and Y. Challal, "Instantaneous proxy-based key update for CP-ABE," in *Proc. IEEE 41st Conf. Local Comput. Netw. (LCN)*, Dubai, United Arab Emirates, Nov. 2016, pp. 591–594.

[17] S. Fugkeaw and H. Sato, "Scalable and secure access control policy update for outsourced big data," *Future Gener. Comput. Syst.*, vol. 79, pp. 364–373, Feb. 2018.

[18] S. Fugkeaw, "A lightweight policy update scheme for outsourced personal health records sharing," *IEEE Access*, vol. 9, pp. 54862–54871, 2021.

[19] K. Yang, X. Jia, and K. Ren, "Secure and verifiable policy update outsourcing for big data access control in the cloud," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 12, pp. 3461–3470, Dec. 2015.

[20] J. Li, S. Wang, Y. Li, H. Wang, H. Wang, H. Wang, J. Chen, and Z. You, "An efficient attribute-based encryption scheme with policy update and file update in cloud computing," *IEEE Trans. Ind. Informat.*, vol. 15, no. 12, pp. 6500–6509, Dec. 2019, doi: 10.1109/TII.2019.2931156.

[21] Y. Guan, S. Guo, P. Li, and Y. Yang, "Secure and verifiable data access control scheme with policy update and computation outsourcing for edge computing," in *Proc. IEEE 26th Int. Conf. Parallel Distrib. Syst. (ICPADS)*, Dec. 2020, pp. 398–405, doi: 10.1109/ICPADS51040.2020.00060.

[22] S. Belguith, N. Kaaniche, M. Hammoudeh, and T. Dargahi, "PROUD: Verifiable privacy-preserving outsourced attribute based SignCryption supporting access policy update for cloud assisted IoT applications," *Future Gener. Comput. Syst.*, vol. 111, pp. 899–918, Oct. 2020.

[23] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proc. IEEE Symp. Secur. Privacy*, Oakland, CA, USA, May 2007, pp. 321–334.
[24] M. Gagné, S. Narayan, and R. Safavi-Naini, "Threshold attribute-based signcryption," in *Proc. Int. Conf. Secur. Cryptogr. Netw.*, 2010, pp. 154–171.
[25] S. Belguith, N. Kaaniche, M. Laurent, A. Jemai, and R. Attia, "Constant-size threshold attribute based signcryption for cloud applications, in *Proc. 14th Int. Conf. Secur. Cryptogr. (SECRYPT)*, vol. 6, 2017, pp. 212–225.
[26] C. Wang and J. Huang, "Attribute-based signcryption with ciphertext-policy and claim-predicate mechanism," in *Proc. 7th Int. Conf. Comput. Intell. Secur.*, Hainan, China, Dec. 2011, pp. 905–909.
[27] Y. Han, W. Lu, and X. Yang, "Attribute-based signcryption scheme with non-monotonic access structure," in *Proc. 5th Int. Conf. Intell. Netw. Collaborative Syst.*, Xi'an, China, Sep. 2013, pp. 796–802.
[28] J. Liu, X. Huang, and J. K. Liu, "Secure sharing of personal health records in cloud computing: Ciphertext-policy attribute-based signcryption," *Future Gener. Comput. Syst.*, vol. 52, pp. 67–76, Nov. 2015.
[29] *PBC (Pairing-Based Cryptography) Library*. Accessed: Jun. 30, 2021. [Online]. Available: https://crypto.stanford.edu/pbc/
[30] F. Deng, Y. Wang, L. Peng, H. Xiong, J. Geng, and Z. Qin, "Ciphertext-policy attribute-based signcryption with verifiable outsourced designcryption for sharing personal health records," *IEEE Access*, vol. 6, pp. 39473–39486, 2018, doi: 10.1109/ACCESS.2018.2843778.
[31] J. Yu, S. Liu, S. Wang, Y. Xiao, and B. Yan, "LH-ABSC: A lightweight hybrid attribute-based signcryption scheme for cloud-fog-assisted IoT," *IEEE Internet Things J.*, vol. 7, no. 9, pp. 7949–7966, Sep. 2020, doi: 10.1109/JIOT.2020.2992288.
[32] P. Sanchol, S. Fugkeaw, and H. Sato, "A mobile cloud-based access control with efficiently outsourced decryption," in *Proc. 10th IEEE Int. Conf. Mobile Cloud Comput., Services, Eng. (MobileCloud)*, San Francisco, CA, USA, Aug. 2022, pp. 1–8, doi: 10.1109/MobileCloud55333.2022.00008.
[33] E. Ahene, J. Walker, R.-M.-O. M. Gyening, G. Abdul-Salaam, and J. B. Hayfron-Acquah, "Heterogeneous signcryption with proxy re-encryption and its application in EHR systems," *Telecommun. Syst.*, vol. 80, no. 1, pp. 59–75, May 2022, doi: 10.1007/s11235-022-00886-2.
[34] Q. Huang, Z. Zhang, and Y. Yang, "Privacy-preserving media sharing with scalable access control and secure deduplication in mobile cloud computing," *IEEE Trans. Mobile Comput.*, vol. 20, no. 5, pp. 1951–1964, May 2021.
[35] S. Qi, W. Wei, J. Wang, S. Sun, L. Rutkowski, T. Huang, J. Kacprzyk, and Y. Qi, "Secure data deduplication with dynamic access control for mobile cloud storage," *IEEE Trans. Mobile Comput.*, early access, pp. 1–18, 2023, doi: 10.1109/TMC.2023.3263901.
[36] S. Banerjee, S. Roy, V. Odelu, A. K. Das, S. Chattopadhyay, J. J. P. C. Rodrigues, and Y. Park, "Multi-authority CP-ABE-based user access control scheme with constant-size key and ciphertext for IoT deployment," *J. Inf. Secur. Appl.*, vol. 53, Aug. 2020, Art. no. 102503, doi: 10.1016/j.jisa.2020.102503.
[37] X. Li, T. Liu, C. Chen, Q. Cheng, X. Zhang, and N. Kumar, "A lightweight and verifiable access control scheme with constant size ciphertext in edge-computing-assisted IoT," *IEEE Internet Things J.*, vol. 9, no. 19, pp. 19227–19237, Oct. 2022, doi: 10.1109/JIOT.2022.3165576.
[38] J. Li, Y. Wang, Y. Zhang, and J. Han, "Full verifiability for outsourced decryption in attribute based encryption," *IEEE Trans. Services Comput.*, vol. 13, no. 3, pp. 478–487, May 2020, doi: 10.1109/TSC.2017.2710190.
[39] J. Li, Y. Chen, J. Han, C. Liu, Y. Zhang, and H. Wang, "Decentralized attribute-based server-aid signature in the Internet of Things," *IEEE Internet Things J.*, vol. 9, no. 6, pp. 4573–4583, Mar. 2022, doi: 10.1109/JIOT.2021.3104585.

PATTAVEE SANCHOL received the bachelor's degree in computer engineering from Kasetsart University, Thailand, and the master's degree in engineering and technology from the Sirindhorn International Institute of Technology, Thammasat University, Thailand, in 2022. His research interests include cyber security, access control, cloud computing security, programming languages, and networking.

SOMCHART FUGKEAW (Member, IEEE) received the bachelor's degree in management information systems from Thammasat University, Bangkok, Thailand, the master's degree in computer science from Mahidol University, Thailand, and the Ph.D. degree in electrical engineering and information systems from The University of Tokyo, Japan, in 2017. He is currently an Assistant Professor with the Sirindhorn International Institute of Technology, Thammasat University. His research interests include information security, access control, cloud computing security, big data analysis, and high-performance computing. He has served as a Reviewer for several international journals, such as IEEE Access, the IEEE Transactions on Information Forensics and Security, the IEEE Transactions on Dependable and Secure Computing, the IEEE Transactions on Cloud Computing, the IEEE Transactions on Big Data, the IEEE Transactions on Network Science and Engineering, the IEEE Transactions on Network and Service Management, the IEEE Transactions on Parallel and Distributed Systems, *Computer & Security*, the IEEE Systems Journal, and *ACM Transactions on Multimedia Computing, Communications, and Applications*.